Direct-Inverse-Solver (DiInSo) 32-bit version

Quick (and easy) Guide

v. 1.0 (20.11.2019) — first public beta version

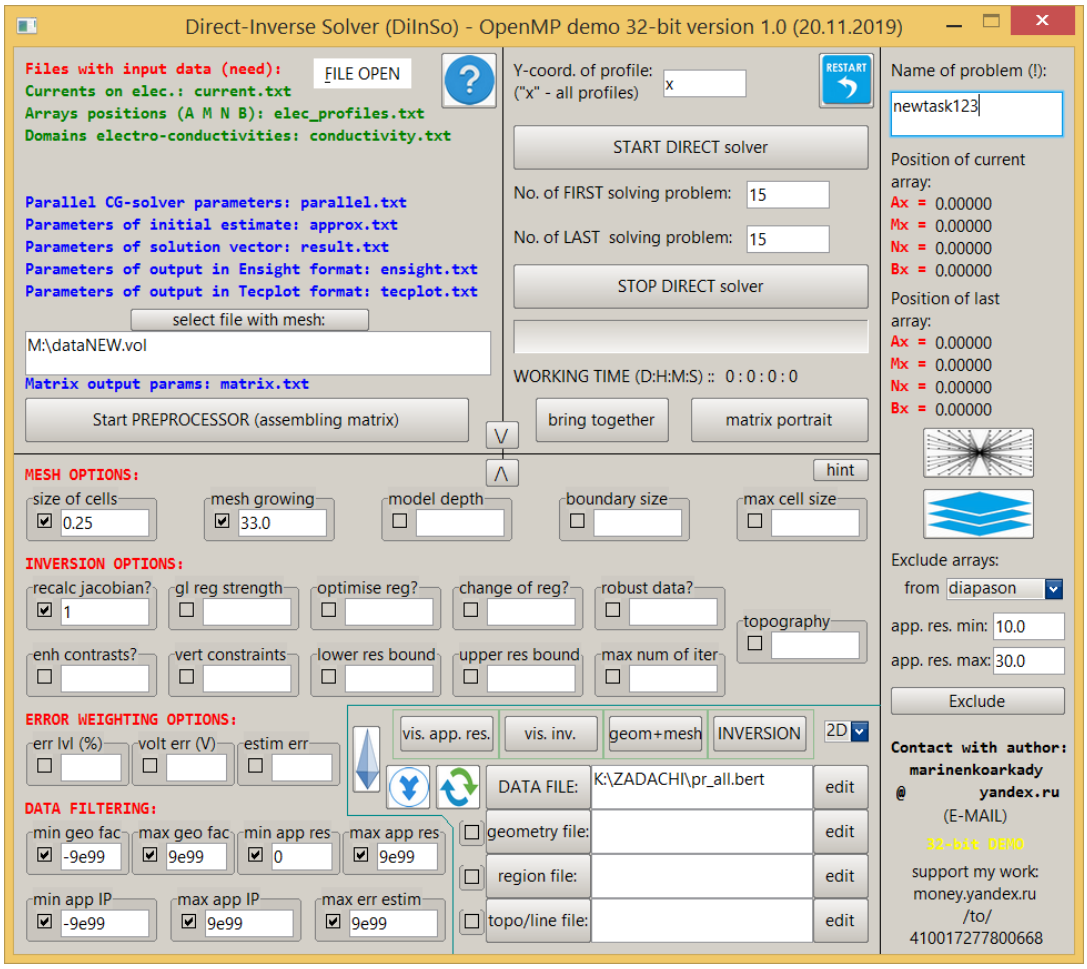
First of all, sorry for mistakes in English translation. It's not my native language. I hope you understand the meaning of the text and study my program without any problems. Truly yours, chief programmer Arkadiy Marinenko.

You are welcomed by the software package DiInSo (x32), intended for academic research, the purpose of which is the solution of direct problems and problems of inversion of electrical tomography on direct current. After reading this short manual, you will study the main features of the program and will be able to solve direct problems and problems of inversion of small and medium complexity. To solve more complex problems, you will need a little more time to study and understand the theory, as well as a 64-bit version of DiInSo. The text designed for its consistent reading, so it is undesirable to skip some chapters, this can make it difficult to understand what was wrote. So let's begin...

About the software package.

The DiInSo (x32) software package written on C/C++, compiled in a 32-bit environment, uses a large number of opensource projects, which will be mention below, and has no limitations, as well as guarantees, for distribution on the Internet and must be used solely for academic purposes.

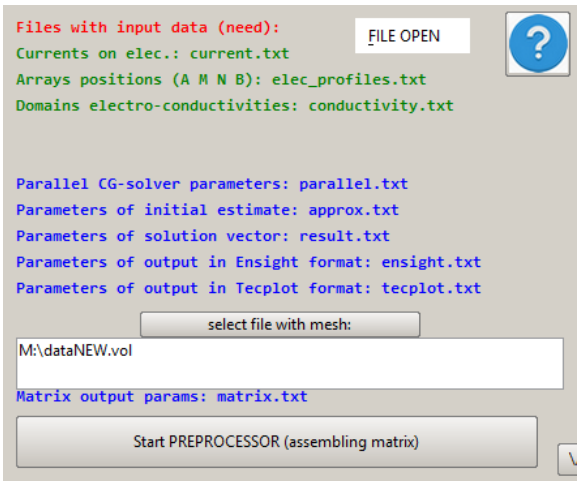
After launching the program using the Solver.exe executable file, you will see the main program window, which looks like this:



The main program window divided into several parts, each of which performs a specific function. Consider each of them in sequence.

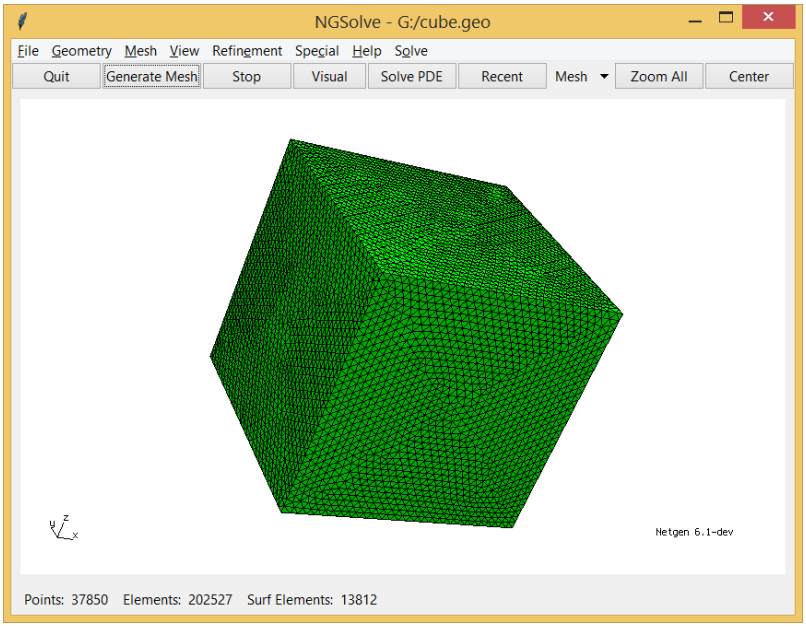
Preprocessor and input files.

The upper left part of the main window of the program contains information about the input files, the ability to edit them, select the file with mesh, the buttons for assembling the matrix and the help button:

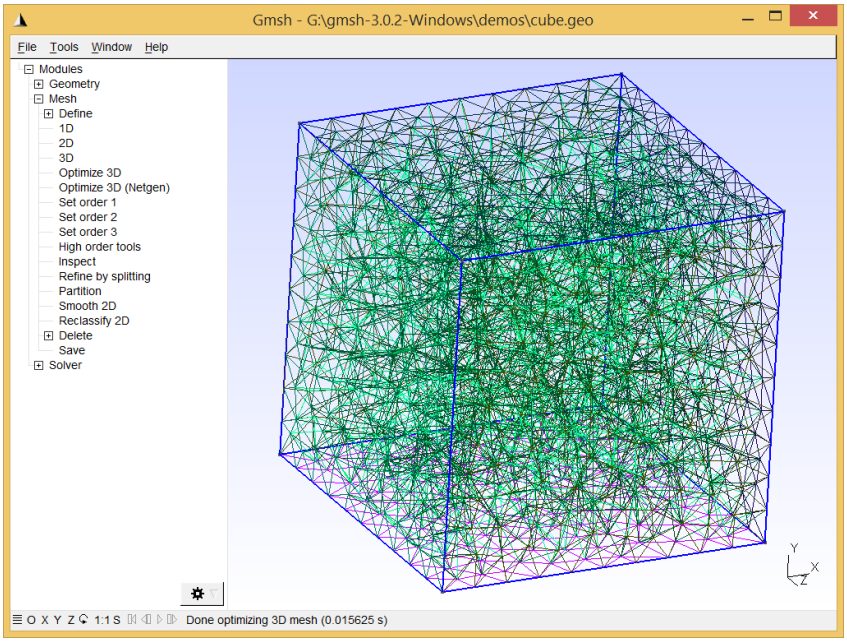


The “select file with mesh:” button allows you to select a file with a finite element mesh that will be use to solve direct problem on direct current. The solution of the direct problem carried out only in the three-dimensional domain defined by the tetrahedral mesh. To generate mesh, you need to use third-party software, some of which is free (that is, you will need to examine at least one tetrahedral mesh generator from the list below). The developed software package DiInSo supports the formats of the following mesh generators:

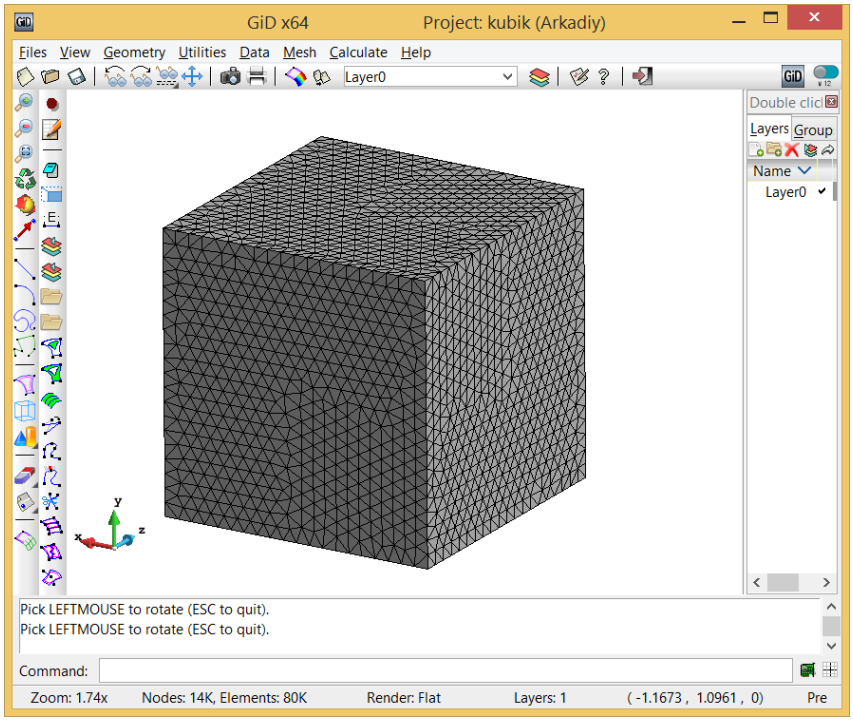
Netgen Mesh Generator (MS Windows and Linux) is a free opensource program distributed under the LGPLv2 license. Program website: https://ngsolve.org/



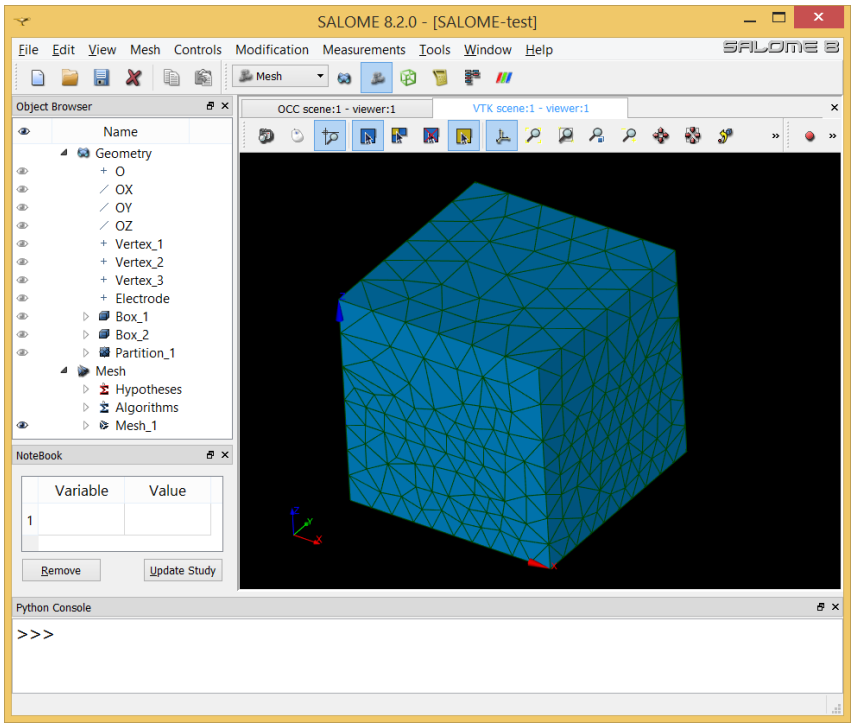
Gmsh (MS Windows and Linux) is a free opensource program distributed under the GPL license. Program website: http://gmsh.info/



GiD (MS Windows and Linux) is a commercial program with closed source code. Program website: https://www.gidhome.com/

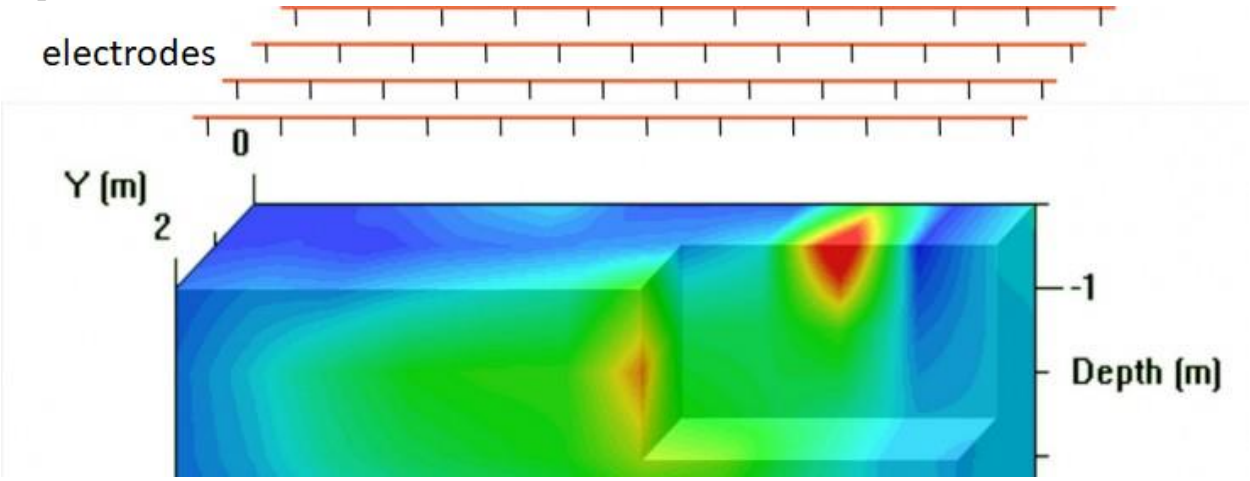


SALOME (MS Windows and Linux) is a free opensource program distributed under the LGPL license. Program website: http://www.salome-platform.org/



Let us deal with features of mesh generation. One of the main difficulties of mesh generation for problems of electrical tomography is the need to specify a

large number of electrodes located on the ground and sometimes with a small depth of them:



The fact is that not all mesh generators support work with such a primitive of geometry as a “separate node”. That is why, during the assign of electrodes, it is often necessary to go to tricks, such as adding to the geometry a set of squares whose vertices coincide with the nodes of location of the electrodes. So, IT'S IMPORTANT TO REMEMBER, the nodes of the electrodes must coincide with the nodes of the constructed mesh!

The next feature related to the external boundaries of the modeling area. Two cases are possible: if you plan to simulate electric field above the “air-ground” boundary (for example, when there are some interferences, as a metal objects, above the “air-ground” boundary), and if you DO NOT plan to do this. In the first case, all the external boundaries of the modeling area should be place at a distance from the current sources, where the resulting values of the electric field will be negligibly small, and the electrodes should be slightly bury in the ground to avoid modeling inaccuracies due to the influence of a strongly contrasting layer of air. In the second case, do the same with all external boundaries, except for the “air-ground” boundary, and in this case it is not necessary to bury the electrodes (there is no air — so there is no influence of a strongly contrasting layer). At all model far away from the sources external boundaries, so-called “Dirichlet zero boundary conditions” are set (remember this abbreviation), which means one thing for generate the mesh — you must mark these boundaries with a single number (the default is “76” — ▄▀▄▄ heptomino order from entertaining mathematics). So, IT'S IMPORTANT TO REMEMBER, if we plan to simulate the air, then we place far away all modeling area external boundaries to where the electric field is very small, and we bury the electrodes into the ground, but if we do not plan to simulate the air, then all modeling area external boundaries, except the “air-ground” boundary, placed far away, and we don’t bury the electrodes into the ground (we

don’t forget to mark the far away external boundaries with the same number — the default is “76”).

In addition, of course, do not forget that the size of the mesh is also of great importance. Usually the rule here is that small tetrahedra should be in regions where the field undergoes the greatest changes (for example, near sources), large tetrahedra can be in regions where the field does not undergo large changes (for example, far from sources).

Now let’s focus on each of the mesh generators separately in order to facilitate your work with them:

NetGen Mesh Generator (MS Windows and Linux) — you can specify nodes (the very ones where the electrodes are located) with the following line:

```
point (X, Y, Z);
```

where X, Y and Z are the coordinates of the node.

Gmsh (MS Windows and Linux) — you can specify nodes (the ones where the electrodes are located) with the following line:

```
Point (N) = {X, Y, Z, S};
```

where N is the node number, X, Y and Z are the coordinates of the node, an optional parameter S is the maximum size of the edges of the elements adjacent to this node.

GiD (MS Windows and Linux) — for the program to work correctly, it is necessary to determine the “type of task”. You can do this with the help of two files, in one of which materials (environments) specified, and in the other — boundary conditions (the modelling area external boundaries that we took far away from sources).

Example file with boundary conditions with number 76:

```
NUMBER: 76 CONDITION: Big_Bak
CONDTYPE: over surfaces
CONDMESHTYPE: over face elements
QUESTION: Zero
VALUE: 76
END CONDITION
```

Example file with materials with numbers 1, 2 and 3:

```
NUMBER: 1 MATERIAL: Air
QUESTION: Number
VALUE: 1
END MATERIAL
NUMBER: 2 MATERIAL: Earth
QUESTION: Number
VALUE: 2
END MATERIAL
NUMBER: 3 MATERIAL: Ferrum
QUESTION: Number
VALUE: 3
END MATERIAL
```

Because **GiD** does not have its own mesh data format as such, for the output of the mesh to the file we should use a specially generated “format file”, which for the presented software package is located in the folder with the DiInSo program

and is called **GID.bas**. This file should be place in the “templates” directory of **GiD** program.

SALOME (MS Windows and Linux) — this generator not able to save the mesh with reference to materials (environments) and boundary conditions, however it is able to save separate parts of the mesh into separate files. A similar procedure, for example, discussed in the following manual: http://www.elmerfem.org/elmerwiki/index.php?title=Multiple_bodies_from_Salome_to_Elmer

Therefore, the result of generating mesh in **SALOME** should be several files — for each individual material (environment) and for boundary conditions. All of these files, in turn, should be list in a file whose name MUST contain the word "SALOME" in capital letters or "salome" in lowercase letters. This file will be submit to the input as a file with mesh, its example shown below:

```
SALOME-CUBE.dat
2
SALOME-AIR.dat 0
SALOME-EARTH.dat 1
1
SALOME-BOUNDARY.dat 76

// FIRST LINE - file of the common mesh
// SECOND LINE - number of materials (environments) and, accordingly, number of files with materials (environments)
// NEXT LINES -
<material_(environment)_file>   <conductivity   number   (numbering   according   to   the conductivity.txt file)>
<material_(environment)_file>   <conductivity   number   (numbering   according   to   the conductivity.txt file)>
                        ...
<material_(environment)_file>   <conductivity   number   (numbering   according   to   the conductivity.txt file)>
// NEXT LINE - the number of files with boundary conditions (in this case always 1)
// NEXT LINES -
<file_with_boundary_condition>                                  <number_of_boundary_condition (76_mean_zero_Dirichlet_boundary_condition)>
<file_with_boundary_condition>                                  <number_of_boundary_condition (76_mean_zero_Dirichlet_boundary_condition)>
                        ...
<file_with_boundary_condition>                                  <number_of_boundary_condition (76_mean_zero_Dirichlet_boundary_condition)>
```

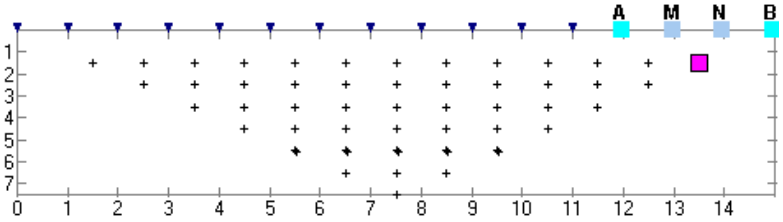
It is also IMPORTANT NOTE that the listed files MUST be located in the same directory as the file in which they listed.

So we used one of the generators, successfully obtained a mesh with the necessary materials (environments), with given zero Dirichlet boundary conditions and with electrodes that coincide with the mesh nodes. It's time to investigate the other input files. The 10 files listed in the FILE OPEN menu are: current.txt, elec_profiles.txt, conductivity.txt, parallel.txt, approx.txt, result.txt, ensight.txt tecplot.txt matrix.txt and null.txt — everything, besides the last, must be in the same folder with the program DiInSo and can be edited if desired and necessary. Let's go from first to last file:

CURRENT.TXT is a file containing currents in Amperes on the node electrodes of the array, for example:

```
2
-1.0
1.0
// FILE FORMAT: <number of currents> <currents>
// PS: If the current number is set to one,
// then its value will be the same on all electrodes.
```

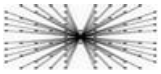
ELEC_PROFILES.TXT is a file containing the coordinates of the set of electrical tomography array positions for all profiles:

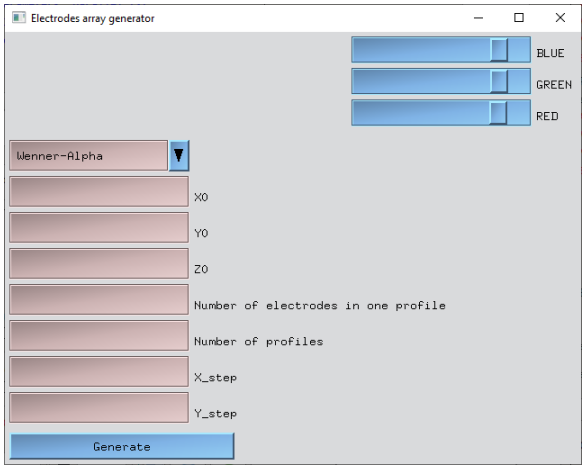


IMPORTANT: A and B electrodes ALWAYS means current electrodes, and M and N are ALWAYS potential electrodes.

The file format is as follows:

```
<number of electrical tomography array positions>
<A(X)> <A(Y)> <A(Z)> <M(X)> <M(Y)> <M(Z)> <N(X)> <N(Y)> <N(Z)> <B(X)> <B(Y)> <B(Z)>
<A(X)> <A(Y)> <A(Z)> <M(X)> <M(Y)> <M(Z)> <N(X)> <N(Y)> <N(Z)> <B(X)> <B(Y)> <B(Z)>
          ...
<A(X)> <A(Y)> <A(Z)> <M(X)> <M(Y)> <M(Z)> <N(X)> <N(Y)> <N(Z)> <B(X)> <B(Y)> <B(Z)>
```

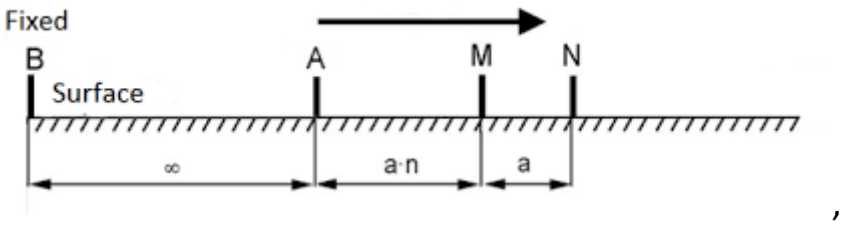
IMPORTANT: it is not necessary to generate this file manually. For automatic generation of file for several types of electrical tomography arrays, look to the right part of the main program window DiInSo. There you can see button  that calls the window:



"BLUE", "GREEN", "RED" sliders in this and similar windows serve to change the background color of the window and do not affect anything else (try it!). Setting the electrical tomography array positions is extremely simple: choose the array type (Wenner-Alpha, Wenner-Beta, Schlumberger, Dipole-Dipole, Pole-Dipole) -> set the position (coordinates) of the first electrode (X0, Y0, Z0) -> set

the number of electrodes in one profile (Number of electrodes in one profile) -> set the number of profiles (Number of profiles) -> set the step between the electrodes in one profile (X_step) -> set the step between the profiles (Y_step) -> press the button “Generate” and your elec_profiles.txt file is ready (it will appear in the same folder as the DiInSo program).

IMPORTANT NOTE: if electrode **B** is at a great distance from the remaining electrodes (the so-called three-electrode arrays):



,

then in the elec_profiles.txt file, you must specify the X coordinate of this electrode as a large number (more than 1.000000e + 100) modulo (for example, – 1.000000e + 300 or 1.000000e + 300).

CONDUCTIVITY.TXT is a file containing the electrical conductivities of the corresponding materials (environments) in S/m, for example:

```
4
0.0
1.0E-12
0.01666666666
7690000.0
// The electrical conductivity is given as follows:
<number of electrical conductivities>
<NULL (for zero material (environment)) 0 electrical conductivity> - DO NOT FORGET that the numbering comes from ZERO
<electrical conductivity of material (environment) 1>
<electrical conductivity of material (environment) 2>
...
<electrical conductivity of material (environment) n>
```

IMPORTANT NOTE: If there are NO materials (environments) with specific numbers, you can set any value in these positions, but ignore these values NOT allowed!

PARALLEL.TXT is a file containing parameters of a parallel solver that will be use to solve direct problems. It contains four numbers in the following sequence: the convergence criterion for the absolute residual, the convergence criterion for the relative residual, the divergence criterion, the maximum number of iterations. Usually it does not make sense to change those values that are set by default (1.0e-8 1.0e-6 1.0e+8 100000). The OpenMP approach is use as the parallelization algorithm, and the conjugate gradient (CG) method with the so-called LU preconditioner is use as a solver.

APPROX.TXT is a file in which you can change an initial approximation for a solver other than zero initial approximation, for example:

```
1
```

```
result415.sol
//
FIRST LINE:
0 or less - do not change initial approximation
(use zero initial approximation);
1 or more - change initial approximation from the file;
SECOND LINE:
initial approximation file.
```

The initial approximation file in this case should have the format:

```
<number of the vector element, starting with ZERO>    <vector element>
<number of the vector element, starting with ZERO>    <vector element>
...
<number of the vector element, starting with ZERO>    <vector element>
```

The initial approximation file does not have to contain all the elements of the vector. Not contained elements will be consider as zero.

RESULT.TXT is a file indicating whether each complete vector solution should be output to a separate file (for example, to analyze the solution result). 1 is need, 0 is not need.

ENSIGHT.TXT and TECPLOT.TXT — these files indicate whether to output the solution results (FIRST in a set of solutions, because outputting all the files of a set of solutions takes too much time and too much disk space) to separate files compatible with Ensigh (https://www.ensight.com/) and Tecplot (http://www.tecplot.com/) visualizers, respectively (for example, to analyze a solution results). The file format is the same and looks like:

```
0
filename.mod
//
FIRST LINE - Output data in EnSight/Tecplot format or not? 1 is need, 0 is not need.
SECOND LINE - The name of the file with the mesh (that is, the mesh that was select to the
input).
```

MATRIX.TXT is a file that indicates whether the generated matrix should be output to a special CSR-like (Compact Sparse Row) format with one-dimensional arrays, each in a separate file (as before, 1 is need, 0 is not need), which may be necessary for data analysis and/or to solve the problem in a third-party solver:

di.txt — file with diagonal elements;

ggl.txt — non-zero elements of the matrix below the main diagonal (in the so-called "lower triangle");

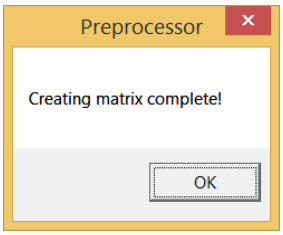
igc.txt — igc [0] = 0, igc [i] = igc [i-1] + (the number of non-zero elements in the i-th row of the lower triangle); the rows numbered from ZERO;

jgc.txt — contains the indices of the columns in the lower triangle of each non-zero element of the lower triangle; the columns numbered from ZERO.

IMPORTANT: By default (and always), the matrix is output in two formats — binary (matr.dat file), which cannot be read by simple text editors (it is needed for the DiInSo program), and Matrix Market format (matr.mtx), which can be read by simple text editors. You can learn more about Matrix Market format, for example, here: http://networkrepository.com/mtx-matrix-market-format.html

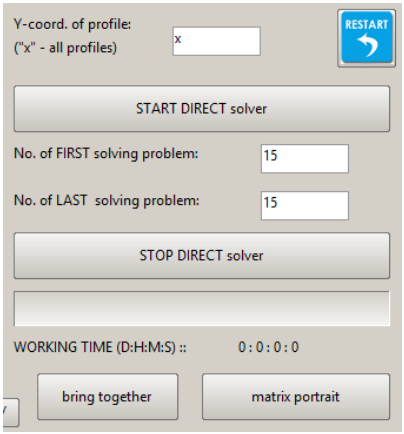
NULL.TXT is an optional file that changes the number of Dirichlet's zero boundary conditions (which we placed far away from current sources). It contains a single number — the new number of the zero Dirichlet boundary conditions (which, by default, are 76).

There are only two buttons left in this part of the main window that we did not consider. The button with a question mark opens the help file that you are currently reading. Finally, the main button here is "Start PREPROCESSOR (assembling matrix)". After pressing it, the process of generating a matrix for solving direct problems will begin. As mentioned above, by default the matrix will be stored in the matr.dat and matr.mtx files. As soon as the process is completed, you will see a message:



Press "Ok" and proceed to the next part of the main program window, which is responsible for solving direct problems.

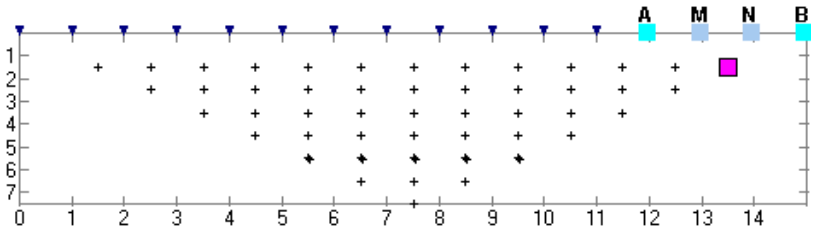
The solution of direct problems.



This part of the program contains entry fields for selecting the Y-coordinate of the profile, the first and last problems which will be solve, the solver start button, the solver stop button, the compilation of the results for further solving the inversion problem, launching the matrix portrait visualizer, restarting the entire program and progress bar time counter. Now look all of this in details...

At the very top of this part of main window, you can see the field for entering the Y-coordinates of the profile (or you can enter the letter "x" without quotes to solve direct problems on all profiles). Please note that in solving direct problems,

the various electrical tomography profiles are “lined up” in a lines with Y = const. Since the solution of the direct problem involves moving the electrical tomography array position:



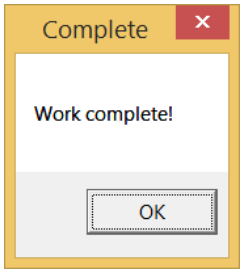
we have to solve not one, but many problems that, by the way, can have the same mathematical solutions in case of coincidence of the positions of the current electrodes (in this case, the program DiInSo will automatically skip the solver cycle and use the existing solution). Sometimes solving process takes a lot of time, so we cannot solve all tasks on all profiles at once, but solve a direct problem on one of the profiles in order to save time and think about the expediency of solving the problem on other profiles. This is the purpose of the input field of the profile coordinates.

Below there are two more input fields — the first problem to be solved and the last problem to be solved. The question arises, where does the concept of the “first” and “last” problems come from, because we can choose an arbitrary profile? Let's go back to the file elec_profiles.txt, which lists many electrical tomography array positions on different profiles. If we consistently view this file, then the first array position that occurred on this profile will be the “first” problem on this profile. Accordingly, when solving a problem on all profiles at once, the “first” task coincides with the first one in the list in elec_profiles.txt file. There is another question, whether you need to know how many problems on a profile (or on all profiles), in order to correctly set the number of the last problem. The answer is NO, this is not require. In the field of the “last” problem, you can specify an arbitrary large number, and the program after the start of the solver will correct it to the correct one.

So, we decided on which profile we will solve the problem (or on all profiles at once), and numbers of first and last solving problems. You can begin the solving process. Press the button "START DIRECT solver" and ... the process begins. The problem counter in the “first” problem field will start to move forward, the progress bar will be fill by green color line, and time will be count. During the solution, a large number of files of the form <number> .sol will be create in the DiInSo program folder. These are files containing information about the solved problem with a certain <number>. Inside these files, numbers of current and potential electrodes in the global numbering of mesh nodes and electric potential

values measured on potential electrodes. Do not worry, later we will “clear” the folder with the DiInSo program from them.

As mentioned above, the solution process can be delayed for many hours and even days (but more often the hours), but computer time can be valuable. Therefore, it is possible to stop a solution at any stage using the “STOP DIRECT solver” button. In this case, the solution of the current problem with the current number is completed and the general solution process is interrupt, as evidenced by the informational message:



A similar message will appear when the solving all problems are completed. You can resume solving process from the same place where the process was stop by pressing the “START DIRECT solver” button again.

So we have solved all the necessary problems on all the profiles we need, the folder with the program filled with files like <number>.sol, what to do next? Next, it is necessary to “composing” all these files, that is, to assemble a file for solve the inversion problem (recall that the goal of solving the direct problem of electrical tomography is to simulate a measurement situation on a certain modelling area where the inversion problem solved). You can done this using the "bring together" button. A special message will be inform about the completion of the composing process:

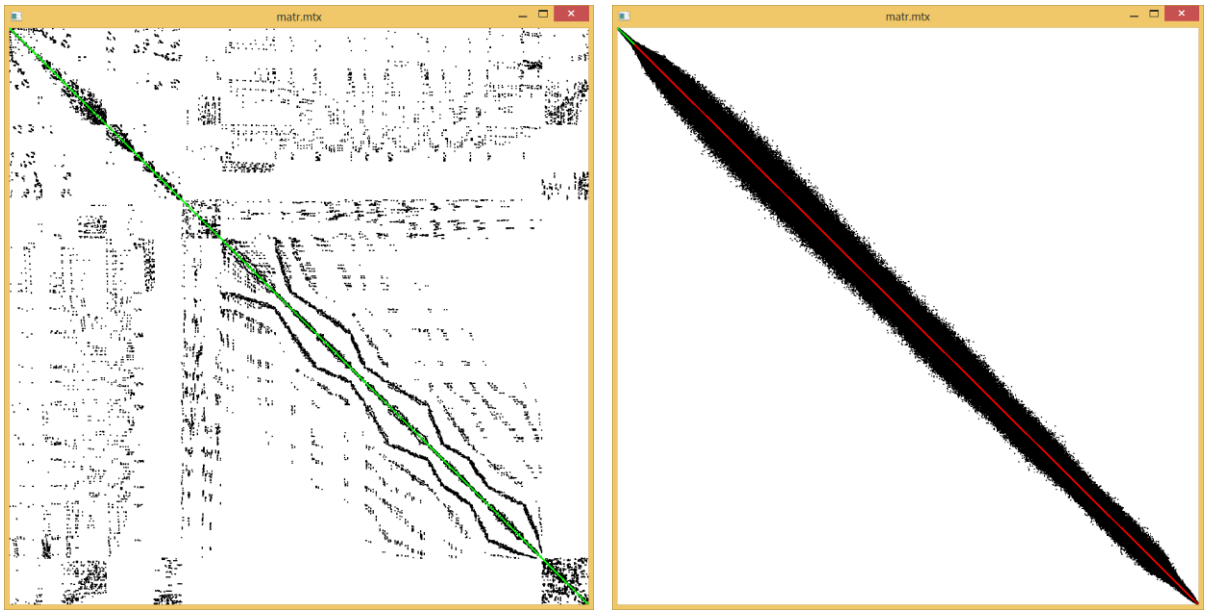
As a result, files for inversion will be create in the folder with mesh file (!). There are will be files for performing 2D inversion on separate profiles with the names profile<number>.bert, as well as the file for performing 3D inversion with name reverse_data.bert. Also will be create files for performing 2D inversions in third-party programs: IP4DI (https://github.com/cageo/Karaoulis-2013) — files looks like profile<number>.d and ZondRes2D (http://zond-geo.ru/software/resistivity-imaging-ves/zondres2d/) — files looks like profile<number>.z2d, as well as for performing 3D inversions in third-party

programs: IP4DI (https://github.com/cageo/Karaoulis-2013) — file reverse_data.d and RES3DINV (http://www.geotomosoft.com/) — file reverse_data.dat. However, the complete compatibility of these files for third-party programs is NOT guarantee.

IMPORTANT NOTE, if some solving results are not present (problems with certain numbers are not present), the resulting files for inversions may not be completely filled and even have incorrect data, watch this carefully.

It remains to consider two more buttons in this part of the main window of the program DiInSo — “matrix portrait” and “RESTART” 

. The first allows you to graphical evaluate the so-called “portrait” of the matrix, that is, the location of non-zero elements within the matrix. Why is it important? The fact is that the speed of the iterative solvers depends, among other things, on the “spread” of non-zero elements within the matrix. The more “chaotic” this “spread” is, the slower the problem will be solve. Consider two “portraits”:

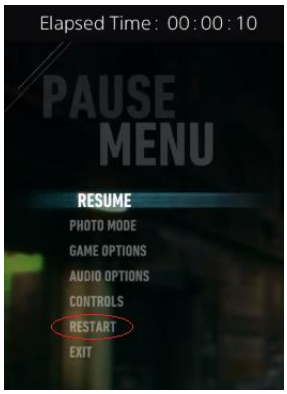


Diagonal elements are highlight in green and/or red colors. In the case of red dots, the diagonal elements are close to zero, which also adversely affects the rate of convergence of iterative solvers. In the presented example, in the left “portrait” all diagonal elements are marked in green (that is, they are NOT close to zero), but non-zero elements outside the diagonal are too far apart. The right “portrait” has a small spacing of nonzero elements, but on the diagonal, there are many elements close to zero. Eliminate a large spacing of nonzero elements can be done by renumbering the nodes of finite elements (for example, by choosing another mesh generator, where this feature is taken into account). Eliminating near-zero values on the diagonal is usually impossible, since they appear due to the strong contrast of the model’s materials (environments). So why do you need all this? Suppose

you are face with a slow convergence of solutions of direct problems. Look at the portrait, maybe you need to use another mesh generator or think about simplifying the model.

Finally, the "RESTART" button: 

... Let's talk a little about programming. One of the most typical programming errors in such splendid, but at the same time difficult languages like C/C++ are errors with the so-called “memory leak”. These errors are the presence of occupied computer’s random access memory, which was not free in time from the already unnecessary data, in short — the memory is occupy by unnecessary data. Due to such errors, the program begins to occupy more and more places in the computer’s random access memory until it takes up all that is available (usually it ends with “termination” of the program). No program written in C/C++, which insured against such errors. You may even have seen the button or the “restart” option in other programs (you may know where this screenshot from):

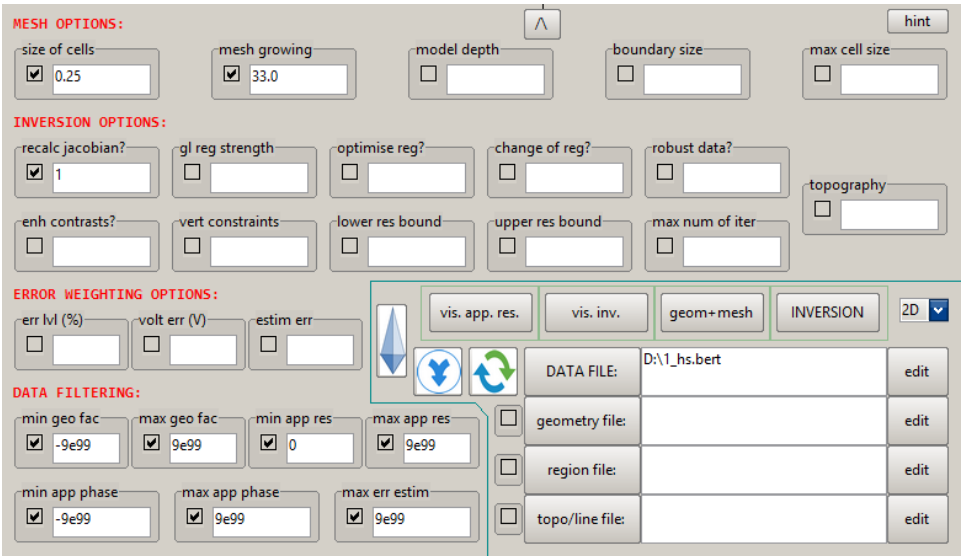


(no, this is not restarting the mission — this is restarting the program)

The purpose of this button is to clear the PC memory of possible “leaks”. Using this button too often does not make sense, but sometimes it can help you (for example, after a very long time working with the program).

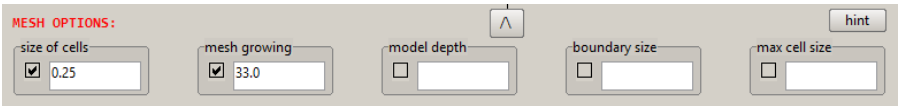
Go to the next part of the program, located at the bottom of main DiInSo window. It is responsible for solving inversion problems.

Solving inversion problems.



Despite the deterrent appearance of this part of the program, most of the functions and settings here are perform in automatic mode and you will need it only if you are unable to obtain an acceptable result in automatic mode. The most important button of this part of the program is the space selection button — 2D or 3D. Two-dimensional inversion requires less memory, runs faster, but may sometimes be insufficient to assess the restored environment. Three-dimensional inversion is more complicated, slower and more demanding on the amount of free computer's random access memory. However, choosing a three-dimensional inversion, when you have only one profile does not make sense. Yes, and for several two-dimensional profiles, it is sometimes possible to reconstruct the three-dimensional model with a sufficient degree of accuracy. Anyway, it is IMPORTANT to REMEMBER, the settings for 2D and for 3D inversions are slightly different, and therefore we start with the settings for 2D inversion, and then consider those settings for 3D inversion that have differences, mentioning those settings that have similarities.

Five 2D options associated with the generated mesh for the inversion problem (not to be confused with the mesh that is generate for the direct problem):



The first option "size of cells" is the size of edge cells (in the distances between the electrodes) on the surface from greater than 0.0 to 0.5. For example, a value of 0.25 will give you about 4 cells (because 1/0.25 = 4) between adjacent electrodes. Why "about"? Because the mesh generation algorithm cannot always build a mesh, that satisfies certain conditions (but it will try).

The second option “mesh growing” is the rate of increase in cell size (triangles) with depth (33 — fast, 35 — slow). It is recommend choosing this option empirically. First try to solve the problem on a coarse mesh (that is, when the cells (triangles) quickly increase with depth), and then increase the mesh growing parameter, while increasing the inversion accuracy to the required (however, the requirements for computer’s random access memory increase too).

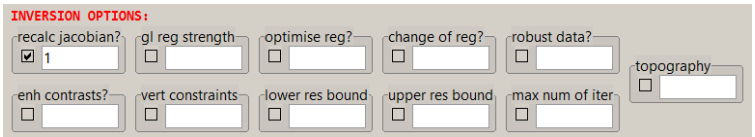
The third option “model depth” is the depth of research in model units of measurement (usually meters) that you can specify. It can be any, even a few kilometers. Nevertheless, remember that if the location of the electrodes is such that the tomography could be “seen” the maximum, for example, 10 meters, then at a depth of 100 meters you will get an incorrect result, which, moreover, will “noise” the result at those depths where it should be correct. So work with this option carefully and do not forget that the program can determine the depth of the method automatically (if you do not enable this option).

The fourth option “boundary size” is the distance of the boundaries from the electrodes in percent. In the case of the inversion, we solve the problem in one degree or another similar to the direct problem and the rules in it are the same — the boundaries must be sufficiently far away from the electrodes. Let the set of electrodes be a line 100 meters long, and then the distance of the boundaries by 500% implies a distance of 500 meters from the boundary to the nearest electrode. If the distance from the boundary to the nearest electrode is not enough, the solution may be incorrect, as in the case with the direct problem.

The fifth option “max cell size” is the maximum cell (triangle) size in the sense of cell area. If you do not want your cells to grow too fast with depth, you can limit their maximum size with this option. However, do not forget the rule — a smaller cell size means greater demands on the amount of computer’s random access memory.

Finally, the small “hint” button allows you to get a brief help on all the options in this part of the main DiInSo window (this is useful because there are a lot of them and you can easy forget them).

Eleven 2D options directly related to inversion:



The “recalc jacobian?” option — whether Jacobian (the determinant of the Jacobi matrix) is recalculate at each iteration step? 1 — yes, 0 — no. What is good in recalculation of Jacobian at every step? The rate of convergence increases in terms of the number of iterations. What is bad with recalculation of Jacobian at

every step? The Jacobian recalculation itself takes a long time in the case of a difficult problem.

The option “gl reg strength” is the value of the regularization parameter. When you perform an inversion procedure, then you solve a problem that can have many “correct” (from a mathematical point of view) solutions. Therefore, some additional parameters are introduce that limit the final solution in some “scopes”. The regularization parameter can have both very small values (for example, 1) and very large values (for example, 1000). What is the difference? With some values (usually small values) result of inversion give sufficiently accurate geometry of the underground structures, but the resistances of these structures could be far from real. Other values (usually large values) strongly diffuse the geometry of the underground structures, but at the same time provide accurate data of the resistances due to a smaller scatter of values. You can find a compromise between these two cases. Very often a compromise means the number 20 (as it happened), but this does not mean that it will always give you the desired result. Experiment!

The option “optimize reg?” is the selection of the regularization parameter, described above, using the method of so-called L-curves. 1 — use the method of L-curves, 0 — do not use the method of L-curves. This method will try to find a regularization parameter for you that will be a compromise in terms of geometry accuracy and resistance values. However, it is worth noting that this method is often mistaken with the selection, so be careful when using it.

The “change of reg?” option is to change the regularization parameter mentioned above at each new iteration step using multiplication operation. For example, if the regularization parameter 100 is set in the option “gl reg strength”, then the option “change of reg?” as a value of 0.5 will reduce the regularization parameter on the second iteration twice (to 50), on the third iteration will reduce twice again (to 25) and so on. Why do you need it? Some complex models allow us to achieve acceptable inversion results only using of such manipulations.

The option “robust data?” is a flag indicating the necessity of evaluating data for the presence of noise: 1 — data should be evaluated for the presence of noise, 0 — data should not be evaluated for the presence of noise. What does this mean? Suppose your measured data for solving the problem of inversion were very noisy and you know about it. By turning on this option, you turn on the data estimation algorithm, which automatically drops or changes the weight of those values that are clearly out of the general trend. If you are sure that the measured data have a high degree of accuracy (for example, if they are obtain by numerically solving a direct problem), then turn on this option is NOT recommend.

The option “enh contrasts?” is a flag indicating the need to “enhance the contrast” between the individual subdomains (environments): 1 — enhance the

contrast, 0 — not enhance the contrast. What does it mean? The solution of the inversion problem is continuous, that is, you will always observe smooth transitions (in resistance sense) between different subdomains (environments). The realities of life are slightly different — a huge metal block can be bury in the ground, for example — that is sharp transition between subdomains (environments). The turn on of this option sometimes allows you to more clearly identify some subdomains (environments), the transition to which is carry out by abruptly values of resistance. However, do not think that this option is a panacea, since it does not cancel the continuity of the result of solving the inversion problem.

The “vert constraints” option — this option is somewhat similar to the previous one and serves as a message for the data inversion algorithm regarding the vertical boundaries of the subdomains (environments). 1 indicates isotropy (i.e., continuity) of the environment, 0.1 indicates that we have vertical subdomains (environments) boundaries that are somewhat diffused, and 0.01 indicates that the vertical boundaries of the subdomains (environments) should stand out quite accurately, that is, the farther away the value “vert constraints” is from 1, the more clear should be the vertical boundaries. Most importantly, do not forget that the term “stratified medium” or “layered medium” in geophysics, as a rule, implies a horizontally layered medium and vertical layers, as a rule, are present where there are some inclusions (anomalies) in the environment. You do not need to think that in a strictly horizontally layered environment, this option should be equal to 1, since “searching” for vertical boundaries (in the case of values less than 1) means only reducing the “diffusion” of the inversion results and more clearly highlighting possible anomalies. Often, on the contrary, it “helps” to better distinguish a purely horizontally layered medium, even though it lose the accuracy of resistance values (as blurring decreases).

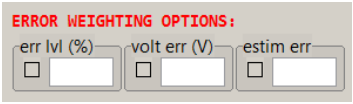
The options “lower res bound” and “upper res bound” indicate the minimum and maximum resistance values in the solving inversion problem. That is, if you precisely know the range of possible resistance values, then you can specify it; otherwise, it will be from zero to a very large value. However, you are not expect that as a result you will receive a solution in the exact range that was specify. It may happen that the inversion method does not allow to "fitting" the solution in your range, although, of course, the solution will be as close as possible to your range.

The “max num of iter” option — you can limit the maximum number of iterations for solving the inversion problem using this option, but it is NOT recommended to do this, it is better to wait for the cycle to exit according to the convergence criteria, so the solution of the inversion problem will be more

accurate. The meaning of using this option is only in one case — you want to see the solution at a certain iteration in order to observe how the solution changes with each iteration.

The option “topography” indicates that it is necessary to take into account the “topographic effect”, namely, to take into account that the surface of the earth on which the measurements were made is NOT flat: 1 — take topography into account, 0 — do not take topography into account. For each array of electrical tomography, there is such a parameter as “geometric factor”. This parameter is calculate in relation of the distances between the current and potential electrodes. When the surface is not flat, it is undesirable to calculate this coefficient by the traditional method, because the electrodes are at different heights relative to each other. The greater the irregularity of the ground, the more the actual geometric factor may differ from the calculated one. By enabling this option, the geometric factors and the model mesh will be recalculate based on the geometry features. On the other hand, the turn on of this option rather complicates the solution algorithm and the requirement for the free computer’s random access memory. On the third hand, not enabling this option in the presence of a topographical effect can lead to the fact that the problem cannot be solve at all. On the fourth hand, if the geometric factor was initially calculated taking into account the effect of topography and, accordingly, taking into account the apparent resistances, then a new recalculation of these parameters will only garble the result. Therefore, you should always understand exactly what data you submit to the input.

Three options for 2D and 3D inversions related to errors:

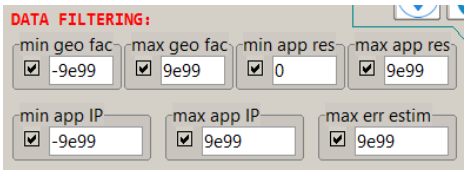


Option “err lvl (%)” — here you can specify the approximate average accuracy in percent of the measurements taken. For example, you are sure that due to noise during the measurements, the data you receive may deviate from the real ones by an average of 10%, so enter “10” in this field without quotes.

The option "volt err (V)" is similar to the previous case, only for the measured electric potentials (on the potential electrodes) in Volts. If you know that your measurements deviate from the real ones by an average of 0.1 V, then enter in this field "0.1" without quotes.

The “estim err” option is an option flag that indicates whether measurement errors should be evaluated algorithmically: 1 — errors should be evaluated, 0 — errors should not be evaluated. Note that measurement errors are not evaluated only if information about them is precisely known in advance and is contained in the input * .bert file (in the “err” column).

Seven 2D and 3D inversions options associated with the filtering of existing and received data:



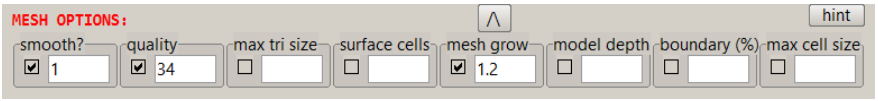
“min geo fac” and “max geo fac” options — electrical tomography arrays that have a geometry factor not from this range will be exclude from the calculations.

The options “min app res” and “max app res” — apparent resistances that are not from this range will be exclude from the calculations.

The options “min app IP” and “max app IP” — apparent values of induced polarization that are not from this range will be exclude from the calculations.

The option “max err estim” — data, errors of which are estimated above this value, will be exclude from the calculations.

Eight 3D options associated with the generated mesh for the inversion problem (not to be confused with the mesh that is generate for the direct problem):



The "smooth?" option is an option flag that indicates the need to smooth the surface mesh, that is, the mesh where current and potential electrodes located. 1 — smooth, 0 — do not smooth. In fact, this option is important only in the presence of a topographical effect, as it allows obtaining a smoother surface without “angularities”, which may be present if this option was not enable.

The “quality” option is the quality of the surface mesh (not the entire mesh, but only the surface mesh). 34 is good, 30 is bad. Please note that the quality of the surface mesh largely affects the quality of the rest of the mesh, since the surface (triangular) mesh is initially build, and then a volumetric (tetrahedral) mesh is obtained from it.

The “max tri size” option is the maximum size of the cell (triangle) of the surface mesh in the sense of the cell area.

The “surface cells” option — this option to some extent duplicates the previous option, setting the cell size of the surface mesh, but only in terms of the size of the edge. Values are set in the range from more than 0.0 to 0.5. What's the point? During the construction of the mesh (including the surface one), the mesh generator goes through several stages — from a coarse mesh constructed from reference points, which is unsuitable for calculations, to a more or less “adequate” mesh in terms of calculations. The rebuilding of the mesh occurs by splitting the

coarse mesh. This value will indicate the step of this partition in the units that are used (usually these are meters). The maximum value of 0.5 was chose based on the logic of the partitioning calculations, since 0.5 is the partitioning of the rough "two" step. In many ways, this option is similar to the "size of cells" option for 2D mesh. Nevertheless, not one to one, since here we are dealing with triangles on the surface with all the consequences.

The "mesh grow" option is the rate of increase in cell size of tetrahedra with depth (2 — fast, 1.1 — slow). Here it is recommend choosing this option empirically. First try to solve the problem on a coarse mesh (that is, when the cells quickly increase with depth), and then increase the mesh growing parameter, while increasing the inversion accuracy to the required (but also increasing the requirements for computer's random access memory). Why are the values for this option different from those for 2D inversions? This is because different generators using for building meshes in 2D and 3D cases with their own characteristics and, accordingly, their own parameters.

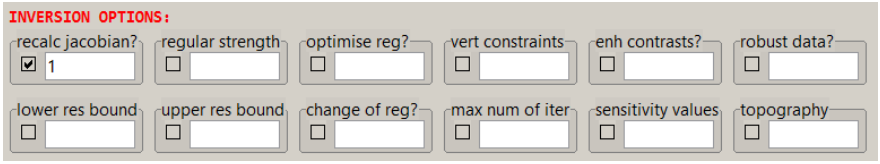
The "model depth" option is the depth of research in model units of measurement (usually meters) that you can specify. It can be any, even a few kilometers. Nevertheless, remember that if the location of the electrodes is such that tomography "seen" a maximum of 10 meters, then at a depth of 100 meters you will receive an incorrect result, which, moreover, will "noise" the result at those depths where it should be correct. So work with this option carefully and do not forget that the program can determine the depth of the method automatically (if you do not enable this option).

The option "boundary (%)" — the distance between the boundaries from the electrodes in percent. In the case of the inversion, we solve the problem in one degree or another similar to the direct problem and the rules in it are the same — the boundaries must be sufficiently far away from the electrodes. Let the set of electrodes (profiles) will be a square with a side of 100 meters, so the distance of the boundaries by 500% implies a distance from the boundary to the nearest electrode of 500 meters. If the distance from the boundary to the nearest electrode is not enough, the solution may be incorrect, as in the case with the direct problem.

The option "max cell size" is the maximum size of a cell (tetrahedron) in the sense of cell volume. If you do not want your cells to grow too fast with depth, you can limit their maximum size with this option. However, do not forget the rule — a smaller cell size means greater demands on the amount of computer's random access memory.

There are twelve 3D options associated directly with inversion (note that many options are similar in 2D and 3D cases, but they will be duplicated here for easy reading of the manual; in addition, similar options may have different

implementation algorithms in 2D and 3D cases, therefore do not think that this is the same thing:



The “recalc jacobian?” option (similar to the “recalc jacobian?” option for 2D) — whether Jacobian (the determinant of the Jacobi matrix) is recalculate at each iteration step? 1 — yes, 0 — no. What is good in recalculation of Jacobian at every step? The rate of convergence increases in terms of the number of iterations. What is bad with recalculation of Jacobian at every step? The Jacobian recalculation itself takes a long time in the case of a difficult problem.

The “regular strength” option (similar to the “gl reg strength” option for 2D) is the value of the regularization parameter. When you perform an inversion procedure, then you solve a problem that can have many “correct” (from a mathematical point of view) solutions. Therefore, some additional parameters are introduce that limit the final solution in some “scopes”. The regularization parameter can have both very small values (for example, 1) and very large values (for example, 1000). What is the difference? With some values (usually small values) result of inversion give sufficiently accurate geometry of the underground structures, but the resistances of these structures could be far from real. Other values (usually large values) strongly diffuse the geometry of the underground structures, but at the same time provide accurate data of the resistances due to a smaller scatter of values. You can find a compromise between these two cases. Very often a compromise means the number 20 (as it happened), but this does not mean that it will always give you the desired result. Experiment!

The “optimize reg?” option (similar to the “optimize reg?” option for 2D) is the selection of the regularization parameter, described above, using the method of so-called L-curves. 1 — use the method of L-curves, 0 — do not use the method of L-curves. This method will try to find a regularization parameter for you that will be a compromise in terms of geometry accuracy and resistance values. However, it is worth noting that this method is often mistaken with the selection, so be careful when using it.

The “vert constraints” option (similar to the “vert constraints” option for 2D) — this option is somewhat similar to the “enh contrasts?” option, which is described immediately after this option, and serves as a message for the data inversion algorithm regarding the vertical boundaries of the subdomains (environments). 1 indicates isotropy (i.e., continuity) of the environment, 0.1

indicates that we have vertical subdomains (environments) boundaries that are somewhat diffused, and 0.01 indicates that the vertical boundaries of the subdomains (environments) should stand out quite accurately, that is, the farther away the value “vert constraints” is from 1, the more clear should be the vertical boundaries. Most importantly, do not forget that the term “stratified medium” or “layered medium” in geophysics, as a rule, implies a horizontally layered medium and vertical layers, as a rule, are present where there are some inclusions (anomalies) in the environment. You do not need to think that in a strictly horizontally layered environment, this option should be equal to 1, since “searching” for vertical boundaries (in the case of values less than 1) means only reducing the “diffusion” of the inversion results and more clearly highlighting possible anomalies. Often, on the contrary, it “helps” to better distinguish a purely horizontally layered medium, even though it lose the accuracy of resistance values (as blurring decreases).

The option “enh contrasts?” (similar to the option “enh contrasts?” for 2D) is a flag indicating the need to “enhance the contrast” between individual subdomains (environments): 1 — enhance the contrast, 0 — not enhance the contrast. What does it mean? The solution of the inversion problem is continuous, that is, you will always observe smooth transitions (in resistance sense) between different subdomains (environments). The realities of life are slightly different — a huge metal block can be bury in the ground, for example — that is sharp transition between subdomains (environments). The turn on of this option sometimes allows you to more clearly identify some subdomains (environments), the transition to which is carry out by abruptly values of resistance. However, do not think that this option is a panacea, since it does not cancel the continuity of the result of solving the inversion problem.

The option “robust data?” (similar to the option “robust data?” for 2D) is a flag indicating the necessity of evaluating data for the presence of noise: 1 — data should be evaluated for the presence of noise, 0 — data should not be evaluated for the presence of noise. What does this mean? Suppose your measured data for solving the problem of inversion were very noisy and you know about it. By turning on this option, you turn on the data estimation algorithm, which automatically drops or changes the weight of those values that are clearly out of the general trend. If you are sure that the measured data have a high degree of accuracy (for example, if they are obtain by numerically solving a direct problem), then turn on this option is NOT recommend.

The “lower res bound” and “upper res bound” options (similar to the “lower res bound” and “upper res bound” options for 2D) indicate the minimum and maximum resistance values in the solving inversion problem. That is, if you

precisely know the range of possible resistance values, then you can specify it; otherwise, it will be from zero to a very large value. However, you are not expect that as a result you will receive a solution in the exact range that was specify. It may happen that the inversion method does not allow to "fitting" the solution in your range, although, of course, the solution will be as close as possible to your range.

The “change of reg?” option (similar to the “change of reg?” option for 2D) is to change the regularization parameter mentioned above at each new iteration step using multiplication operation. For example, if the regularization parameter 100 is set in the option “regular strength”, then the option “change of reg?” as a value of 0.5 will reduce the regularization parameter on the second iteration twice (to 50), on the third iteration will reduce twice again (to 25) and so on. Why do you need it? Some complex models allow us to achieve acceptable inversion results only using of such manipulations.

The “max num of iter” option (similar to the “max num of iter” option for 2D) — you can limit the maximum number of iterations for solving the inversion problem using this option, but it is NOT recommended to do this, it is better to wait for the cycle to exit according to the convergence criteria, so the solution of the inversion problem will be more accurate. The meaning of using this option is only in one case — you want to see the solution at a certain iteration in order to observe how the solution changes with each iteration.

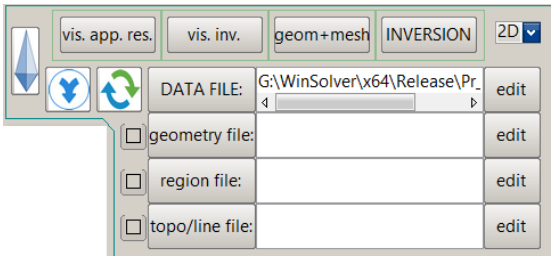
The “sensitivity values” option is for exclude measurements from the inversion algorithm whose sensitivities are below the specified value. Why do you need it? First of all to save computer’s memory. In the 3D case, where the memory consumption is very large, this is relevant. The fact is that measurements with weak sensitivity have little effect on the final result, but they participate in the inversion algorithm, taking up extra PC memory. By eliminating these values, you can often save memory by not losing much of the accuracy of the solution.

The “topography” option (similar to the “topography” option for 2D) indicates that it is necessary to take into account the “topographic effect”, namely, to take into account that the surface of the earth on which the measurements were made is NOT flat: 1 — take topography into account, 0 — do not take topography into account. For each array of electrical tomography, there is such a parameter as “geometric factor”. This parameter is calculate in relation of the distances between the current and potential electrodes. When the surface is not flat, it is undesirable to calculate this coefficient by the traditional method, because the electrodes are at different heights relative to each other. The greater the irregularity of the ground, the more the actual geometric factor may differ from the calculated one. By enabling this option, the geometric factors and the model mesh will be recalculate

based on the geometry features. On the other hand, the turn on of this option rather complicates the solution algorithm and the requirement for the free computer's random access memory. On the third hand, not enabling this option in the presence of a topographical effect can lead to the fact that the problem cannot be solve at all. On the fourth hand, if the geometric factor was initially calculated taking into account the effect of topography and, accordingly, taking into account the apparent resistances, then a new recalculation of these parameters will only garble the result. Therefore, you should always understand exactly what data you submit to the input.

As already mentioned above, the options associated with errors and with filtering existing and received data are identical for 2D and 3D cases.

Let us turn to the general element of this part of the main DiInSo window — to the buttons for selecting files, inversion, building geometry and meshes, visualizing and manipulating the input data:



Once again, pay attention to the button for selecting the dimension of the problem: "2D/3D". Once again, we recall that this button not only "tells" to the program which inversion algorithm will be use, but also replaces some parameter input fields with the necessary ones, and also changes the functionality of some buttons.

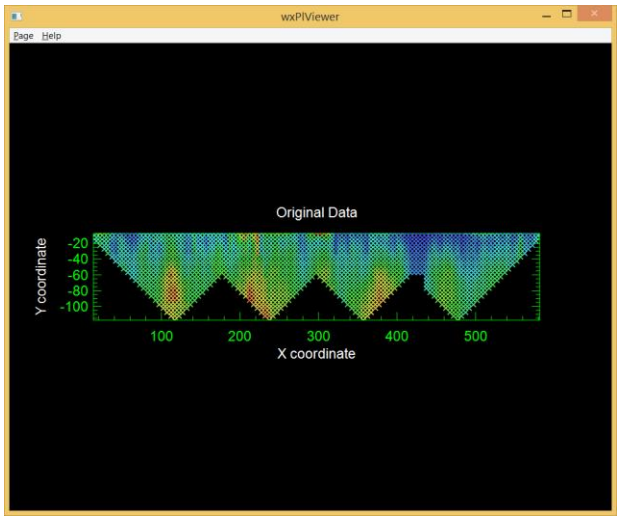
Choosing the correct dimension, go to the choice of the input data file for the inversion — the "DATA FILE" button. The input file must have the extension *.bert and have the following format:

<number of electrodes, or electrodes + points defining the topography>
"#x z" (without quotes) - in the 2D case, or "# x y z" (without quotes) - in the 3D case
<X-coordinate of the electrode (point)> <Y-coordinate of the electrode (point), which is only in the 3D case> <Z-coordinate of the electrode (point)>
<number of array positions>
"#a b m n rhoa" (without quotes), either "#a b m n rhoa ip" (without quotes), if there is data of induced polarization (IP), or "#a b m n rhoa err" (without quotes), if there is data of measurement errors, or "#a b m n rhoa ip err" (without quotes) if there is both induced polarization (IP) and measurement errors
<number of current electrode A> <number of current electrode B> <number of potential electrode M> <number of potential electrode N> <apparent resistance value> <value of induced polarization, if it represent> <value of measurement error, if it represent>

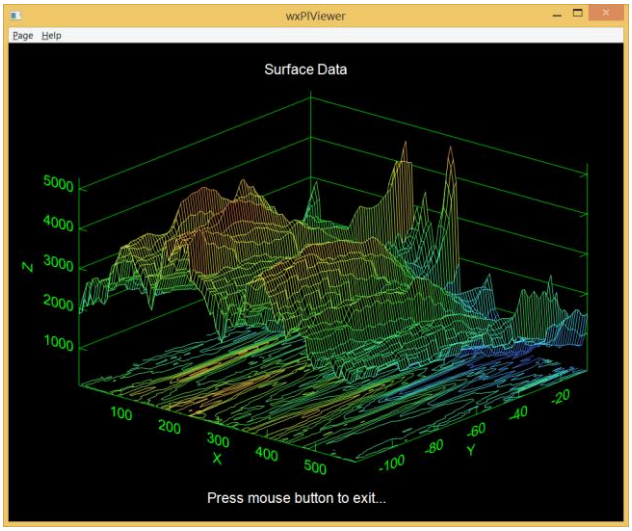
EXPLANATION: Where do electrode numbers come from? After the field "#x z" or "# x y z" are electrodes. It is assumed that the first electrode in this list is number 1, the second is number 2, and so on. What is the number of electrodes that are "at infinity", that is, in the case of the so-called "three-electrode array"? For

such electrodes, the number is ALWAYS zero — 0. Headers starting with # are not just comments; therefore, it is important to set them correctly.

After selecting the input file, you can see the distribution of apparent resistances in various forms of graphs by clicking the “vis. app. res.” button (now this function is available only in the 2D case):



You can switch between pages with different graph forms of data presentation either using the “Page” menu or using the “Enter” keyboard button — for next graph, “Page Up” keyboard button — for previous graph. To exit the window, go to the last page (for example, by pressing “Enter”), and then click the left mouse button in any part with graph of the last page (the last page has the inscription “Press mouse button to exit...” below):

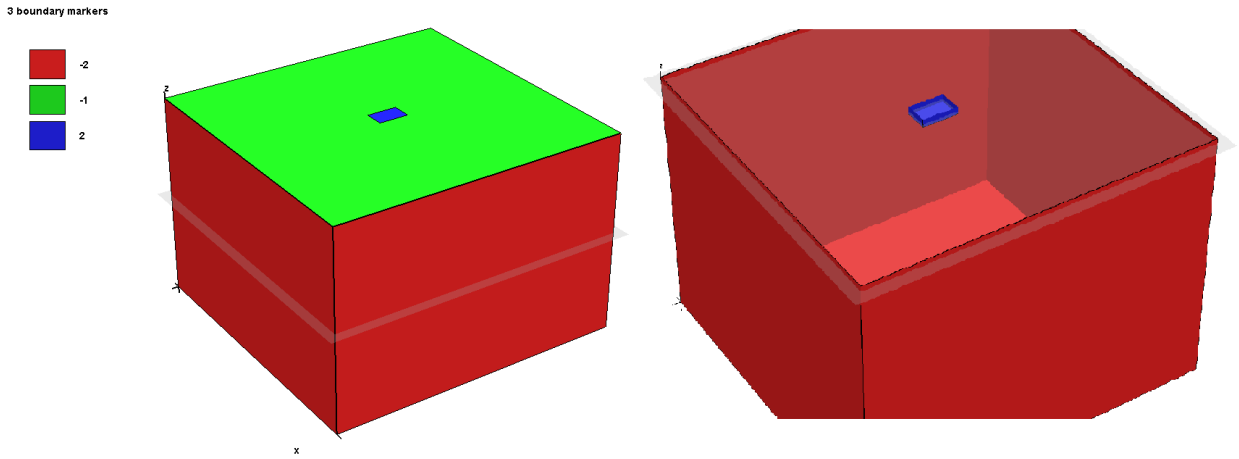


Pay attention that all viewed graphs will be saved in the same folder as the input *.bert file (!), in a file with the extension *.png and with the same name as the input file.

By clicking the “geom+mesh” button, you can get files with geometry, which, if necessary, will also contain the elements of the surface mesh, which using to

build the final mesh to solve the inversion problem. The files we need will have the extension *.poly, the file names will contain the name of the source file plus the postfix “GEOM” (the output files will be located in the same folder as the input file *.bert (!)). In addition, at the output you will find files with the *.vtk extension and the postfix “MESH” in the 2D case or the postfix “MESHPOLY” in the 3D case. These files allow you to view the resulting geometry/meshes using any VTK files visualizer (https://vtk.org/). Let’s back to the POLY files. For 2D and 3D cases, they are somewhat different.

In the 2D case, POLY files have a format designed for the Triangle triangulator, which is describe, for example, here: https://www.cs.cmu.edu/~quake/triangle.poly.html It is highly recommended to study this format. It is also necessary to study the features that are present in the created POLY file. !(Further, pay attention to the difference between the concepts of “nodes”, “boundaries” and “areas”)! For example, nodes that contain electrodes have a “–99” marker (we can remember this as a rule — “the smallest negative number” FOR NODES), and the remaining nodes have a “0” marker. Boundaries far away from the electrodes have a “–2” marker (we can remember this as a rule — “the smallest negative number” FOR BOUNDARIES). Internal (not upper) boundaries of the area where the solution is interesting for us have a marker “1”. Boundaries within this solution area can have any marker, starting with “–1” and more. All other boundaries (that is, in this case, the upper boundaries) have a “–1” marker. This is usually the boundary where the electrodes are located. The area where the solution is interesting for us has a marker “2”. The area where the solution is NOT interesting for us (the area associated with the boundaries far-far away from the electrodes) has a marker “1” (we can remember this as a rule — “the smallest positive number” FOR AREAS).



In the 3D case, the POLY files have a format designed for the Tetgen tetrahedral mesh generator, which is described, for example, here: http://wias-

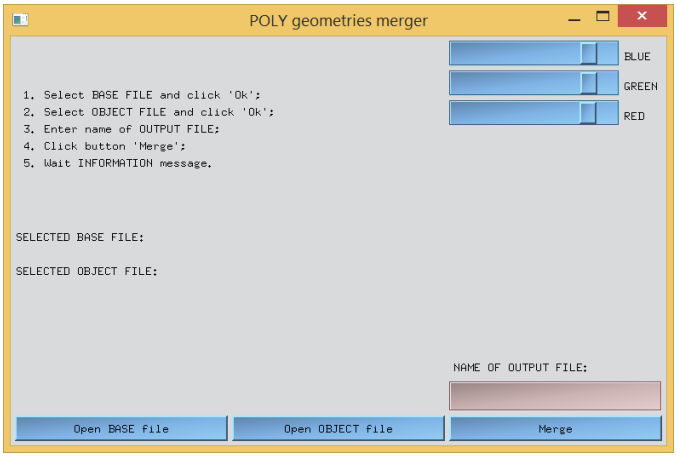
berlin.de/software/tetgen/fformats.poly.html To study this format is highly recommended. It is also necessary to study the features that are present in the created POLY file. !(Further, pay attention to the difference between the concepts of “nodes”, “boundaries” and “areas”)! For example, the boundary nodes far-far away from the electrodes have a “–2” marker. Boundary nodes that define the area where the solution is interesting for us have a marker “2”. Nodes that contain electrodes have a “–99” marker (we can remember this as a rule — “the smallest negative number” FOR NODES). The remaining nodes have a “0” marker (these are auxiliary nodes that lie under the electrode nodes at a certain depth). The upper boundary of the region WITHOUT the boundary where the solution is interesting for us (that is, WITHOUT the region where the electrodes are located) has a “–1” marker. The outer boundary, far-far away from the electrodes, WITHOUT the upper boundary, has a “–2” marker (we can remember this as a rule — “the smallest negative number” FOR BOUNDARIES). The boundary of the WHOLE area where the solution is interesting for us has a marker “2”. Internal objects must also have a boundary with a marker “2” or more. The area where the solution is interesting for us has a marker “2”. The area where the solution is NOT interesting for us (the area associated with the boundaries far-far away from the electrodes) has a marker “1” (we can remember this as a rule — “the smallest positive number” FOR AREAS).

It may seem that the rules for defining boundaries and areas in the 2D and 3D case are somewhat different, but this is a necessary measure associated with a different set of primitives. What is an object in the 2D case is the boundary in the 3D case. What is an object in the 3D case does not exist at all in the 2D case. In fact, the most important thing is to REMEMBER the rules for internal objects, and it is very simple — internal points always have a marker “0”, internal lines in the 2D case start with a value of “–1” and more, and internal polygons in the 3D case are “2” and more. Finally, just give to the internal object the marker following the last one.

Why do we need files with geometry? Only for one purpose — to carry out manipulations with them. At the entrance of the program, you can submit a newly created file with the geometry and get a solution on the created geometry, and not on the automatically generated one. The file with geometry is select using the “geometry file:” button. If the geometry was select, then it will be use to solve the inversion problem.

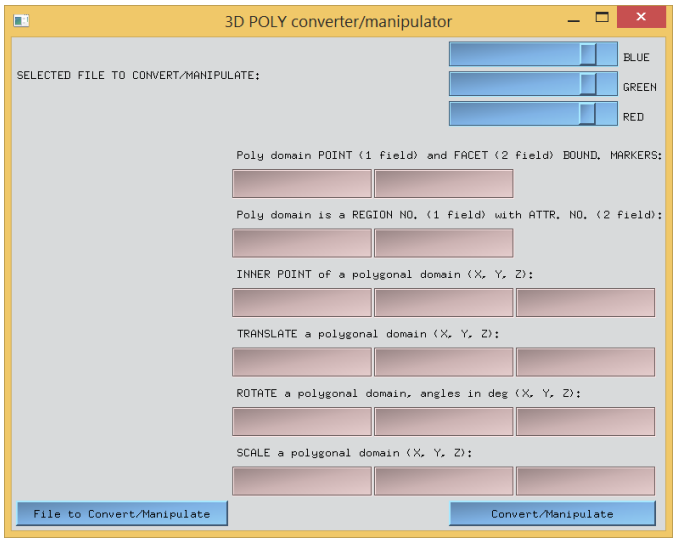
What else can you do with geometry? Combine TWO files with geometries using the button  . Why do you need it? You can add objects to geometry with known physical characteristics. In addition, you can add boundaries of objects without knowledge of their physical characteristics. It is important to note that it is

highly undesirable that the added object intersects the internal and external boundaries of the geometry to which it added, since in this case the final model may be incorrect. If you still need to cross the boundaries, it is recommended to manually check the correctness of the results obtained and, if necessary, to make corrections (in particular, this requires an understanding of the *.poly formats). The merging of two files window has an intuitive user-friendly interface:



Using the “Open BASE file” button, select the basic geometry (to which the object will be add), select the object with the “Open OBJECT file” button, which you will add to the basic geometry. In the field “NAME OF OUTPUT FILE:” enter the name of the new file with the geometry (with the file extension). Press the "Merge" button and wait for the message about the completion of the merge operation. In the upper right pane of the window, you can see “sliders” with the words “RED”, “GREEN” and “BLUE” — by moving them, you can change the background color of the window for better readability of the text on your monitor.

Let us assume that you need to add a three-dimensional object of quite an irregular shape to the geometry, which is not so easy to set manually even if you know the *.poly formats. In this case you need a button , which will help you convert an object created in a 3D editor into *.poly format. This window has the following form:



For the conversion, the following formats are support, arranged in alphabetical order:

3D, 3DS, 3MF, AC, AC3D, ACC, AMJ, ASE, ASK, B3D, BLEND, BVH, CMS, COB, DAE / Collada, DXF, ENFF, FBX, glTF 1.0 + GLB, glTF 2.0, HMB, IFC-STEP, IRR / IRRMESH, LWO, LWS, LXO, MD2, MD3, MD5, MDC, MDL, MESH / MESH.XML, MOT, MS3D, NDO, NFF, OBJ, OFF, OGEX, PLY, PMX, PRJ, Q3O, Q3S, RAW, SCN, SIB, SMD, STP, STL, TER, UC, VTA, X, X3D, XGL, ZGL.

Information on each of these formats you can easily find in the Internet, so there is no point in detailed study on each of them. Consider working with this window. So, choose the file to convert using the “File to Convert/Manipulate” button. In this case, the selected file may be a *.poly file, the conversion of which is not required. In the fields “Poly domain POINT (1 field) and FACET (2 field) BOUND. MARKERS:” you can enter markers of ALL nodes and faces that will be used in the output *.poly file. In the fields “Poly domain is a REGION NO. (1 field) with ATTR. NO. (2 field):” you can enter the number of the area and the number of the attribute of the area that used for your 3D model. In the fields “INNER POINT of a polygonal domain (X, Y, Z):” you can enter X, Y and Z — the coordinates of the internal point of your 3D model. In the fields “TRANSLATE a polygonal domain (X, Y, Z):” you can specify X, Y and Z — the coordinates of the offset vector by which your 3D model will be moved. In the fields “ROTATE a polygonal domain, angles in deg (X, Y, Z):” you can specify the CLOCKWISE rotation angles in degrees along each of the X, Y and Z axes, respectively, to which your 3D model will be rotated. In the fields “SCALE a polygonal domain (X, Y, Z)” you can specify the stretching/compression along each of the X, Y and Z axes, respectively, for your 3D model (for example, “2” without quotes means stretch twice along this axis, and “0.5” without quotes — compress twice along this axis). After clicking the “Convert/Manipulate” button,

you will receive a file with the extension *.poly, having the same name as the input file (if the file *.poly was originally chosen, then it will be subjected to manipulations). If you do not understand what is being discussed above, then you, most likely, have not studied the *.poly format. What happens if you do not specify an internal point? Then it will be consider that your 3D model is “empty”. This is true, for example, to define a 3D surface, which never has interior points. Is there support for 2D models for this window? No, but they are not needed here. 2D models are nothing complicated and can always be set manually using multiple lines. In the upper right pane of the window, you can see “sliders” with the words “RED”, “GREEN” and “BLUE” — by moving them, you can change the background color of the window for better readability of the text on your monitor.

Thus, you got a new geometry with some objects known in advance. Now you can use it to solve the inversion problem on your geometry. First, choose the geometry using the “geometry file:” button, then you can specify for the inversion algorithm the characteristics of those objects that you know in advance. This can be done using the button “region file:”, which selects a file with the characteristics of “regions”, having the extension *.control. This file has a view that is better to demonstrate by the example:

```
#no start Ctype MC  zWeight Trans lBound uBound
0   100   1     1   0.2     log   50     1000
1   30    0     0.2 1       log   10     200
#no single start
2   1      100
#no fix
3   22.5
#no background
-1  1
#inter-region
0   2  0.2
```

The number “#no” means the region number of the *.poly file, but it can also be “*” without quotes, which means all regions, or even “1*0”, which means all regions whose numbers start with 1 and end with 0. The “start” value sets the starting conductivity for a given region. The value of “Ctype” is similar to the option “enh contrasts?” described above. The value “MC” sets the value of the regularization parameter relative to the global value of the regularization. For example, 1 means that the regularization parameter for this region should be taken the same as for the rest of the problem, and the value 0.1, that the regularization parameter for this region should be taken 10 times smaller. The value of "zWeight" is similar to the option "vert constraints", which is written above. The value “trans” indicates to the inversion algorithm the so-called “model transformation” rule and

can have the values “log” (logarithmic), “lin” (linear) or “tan” (tangential) without quotes. In fact, this parameter affects primarily the process of convergence of the solving algorithm and is rarely chose to be different from the “log”, which is the default. The “lBound” and “uBound” values are similar to the “lower res bound” and “upper res bound” options described above. The value “single” is a flag indicating that this region must be consider as a region with constant resistance, the starting value of which is indicate in the “start” field. The “background” value is a flag that indicates that this region must be consider as a region where solution is not interesting for us, that is, this is the region that serves to far away the external boundaries of the solution of the inversion problem from the electrodes (described above). The “fix” value indicates a fixed resistance value for this region. However, when using this approach (“fix”), you need to understand that it implies the exclusion of a region from the inversion algorithm by the method of “filling” this region with a fixed value. In other words, this region simply “informs” the inversion algorithm of its value, but the calculations will take place without calculations within this region (the values are strictly fixed and do not change). Usually, the “fix” option use for reduce the complexity of the algorithm (there will not be an “extra” mesh in model, moreover the solution will not be “fitted” to the values of electrical resistance in certain regions). Finally, the “inter-region” values connect the two specified regions with each other in the same sense as it was with the “vert constraints” option described above. If the parameters for some regions in the *.control file are not specified, then these parameters are selected based on the global options.

Let's go to the lowest button, which is called “topo/line file”. This button has a different meaning for 2D and 3D inversions, and this is important to remember. In the 2D case, you must select a file with the extension *.xz, which has a simple format:

```
<X – point coordinate> <Z – point coordinate>
<X – point coordinate> <Z – point coordinate>
...
<X – point coordinate> <Z – point coordinate >
```

In this file, points are correspond the set that will be add to the 2D geometry of the model and then sequentially connected to a line. Why do you need it? With the help of such a manipulation, you can set the boundaries of regions that you know in advance within the model. Of course, you could do the same thing by modify the geometry file, as described above. However, sometimes the approach shown here is much easier. If you want to add several lines, you need to separate these lines in the file using blank lines. IT IS IMPORTANT TO REMEMBER that the points must be strictly inside the model geometry and not go beyond it.

In the 3D case, you can select a file with the extension *.xyz, which also has a simple format:

```
<X - point coordinate> <Y - point coordinate> <Z - point coordinate>
<X - point coordinate> <Y - point coordinate> <Z - point coordinate>
...
<X - point coordinate> <Y - point coordinate> <Z - point coordinate>
```

In this file, points are correspond the set that will be added to the 3D geometry of the model as points that change the topography of the surface on which the current and potential electrodes located. In other words, these points will ALWAYS be located on the "air–ground" boundary after the rebuilding of the geometric model. Of course, you could do the same thing by setting fake "electrodes" at topography points that are not used later as electrodes, but sometimes it's easier to do as described here. If you want to set the surface-boundaries inside the 3D model, then this can be realize only by working with the geometry file, as described above.

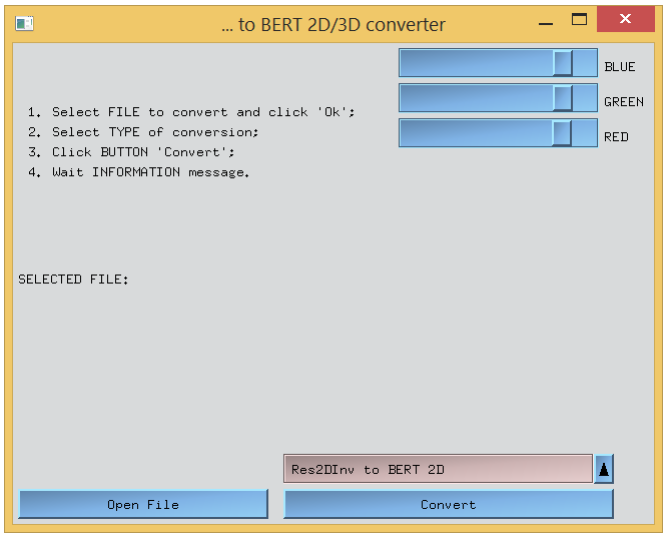
So, RESUME:

1) 2D case — these are files with the extension *.xz, which define the boundaries-lines of the INSIDE model geometry;

2) 3D case — these are files with the extension *.xyz, which define the surface topography by specifying points ON the "air–ground" boundary.

To the right of the buttons "DATA FILE:", "geometry file:", "region file:" and "topo/line file:", one can observe the "edit" buttons. Their purpose is quite simple — by clicking on such a button, a Notepad-like text file editor opens, which allows you to view and/or edit the corresponding file.

Another button with a picture, the functionality of which has not yet reviewed, is a button  that you can click on to see the window:



This is a converter window from formats used in third-party programs for solving inversion problems to the format of this program, that is, in *.bert files.

Currently, it is supported to convert Res2DInv/Res3DInv programs from the *.dat type “general array data” formats (https://www.geotomosoft.com), but this support is not full compatible due to the large variety of structures of the input data files of this two programs that, apparently, were added during the development of software, which contains a lot of years. Therefore, use this converter carefully — checking what comes out and knowing that some files CANNOT be convert at all. Using the “Open File” button, the file to be converted is select. The drop-down menu at the bottom right has three options: “Res2DInv to BERT 2D” — convert between 2D data, “Res3DInv to BERT 3D” — convert between 3D data and, finally, “List of Res2DInv's to BERT 3D” — convert several 2D data into 3D data. On the last case we need to stop in more details. Having several *.dat Res2DInv files, you have several two-dimensional profiles. It is obvious that several two-dimensional profiles can be combine into one three-dimensional. You can do this with the help of internal Res2DInv/Res3DInv converters, on which we will not dwell. Let’s show how to do such conversion in the program DiInSo. To do this, manually create a special file that goes to the converter input and has the following format:

```
<Res2DInv file name with FIRST profile>
<X displacement of NEXT profile> <Y displacement of NEXT profile relative to the FIRST profile (i.e., relative to zero)> <rotation angle (in degrees) of NEXT profile clockwise>
<Res2DInv file name with NEXT profile>
<X displacement of NEXT profile> <Y displacement of NEXT profile relative to the FIRST profile (i.e., relative to zero)> <rotation angle (in degrees) of NEXT profile clockwise>
<Res2DInv file name with NEXT profile>
...
```

That is, the lines with the file names are always one more than the lines with information about the displacement/rotation.

Therefore, the first profile is ALWAYS located along the Y-axis at the zero point and ALWAYS fixed. All other profiles are displace and rotate to the specified values. To summarize:

- the corresponding displacement value will be added to each X coordinate of all profiles except the base (first) one;

- all profiles, except the base (first), will be displace by Y relative to zero (0) by the corresponding value;

- after displacement, all profiles, except the base (first), will be rotated clockwise to the corresponding specified degree, where the rotation means rotation with a fixed first profile point;

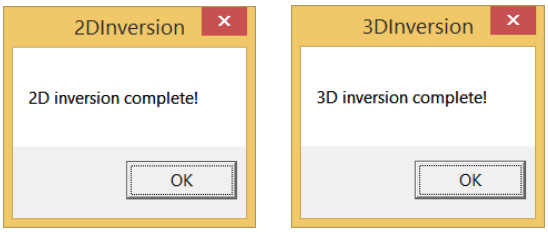
- if you do not need to displace the profile on some of the coordinates, then specify the corresponding value to zero (0), if you do not need to rotate the profile, then also specify the corresponding value to zero (0).

It is also IMPORTANT to NOTE that the listed files MUST be locate in the same directory as the file in which they are listed.

Finally, the “Convert” button is use to start the process of converting a file. The process of conversion ends with a message in the window area. At the output in the folder where the input file (!) is located, a file with the same name as the input file, but with the extension *.bert will be create.

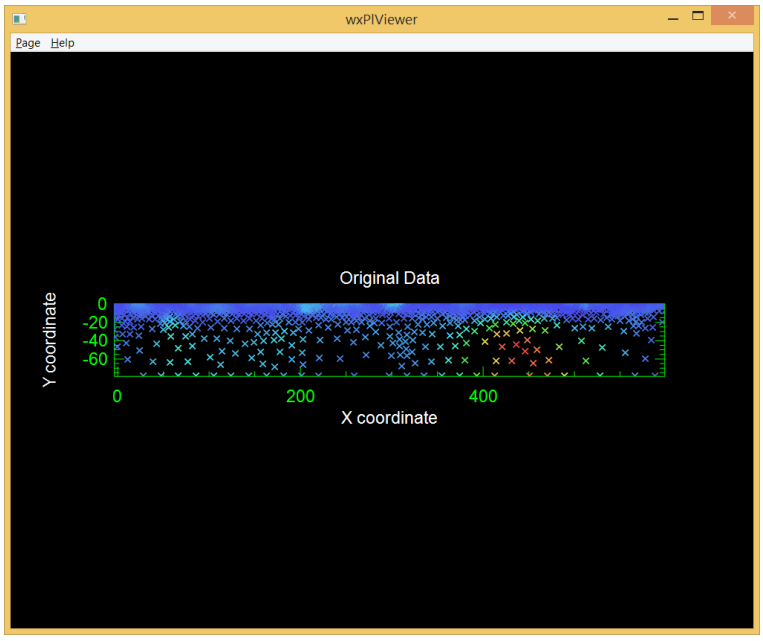
In the upper right part of the window, you can see “sliders” with the words “RED”, “GREEN” and “BLUE” — by moving them, you can change the background color of the window for better readability of the text on your monitor.

Now you can go to the main button of this part of the main DiInSo window — the “INVERSION” button. After the input file was selected, if necessary (recall that this is not necessary) files with geometry, region properties, topography or boundaries between regions are selected, all options needed by the user are set up (which is also optional, the options can be set automatically, however this does not guarantee a correct solution), and, MAIN, the problem is correctly indicated — 2D or 3D ... You can press the “INVERSION” button for activate the inversion process, which will end with an informational message like:



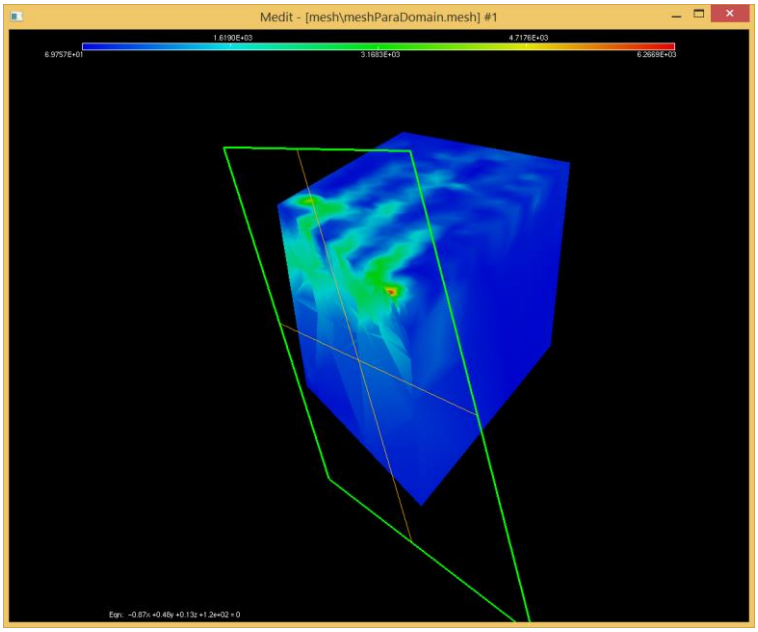
After performing the inversion, you can see the results (resistance values) in a graphical representation by clicking the “vis. inv.” button (the visualizers for 2D and for 3D cases will be different).

In the 2D case, the visualizer is similar to what you saw when you click the “vis. app. res.” button:



Rules for working with the visualizer are the same — switching between pages is perform using the “Page” menu, or using the “Enter” button — forward and “PageUp” button — backward. For exit the visualizer, as wrote before, you must reach the last page of the graphs and click on any graph area of the window with the left mouse button (the last page labeled by “Press mouse button to exit...” message below). At the same time note that among the graphs you will see both logarithmic results and regular ones. After viewing the graphs, they are also saved in picture files of the *.png format and have a name, the beginning of which coincides with the name of the input data file for inversion *.bert, and the end has the postfix "dcinv" in the case of regular graphs and "dcinvLOG" in the case of logarithmic graphs. In addition, after solving the 2D inversion problem, files will be obtain that can be load in other visualizers. First of all, these are files with *.vtk and *.vtu extensions (https://vtk.org/). PLEASE NOTE that the *.vtk and *.vtu files contain several information fields at once, that is, they contain both information about the mesh on which the inversion problem was solved, and the resulting resistances, logarithmic resistances, sensitivity, inversion of the induced polarization (if the data of the induced polarization were in the input file, otherwise such information in the output *.vtk and *.vtu files will not be). The output file names will match the name of the input data file for inversion, and some of them will have the postfix “P”. This postfix means that the data in these files is linked to nodes of mesh, while in files without the “P” postfix, the data is linked to cells of mesh. What is the difference? First of all, in the visual presentation. The data associated with the mesh nodes is better perceive by the eye. However, it is important to REMEMBER that the data associated with the mesh cells is more “correct” from the point of view of the inversion algorithm, since the results of the inversion obtain in the cells, and not in the nodes. In addition, files with *.surf extensions will be received. These are files for the “Surfer” commercial visualizer (https://www.goldensoftware.com/products/surfer). The names of these files have the postfix "P" — regular data with resistances associated with mesh nodes; “PLOG” — logarithmic resistance data associated with mesh nodes; “P_IP” — results of inversion of induced polarization associated with mesh nodes (if the data of the induced polarization were in the input file, otherwise there will be no files with the postfix “P_IP”).

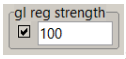
In the 3D case, the visualizer has the form:



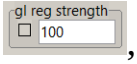
This visualizer based on the Medit visualizer (https://hal.inria.fr/inria-00069921/document) and has a similar interface. In this manual we will not dwell in detail on each function of this visualizer. We can only recommend — press the "h" button, being in the visualizer window, and look in the console DiInSo window — which buttons are responsible for what. The mouse also works in the visualizer: the left mouse button is responsible for rotation/movement, the right one brings up a drop-down menu, which, in particular, duplicates functions called by pressing keys on the keyboard. It is IMPORTANT TO REMEMBER that the keyboard language switch when working with the visualizer must be "English". In addition, after solving the 3D inversion problem, files will be obtain that can be loaded in other visualizers. First of all, these are files with *.vtk and *.vtu extensions (https://vtk.org/). PLEASE NOTE that the *.vtk and *.vtu files contain several information fields at once, that is, they contain both information about the mesh on which the inversion problem was solved, and the resulting resistances, logarithmic resistances, sensitivity, inversion of the induced polarization (if the data of the induced polarization were in the input file, otherwise such information in the output *.vtk and *.vtu files will not be). The output file names will match the name of the input data file for inversion, and some of them will have the postfix "P". This postfix means that the data in these files is linked to nodes of mesh, while in files without the "P" postfix, the data is linked to cells of mesh. What is the difference? First of all, in the visual presentation. The data associated with the mesh nodes is better perceive by the eye. However, it is important to REMEMBER that the data associated with the mesh cells is more "correct" from the point of view of the inversion algorithm, since the results of the inversion obtain in the cells, and not in the nodes. Also in the 3D case, you will see a special *.vtk file

with the “3D” postfix. This file is intend primarily for one freely distributed visualizer — VolView (https://www.kitware.com/volview/), which is characterized by simplicity and convenience in working with similar to our data. In addition, files with *.vox extensions will be received. These are files for the “Voxler” commercial visualizer (https://www.goldensoftware.com/products/voxler). The names of these files have the postfix "P" — regular data with resistances associated with mesh nodes; “PLOG” — logarithmic resistance data associated with mesh nodes; “P_IP” — results of inversion of induced polarization associated with mesh nodes (if the data of the induced polarization were in the input file, otherwise there will be no files with the postfix “P_IP”).

Thus, we have almost finished the examination of the part of the main DiInSo window that is responsible for solving the inversion problems. However, before moving on, VERY IMPORTANT NOTICE! Close to some of the options (including selection of files) can be seen the square, where you can put a check like this ☑. If this box is checked, then the parameters for this option from the input field will use. If it is not checked, then the parameters for this option will take from the default values. For example, for the 2D option “values of the regularization parameter” — “gl reg strength”:

- let the checkbox be active (checked) and enter the value “100” without quotes in the input field: 

, then the inversion algorithm will use the regularization parameter equal to 100;

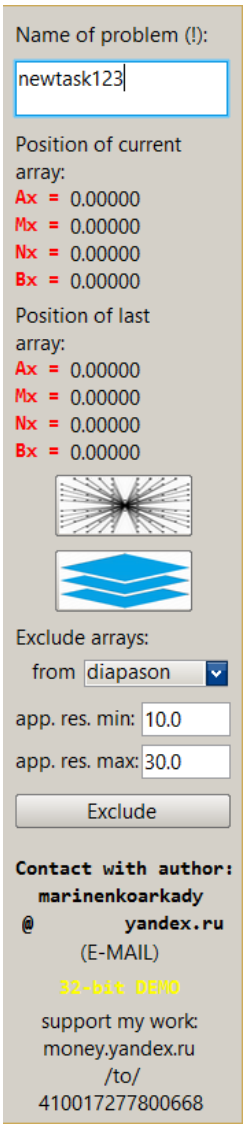
- let the checkbox is NOT active (NOT checked), and the input field still contains the value “100” without quotes: 

, then the inversion algorithm will use the “default” regularization parameter (i.e. 20).

This done solely for convenience, so that there is no need to erase the typed values each time — just uncheck the box!

Finally, go to the last part of the main DiInSo window, which is located on the right...

Supporting and informational functions of the program.



This part of the program is mainly auxiliary and informational, but it also has some very important functions.

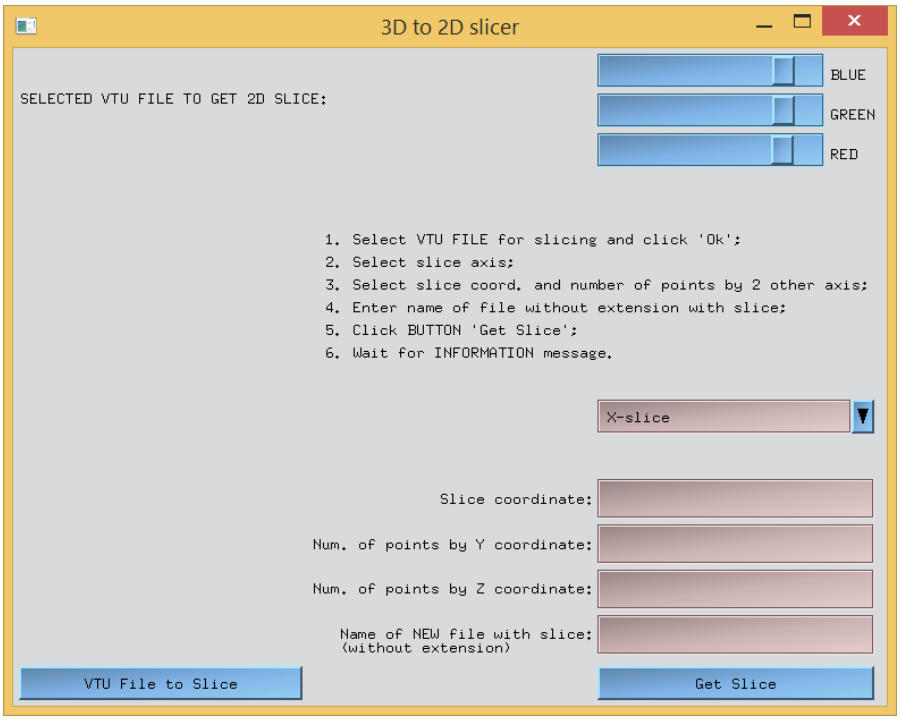
At the very top of this part of the program, you can see the field for entering the name of the problem, which is called “Name of problem (!):”. The exclamation point is not at all accidental. The name of the problem is very important for solving DIRECT PROBLEMS. As you have already read above, the process of solving direct problems is divided into many stages, where there is, in particular: the generation of a matrix of a system of linear algebraic equations; the establishment of correspondences between the numbers of nodes of the mesh and the electrodes of the array; assembling all received solutions into *.bert files (“bring together” button) and so on. At each of these stages, we get some files that are assign to the name of the problem, which specified in the “Name of problem (!):” field. When the resulting solutions collected in *.bert files (the “bring together” button), files with the *.sol extension not deleted, but moved to a folder with the same name as

the “Name of problem (!):” which will be create in the root directory of the DiInSo program. All this done in order to avoid confusion between the different solvable direct problems, and to save the solution algorithm from repetitive actions, if they already done before. Therefore, as soon as you begin to solve a new direct problem, enter a new name in the "Name of problem (!):" field. At the same time, the recommendations and requirements for the problem name are as follows: it is desirable not to use spaces in the problem name and DO NOT use service characters in the problem name: tilde (~), number sign (#), percent (%), ampersand (&), asterisk (*), curly brackets ({}), backslash (\), colon (:), angle brackets (<>), question mark (?), slash (/), plus sign (+), vertical bar (|), the quotation mark ("). When solving the inversion problem, the name of the problem plays no role, since the solution of the inversion problem does not contain as many user-dependent steps as the solution of the direct problem.

Below the problem name you can see non-editable fields named “Position of current array:”, “Position of last array:” and “Ax =”, “Mx =”, “Nx =”, “Bx =” after them. These fields are also associated with a direct problem. During the solution of many direct problems, the electrical tomography arrays moves from one position to another. This is the X–coordinate of the current (A, B) and potential (M, N) electrodes which will be displayed below the words “Position of current array:”. Below the words “Position of last array:” will be displayed the X–coordinates of the array electrodes for the direct problem that will be solved last in our task-list, that is, the electrical tomography array position, to which we are consistently “verge towards”, solving many direct problems.

The function of the button,  we have already considered above — this is an automatic generator of the coordinates of the set of array positions for all profiles, which will save in the file elec_profiles.txt.

The button  gives a window:



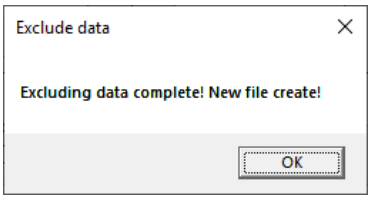
Using this window, you can get a two-dimensional section (slice) perpendicular to the X, Y or Z axis from your three-dimensional model saved in a VTU format file (VTK format is not supported!). The procedure of this is quite simple: 1. Select a VTU file using the “VTU File to Slice” button; 2. Choose which axis the section (slice) will be perpendicular using the drop-down menu; 3. Enter the coordinates of the section (slice) in the field “Slice coordinate:”; 4. Enter the number of data points at the corresponding coordinates in two fields below (the more points, the more detailed information about the section (slice) will be, however, do not forget that the accuracy of the data determines the number of points in the input file); 5. Enter the name of the output file without extension; 6. By clicking the “Get Slice” button, you will get output files with a section (slice) in VTK, VTU and Surfer formats.

Of course, you can do all this in the corresponding 3D visualizer, however, practice shows that people who have been working with 2D visualizers for a long time can hardly relearn for working with 3D visualizers (for example, from Surfer to Voxler). This feature has been implement for them.

Below the words "Exclude arrays:" is the program's feature, which created to help in adjusting the input file for the INVERSION PROBLEM. We have already considered the option when it becomes necessary to exclude some data from solving the inversion problem. We can do this using the options in the “DATA FILTERING:” section. However, sometimes you need to change the input file itself, permanently removing from it the data that we are not satisfied with for one

reason or another. After the word “from:” you can see a drop-down menu with three values: “diapason”, “file”, “graphic”. Let's go in order...

In the case of “diapason”, specify the range of values of apparent resistances in the fields “app. res. min:” and “app. res. max:”, click the “Exclude” button, wait for the information message:

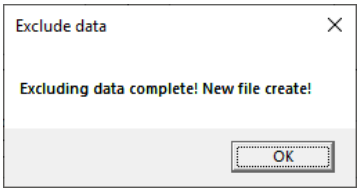


and in the same directory as the input file (specified in the “DATA FILE:” field), you will receive a new inversion problem *.bert file whose name matches the input file name plus postfix “newdiap”. In this new file, all the data included in the range of apparent resistances from the “app. res. min:” to “app. res. max:” will be excluded.

In the case of “file”, specify the name of the file with electrical tomography array positions to be exclude. The format of this file is:

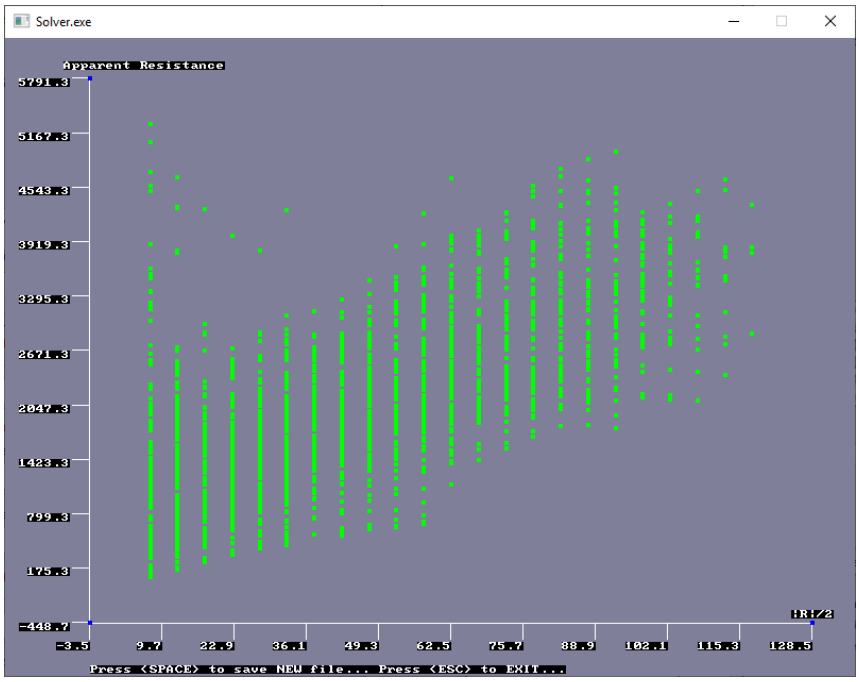
```
<number of excluded positions of electrical tomography array>
<number of first excluded position of array>
<number of second excluded position of array>
...
<number of last excluded position of array>
```

Press the “Exclude” button and wait for the information message:

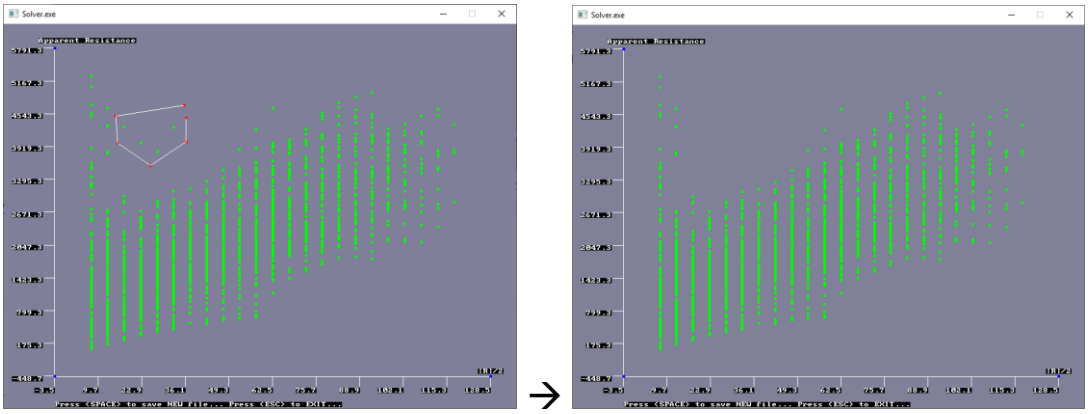


and in the same directory as the input file (specified in the “DATA FILE:” field), you will receive a new inversion problem *.bert file whose name matches the input file name plus the postfix “newarr”. In this new file, all electrical tomography array positions listed in the selected file specified in the “select file:” field will be exclude.

In the case of “graphic”, after clicking the “Exclude” button, the interactive deletion of “unnecessary” data will open:



On the X-axis, the half of the electrode spacing (R/2) shown in this window, and on the Y-axis, the values of apparent resistances obtained on the electrical tomography arrays. Using the left mouse button “surround” unnecessary values of the closed curve:

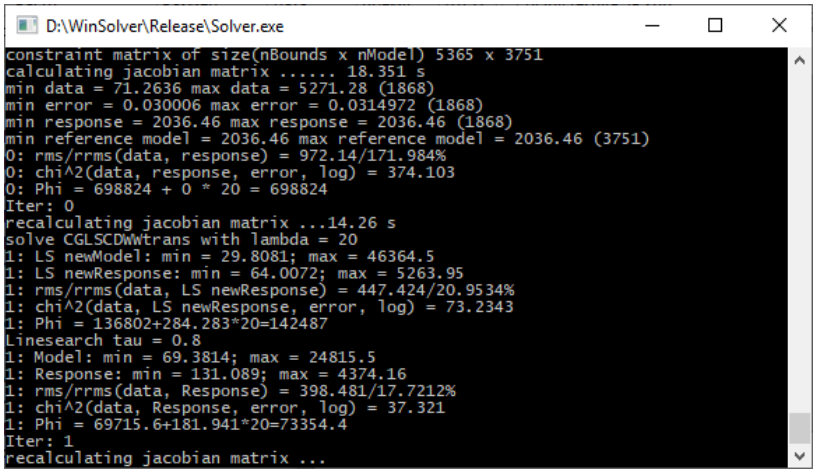


... and these values will be excluded. Repeat the process of “deleting points” as much as you need. Once finished, you can save a NEW file with excluded electrical tomography array positions by pressing the space bar on the keyboard. A new data file *.bert for inversion problem will be created in the same directory as the input file (specified in the “DATA FILE:” field), and its name will be the same as the input file name plus the postfix “newgraph”. To exit the “point exclusion” window, press ESC.

Finally, at the very bottom of this part of main DiInSo window, you will see information about chief programmer, his e-mail and how to support the project.

It remains to tell about two buttons that change the interface of the main DiInSo window:  . The “/\” button hides the upper (related to direct problems), and the “\/” button — the lower (related to inversion problems) areas of the main DiInSo window. Why do you need it? For example, you do not plan to solve inversion problems, and you are only interested in direct problems. In this case, you can press the "\/" button and the main window of DiInSo program does not occupy extra space on the monitor screen.

In addition to the main DiInSo window, there is also an auxiliary console window, which is always open together with the main window and which SHOULD NOT be close, because closing this window leads to the emergency closing of the entire program:



A similar “two windows” program structure can be find in many scientific packages, for example, in the same NetGen. The console window contains the so-called LOG of the program, that is, a lot of detailed information, both programmer and algorithmic type. Here you can, for example, monitor the convergence of the direct problem and the inversion problem, track some of the information which was read from the files, get information about the memory consumed, and much more. Usually, this window is interesting for user in two cases — check whether the solution “stagnate” or find the cause of the error. For example, it is possible that the mesh algorithm cannot correctly construct a mesh for solving the inversion problem and goes into an “infinite loop”. In this case, in the console window, you will see repetitive messages of the same type. To continue the work of the DiInSo program in this case does not make sense and it needs to be close in an emergency way. Another example would be the case when a very big problem is being solve and the program tries to occupy all the free space of the computer’s random access memory. The operating system will try to unload all processes, except the critical ones, into a so-called paging file (swap) in order to free up more computer’s

random access memory. All this causes unpleasant “hangs” in work and it may even seem that the computer has stopped responding to user actions. In this case, it is sometimes better to shut down the DiInSo program in an emergency way — by pressing “CTRL+C” in the console window (this method works in most console applications). In the console window there may be words unfamiliar to you (if you are reading this manual in English) in Russian. A good incentive to learn Russian, well, or just do not pay attention to them.

IMPORTANT questions that occur most often and answers on them.

1. Why you create this project?
   DiInSo was create solely for academic purposes, that is, to help the scientific community engaged in electro-tomographic studies.
2. How many people participated in the development of the DiInSo program?
   Marinenko Arkadiy Vadimovich — the chief programmer of DiInSo,
   Olenchenko Vladimir Vladimirovich, Epov Mikhail Ivanovich and the laboratory of geoelectrics of IPGG SBRAS — help in testing the DiInSo program and interesting ideas,
   OpenSource-community — people without whom this project could would be much worse and without which the progress of programming at all is almost impossible.
3. Who is the chief programmer of DiInSo?
   Mathematician, programmer and geophysicist.
4. I perfectly set up the program parameters/options and now I’m afraid of losing them!
   Do not be afraid, if you shut down the program correctly (by pressing the cross in the main window of the DiInSo program), all the parameters/options you entered will be saved and restored after the program will be restart (if you uncheck the parameter/option in checkbox, it will not be save).
5. I still do not understand what this parameter means when solving the inversion problem and I am very afraid to set it wrong!
   Just do not enter anything; the DiInSo program uses the “default” value, which is often not so bad.
6. It seems to me that the program has “stalled”/“hung” and it needs to be close. Is it possible? How to close DiInSo better?
   Yes, unfortunately, this is possible and there are many reasons for this — from an error in the program itself to special cases, such as the inability to build a mesh with the specified parameters for this particular model when solving the inversion problem. The best way to close DiInSo in such

case by pressing the "CTRL+C" button combination in the console window.

7. I definitely found a bug in the program! Is it possible? What should I do?
   Of course. This program is not devoid of errors, perhaps even critical. Write to the chief programmer's e-mail (marinenkoarkady@yandex.ru) information about the error, the sequence of actions that leads to the error and attach a log-file that is located in the folder with the DiInSo program and has name logging.log.
8. I have already read the manual many times, but I just can't understand IT! EITHER: I have a great idea for a DiInSo program! EITHER: You have an inaccuracy in the manual! Can I write to the author?
   Of course, do not hesitate to write to the chief programmer by e-mail (marinenkoarkady@yandex.ru) with a detailed description of your problem, my error in the manual (which are not excluded, because the program uses not only handwritten code, but also codes of other scientists and developers) or suggestions for improving the DiInSo program and do not take offence if you get an answer not immediately.
9. In what language can I write to the chief programmer of DiInSo?
   Russian or English.
10. How to understand that the program does not have enough computer's random access memory?
    Usually, a "not enough memory error" is no different from other program errors, that is, the program simply close by incorrect way. Often this closing is accompanied by the inscription "Not enough memory" in the console window of the DiInSo program or a separate window with the error "The memory could not be written":

11. I have A LOT OF PC RAM, but the DiInSo program still does not have enough memory to solve the problem. How can this be?
    You are probably using a 32-bit version of the DiInSo program, which has a native limit of 2 gigabytes of PC RAM.
12. What is the difference between the 64-bit version and the 32-bit version of DiInSo?

First of all, and most importantly, there is no limit of 2 gigabytes of PC RAM used, which is characteristic of all 32-bit applications. In addition, in the 64-bit version of the DiInSo program there is the possibility of choosing an algorithmic method for solving direct problems; direct problems can be solved on graphics cards (as a rule, much faster than solving problems on processors) using CUDA and/or OpenCL technologies; there are alternative data visualizers built into the DiInSo program; higher solving accuracy is provided due to the use of 64-bit data types.

13. How can I get a 64-bit version of the program?

    Contact the chief programmer Marinenko Arkadiy Vadimovich, or the head of the laboratory of geoelectrics of IPGG SBRAS Olenchenko Vladimir Vladimirovich (or better both at once) via e-mail (marinenkoarkady@yandex.ru and olenchenkovv@ipgg.sbras.ru).

14. I want to support the author of the project DiInSo? How can I do this?

    With the help of sponsorship donations to one of the following addresses:
    Yandex money: https://money.yandex.ru/to/410017277800668
    Qiwi: https://qiwi.com/n/ARKADIY1983
    PayPal: https://www.paypal.me/arkadiy1983
    Patreon: https://www.patreon.com/arkadiy1983
    Steady: https://steadyhq.com/en/arkadiy1983
    LP: https://liberapay.com/arkadiy1983

15. How much time does it take to upgrade a project to a state close to ideal?

    There is no clear framework, as with any similar projects, but if there is interest of this project, then its development will definitely continue.

16. I have learned the 32-bit version of the DiInSo program. Will I need to re-learn the 64-bit version?

    Of course not, the 64-bit version has very few differences from the 32-bit one. However, the 64-bit version allows solving large and complex problems that do not fit into 2 gigabytes of PC RAM.

17. I do not recognize Windows OS, serious scientific programs are created only on Linux! Is there a version of the DiInSo program for Linux?

    You are right, Linux is much better suited for scientific programs, but not every user is able to learn Linux. A version for Linux (Ubuntu) exists, but its development is currently suspend due to lack of time. At the same time, I hasten to assure you that if the interest in the program will be enough, this version will quickly see the light, since the DiInSo program was originally write without being strictly tie to the operating system and is essentially multiplatform.

18. Is it possible that the DiInSo project will be fully OpenSource?

Perhaps in the future, but now it does not make sense. Compiling of DiInSo at this stage is not an easy procedure, requires consideration of many details and does not yet have automatic "configurators", like CMAKE (https://cmake.org/).

19. I used earlier another software for solving inversion problems; it was much easy there...

The easier it is to use the software, the more the software does "automatically". The more automation in the software, the less opportunities to get an adequate scientifically verified result. In DiInSo an attempt was made to create a balance between the simplicity and the scientific approach to solving the problems.

20. Does the DiInSo program use parallel approaches?

Yes, and both in solving direct problems, and in solving inversion problems. The more processors/cores and PC RAM you have, the faster the problem will be solved and the more difficult the problems you will be able to solve (although the latter is more likely relevant to the 64-bit version).

21. Can I freely distribute the DiInSo program downloaded from sourceforge.net / github.com / gitlab.com / forum / Google.disk / Yandex.disk / friend's flash drives?

Yes, and it is even welcome. The more the DiInSo program spreads, the more interested there will be and the greater the desire to develop the project.

22. What is the best way to view VTK files?

The best programs for viewing VTK files, according to many people, are ParaView — https://www.paraview.org/, VisIt — https://wci.llnl.gov/simulation/computer-codes/visit and Cassandra — http://dev.artenum.com/projects/cassandra

23. Can I somehow see the geometry/mesh of 2D *.poly files of the Triangle program and the geometry/mesh of 3D *.ply files of the Tetgen program?

Yes, in the 2D case you can use the Triangle.NET program — https://archive.codeplex.com/?p=triangle or https://github.com/eppz/Triangle.NET or https://github.com/garykac/triangle.net, and in the 3D case the TetView program — http://wias-berlin.de/software/tetgen/tetview.html

The main OpenSource projects, without which the creation of DiInSo would be delaye for many years.

Interface of the main window of the DiInSo program: http://nanapro.org/en-us/

Interface of additional windows of the DiInSo program: http://plib.sourceforge.net/ and https://www.wxwidgets.org/

Drawing 2D graphs: http://plplot.sourceforge.net/

Drawing 3D graphs: https://www.ljll.math.upmc.fr/frey/software.html

Drawing 3D surfaces of equal data levels (64-bit version only): http://visualizationlibrary.org /docs/2.0/html/index.html

Drawing a portrait of the matrix: https://github.com/INMOST-DEV/INMOST

Converting 3D data formats: http://www.assimp.org/

Logging: https://sourceforge.net/projects/getset/ or https://github.com/aaichert/LibGetSet

Working with PDF files: https://mupdf.com/

Interactive work with graphs: https://liballeg.org/

OpenGL support: http://freeglut.sourceforge.net/

OpenCL support (64-bit version only): https://www.khronos.org/opencl/

CUDA support (64-bit version only): https://developer.nvidia.com/cuda-zone

Solving direct problems: http://www.paralution.com/

Solving inversion problems, working with geometries for inversion problems: https://www.pygimli.org/ and https://gitlab.com/resistivity-net/bert

Automatic generation of a triangular mesh when solving 2D inversion problems: https://www.cs.cmu.edu/~quake/triangle.html

Automatic generation of a tetrahedral mesh when solving 3D inversion problems: http://wias-berlin.de/software/index.jsp?id=TetGen

VTK data format support: https://vtk.org/

Tetrahedral mesh generators for solving direct problems of electro-tomography (not included in the project, but the results of the work of these programs are used as input data): http://gmsh.info/ and https://ngsolve.org/ and https://www.salome-platform.org/

As well as other projects that are associated with the above projects.

An example of solving a direct problem of electro-tomography with the subsequent solution of the inversion problem without complex objects using the NetGen mesh generator.

Consider the simplest direct problem of electro-tomography, where there are no complex objects, but there is only a two-layer environment, the boundary of

which is at a depth of 10 meters. Let's proceed to create a file with the geometry for the mesh generator NetGen. Such files begin with a keyword:

```
algebraic3d
```

Next, we set 6 planes that will restrict our modeling area. One of these planes will cross the points where the electrodes are set, and the remaining 5 planes are located far away from the electrodes:

```
# Lower plane
solid planev1 = plane (100.0, 100.0, -200.0; 0,0, -1) -bc = 76;
# Front plane
solid planev2 = plane (100.0, -50.0, 50.0; 0, -1,0) -bc = 76;
# Left plane
solid planev3 = plane (-50.0, 100.0, 50.0; -1,0,0) -bc = 76;
# Upper plane
solid planev4 = plane (100.0, 100.0, 0.0; 0,0,1);
# Rear plane
solid planev5 = plane (100.0, 200.0, 50.0; 0,1,0) -bc = 76;
# Right plane
solid planev6 = plane (250.0, 100.0, 50.0; 1,0,0) -bc = 76;
```

At the planes far away from the electrodes, we set the zero Dirichlet boundary conditions. This is done using the "–bc=76" parameter, that is, we will assume that the zero Dirichlet boundary conditions are specified using the value "76". We merge the planes into a single unit and get a region in the form of a "tank":

```
solid bak = planev1 and planev2 and planev3 and planev4 and planev5 and planev6;
```

Next, we divide our "tank" into several layers, two of which will be "natural" layers with different values of electrical conductivity, and two more — "fictitious" layers that will serve to adjust the parameters of mesh refinement. To do this, first set three separator planes:

```
solid cutplane1 = plane (100.0, 100.0, -1.0; 0,0,1);
solid cutplane2 = plane (100.0, 100.0, -4.0; 0,0,1);
solid cutplane3 = plane (100.0, 100.0, -10.0; 0,0,1);
```

And then four layers:

```
solid slice1 = bak and not cutplane1;
solid slice2 = bak and cutplane1 and not cutplane2;
solid slice3 = bak and cutplane2 and not cutplane3;
solid slice4 = bak and cutplane3;
```

Now we list these layers as objects-environments (actually, there are no other objects-environments in our model). We can do this using the keyword "tlo":

```
tlo slice1 -col = [0,1,0] -transparent -material = slice1 -maxh = 1.0;
tlo slice2 -col = [0,1,0] -transparent -material = slice2 -maxh = 2.0;
tlo slice3 -col = [0,1,0] -transparent -material = slice3 -maxh = 5.0;
tlo slice4 -col = [0,1,0] -transparent -material = slice4 -maxh = 20.0;
```

Keyword "-col" set the color of object-environment in the NetGen generator, "-transparent" indicates transparency of object-environment, and "-material" indicates the mesh generator that this object-environment belongs to the corresponding "material". Finally, "-maxh" sets the mesh parameters, namely the maximum size of the tetrahedron edge inside this object-environment. As mentioned above, the first three layers are essentially one layer from the point of view of physics, but we have divided them into different materials, because the habit of setting each object (even "fictitious") by separate parameters is a good habit.

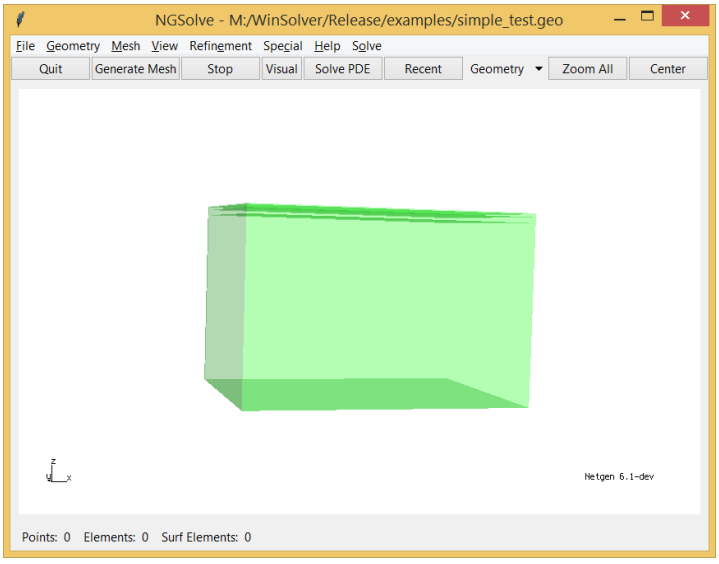
The last step is to list all points where the electrodes are:

```
point (50.0, 50.0, 0.0);
point (55.0, 50.0, 0.0);
point (60.0, 50.0, 0.0);
point (65.0, 50.0, 0.0);
point (70.0, 50.0, 0.0);
point (75.0, 50.0, 0.0);
point (80.0, 50.0, 0.0);
point (85.0, 50.0, 0.0);
point (90.0, 50.0, 0.0);
point (95.0, 50.0, 0.0);
point (100.0, 50.0, 0.0);
point (105.0, 50.0, 0.0);
point (110.0, 50.0, 0.0);
point (115.0, 50.0, 0.0);
point (120.0, 50.0, 0.0);
point (125.0, 50.0, 0.0);
point (130.0, 50.0, 0.0);
point (135.0, 50.0, 0.0);
point (140.0, 50.0, 0.0);
point (145.0, 50.0, 0.0);
point (150.0, 50.0, 0.0);
point (50.0, 55.0, 0.0);
point (55.0, 55.0, 0.0);
point (60.0, 55.0, 0.0);
point (65.0, 55.0, 0.0);
point (70.0, 55.0, 0.0);
point (75.0, 55.0, 0.0);
point (80.0, 55.0, 0.0);
point (85.0, 55.0, 0.0);
point (90.0, 55.0, 0.0);
point (95.0, 55.0, 0.0);
point (100.0, 55.0, 0.0);
point (105.0, 55.0, 0.0);
point (110.0, 55.0, 0.0);
point (115.0, 55.0, 0.0);
point (120.0, 55.0, 0.0);
point (125.0, 55.0, 0.0);
point (130.0, 55.0, 0.0);
point (135.0, 55.0, 0.0);
point (140.0, 55.0, 0.0);
point (145.0, 55.0, 0.0);
point (150.0, 55.0, 0.0);
point (50.0, 60.0, 0.0);
point (55.0, 60.0, 0.0);
point (60.0, 60.0, 0.0);
point (65.0, 60.0, 0.0);
point (70.0, 60.0, 0.0);
point (75.0, 60.0, 0.0);
point (80.0, 60.0, 0.0);
point (85.0, 60.0, 0.0);
point (90.0, 60.0, 0.0);
point (95.0, 60.0, 0.0);
point (100.0, 60.0, 0.0);
point (105.0, 60.0, 0.0);
point (110.0, 60.0, 0.0);
point (115.0, 60.0, 0.0);
point (120.0, 60.0, 0.0);
point (125.0, 60.0, 0.0);
point (130.0, 60.0, 0.0);
point (135.0, 60.0, 0.0);
point (140.0, 60.0, 0.0);
point (145.0, 60.0, 0.0);
point (150.0, 60.0, 0.0);
point (50.0, 65.0, 0.0);
point (55.0, 65.0, 0.0);
point (60.0, 65.0, 0.0);
point (65.0, 65.0, 0.0);
point (70.0, 65.0, 0.0);
point (75.0, 65.0, 0.0);
point (80.0, 65.0, 0.0);
point (85.0, 65.0, 0.0);
point (90.0, 65.0, 0.0);
point (95.0, 65.0, 0.0);
point (100.0, 65.0, 0.0);
point (105.0, 65.0, 0.0);
point (110.0, 65.0, 0.0);
point (115.0, 65.0, 0.0);
point (120.0, 65.0, 0.0);
point (125.0, 65.0, 0.0);
point (130.0, 65.0, 0.0);
point (135.0, 65.0, 0.0);
point (140.0, 65.0, 0.0);
point (145.0, 65.0, 0.0);
point (150.0, 65.0, 0.0);
point (50.0, 70.0, 0.0);
point (55.0, 70.0, 0.0);
point (60.0, 70.0, 0.0);
point (65.0, 70.0, 0.0);
point (70.0, 70.0, 0.0);
point (75.0, 70.0, 0.0);
```

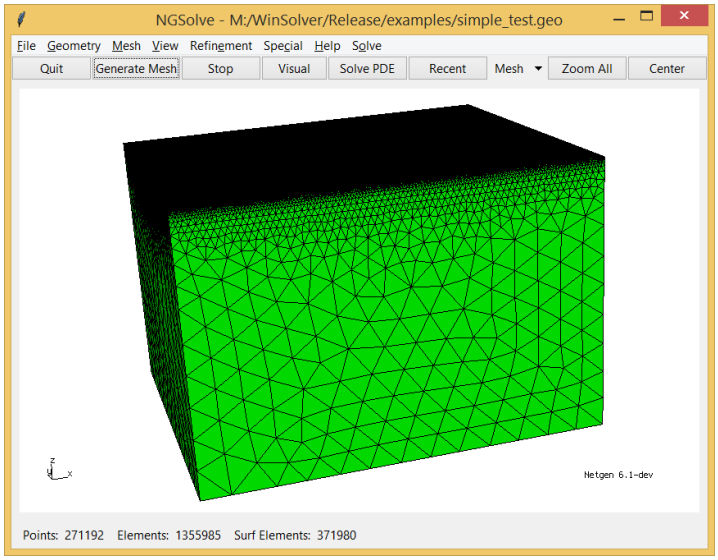
```
point (80.0, 70.0, 0.0);
point (85.0, 70.0, 0.0);
point (90.0, 70.0, 0.0);
point (95.0, 70.0, 0.0);
point (100.0, 70.0, 0.0);
point (105.0, 70.0, 0.0);
point (110.0, 70.0, 0.0);
point (115.0, 70.0, 0.0);
point (120.0, 70.0, 0.0);
point (125.0, 70.0, 0.0);
point (130.0, 70.0, 0.0);
point (135.0, 70.0, 0.0);
point (140.0, 70.0, 0.0);
point (145.0, 70.0, 0.0);
point (150.0, 70.0, 0.0);
point (50.0, 75.0, 0.0);
point (55.0, 75.0, 0.0);
point (60.0, 75.0, 0.0);
point (65.0, 75.0, 0.0);
point (70.0, 75.0, 0.0);
point (75.0, 75.0, 0.0);
point (80.0, 75.0, 0.0);
point (85.0, 75.0, 0.0);
point (90.0, 75.0, 0.0);
point (95.0, 75.0, 0.0);
point (100.0, 75.0, 0.0);
point (105.0, 75.0, 0.0);
point (110.0, 75.0, 0.0);
point (115.0, 75.0, 0.0);
point (120.0, 75.0, 0.0);
point (125.0, 75.0, 0.0);
point (130.0, 75.0, 0.0);
point (135.0, 75.0, 0.0);
point (140.0, 75.0, 0.0);
point (145.0, 75.0, 0.0);
point (150.0, 75.0, 0.0);
point (50.0, 80.0, 0.0);
point (55.0, 80.0, 0.0);
point (60.0, 80.0, 0.0);
point (65.0, 80.0, 0.0);
point (70.0, 80.0, 0.0);
point (75.0, 80.0, 0.0);
point (80.0, 80.0, 0.0);
point (85.0, 80.0, 0.0);
point (90.0, 80.0, 0.0);
point (95.0, 80.0, 0.0);
point (100.0, 80.0, 0.0);
point (105.0, 80.0, 0.0);
point (110.0, 80.0, 0.0);
point (115.0, 80.0, 0.0);
point (120.0, 80.0, 0.0);
point (125.0, 80.0, 0.0);
point (130.0, 80.0, 0.0);
point (135.0, 80.0, 0.0);
point (140.0, 80.0, 0.0);
point (145.0, 80.0, 0.0);
point (150.0, 80.0, 0.0);
point (50.0, 85.0, 0.0);
point (55.0, 85.0, 0.0);
point (60.0, 85.0, 0.0);
point (65.0, 85.0, 0.0);
point (70.0, 85.0, 0.0);
point (75.0, 85.0, 0.0);
point (80.0, 85.0, 0.0);
point (85.0, 85.0, 0.0);
point (90.0, 85.0, 0.0);
point (95.0, 85.0, 0.0);
point (100.0, 85.0, 0.0);
point (105.0, 85.0, 0.0);
point (110.0, 85.0, 0.0);
point (115.0, 85.0, 0.0);
point (120.0, 85.0, 0.0);
point (125.0, 85.0, 0.0);
point (130.0, 85.0, 0.0);
point (135.0, 85.0, 0.0);
point (140.0, 85.0, 0.0);
point (145.0, 85.0, 0.0);
point (150.0, 85.0, 0.0);
point (50.0, 90.0, 0.0);
point (55.0, 90.0, 0.0);
point (60.0, 90.0, 0.0);
point (65.0, 90.0, 0.0);
point (70.0, 90.0, 0.0);
point (75.0, 90.0, 0.0);
point (80.0, 90.0, 0.0);
point (85.0, 90.0, 0.0);
point (90.0, 90.0, 0.0);
point (95.0, 90.0, 0.0);
point (100.0, 90.0, 0.0);
point (105.0, 90.0, 0.0);
point (110.0, 90.0, 0.0);
point (115.0, 90.0, 0.0);
```

```
point (120.0, 90.0, 0.0);
point (125.0, 90.0, 0.0);
point (130.0, 90.0, 0.0);
point (135.0, 90.0, 0.0);
point (140.0, 90.0, 0.0);
point (145.0, 90.0, 0.0);
point (150.0, 90.0, 0.0);
point (50.0, 95.0, 0.0);
point (55.0, 95.0, 0.0);
point (60.0, 95.0, 0.0);
point (65.0, 95.0, 0.0);
point (70.0, 95.0, 0.0);
point (75.0, 95.0, 0.0);
point (80.0, 95.0, 0.0);
point (85.0, 95.0, 0.0);
point (90.0, 95.0, 0.0);
point (95.0, 95.0, 0.0);
point (100.0, 95.0, 0.0);
point (105.0, 95.0, 0.0);
point (110.0, 95.0, 0.0);
point (115.0, 95.0, 0.0);
point (120.0, 95.0, 0.0);
point (125.0, 95.0, 0.0);
point (130.0, 95.0, 0.0);
point (135.0, 95.0, 0.0);
point (140.0, 95.0, 0.0);
point (145.0, 95.0, 0.0);
point (150.0, 95.0, 0.0);
point (50.0, 100.0, 0.0);
point (55.0, 100.0, 0.0);
point (60.0, 100.0, 0.0);
point (65.0, 100.0, 0.0);
point (70.0, 100.0, 0.0);
point (75.0, 100.0, 0.0);
point (80.0, 100.0, 0.0);
point (85.0, 100.0, 0.0);
point (90.0, 100.0, 0.0);
point (95.0, 100.0, 0.0);
point (100.0, 100.0, 0.0);
point (105.0, 100.0, 0.0);
point (110.0, 100.0, 0.0);
point (115.0, 100.0, 0.0);
point (120.0, 100.0, 0.0);
point (125.0, 100.0, 0.0);
point (130.0, 100.0, 0.0);
point (135.0, 100.0, 0.0);
point (140.0, 100.0, 0.0);
point (145.0, 100.0, 0.0);
point (150.0, 100.0, 0.0);
```

The resulting file (available in the “examples” folder of the DiInSo program, called simple_test.geo) can be open in the NetGen mesh generator:



Remains only to create a mesh by clicking the “Generate Mesh” button (it will take several minutes to create the mesh):



A good fine detail mesh near electrodes — one of the keys to the success of solving a direct problem. However, do not forget that too detailed mesh requires a lot of PC memory, so overdoing with the fineness of the mesh is also not necessary. Save the mesh to the simple_test.vol file via the File -> Save Mesh drop-down menu...

Open the DiInSo program. In the field “Name of problem (!):” we think up a name of our task. Let it be, for example, “my_simple_test”. Click the “select file with mesh:” button and select the resulting mesh file simple_test.vol. Now you can think about the type of electrical tomography array that will be use for the solution.

Let it be pole-dipole array. Press the button  , in the drop-down menu select “Pole-Dipole”. Next, we recall how our electrodes are located and how many are them: X0 = 50, Y0 = 50, Z0 = 0, Number of electrodes in one profile — 21, Number of profiles — 11, X_step — 5, Y_step — 5. Click the “Generate" button and a new file elec_profiles.txt will be create. For the sake of interest, you can open it (via the dropdown menu “FILE OPEN”) and find out that the total number of problems that we have to solve is 4290. Make sure that the zero Dirichlet boundary conditions have the number 76. To do this, open the file null.txt (through the dropdown menu “FILE OPEN”). Now we check the current settings by opening the current.txt file (via the “FILE OPEN” dropdown menu). Its first three lines should be like this:

```
2
-1.0
1.0
```

We know that this electrical tomography array has one of the current electrodes at infinity, but we still need to set the currents on both electrodes, since in the so-called “three-electrode” arrays both current electrodes exists in fact and that means that a current must be set at both electrodes.

To understand how to fill the conductivity.txt file, you need to remember in which order we enumerated the objects-environments (“tlo”) in the input file for the NetGen mesh generator and understand that NetGen numbers the objects-environments starting from 1. The order was this:

1. top layer (fictitious) ,
2. top layer (fictitious),
3. top layer,
4. bottom layer.

Thus, the conductivity.txt file will contain 5 environments (4 environments we need plus a “zero” environment):

```
5
0.0
0.1
0.1
0.1
0.01
```

In the “zero” environment set an arbitrary value (since we don’t have it at all), let it be 0.0. Let the upper layer have an electrical conductivity value of 0.1 S/m and a lower layer of 0.01 S/m.

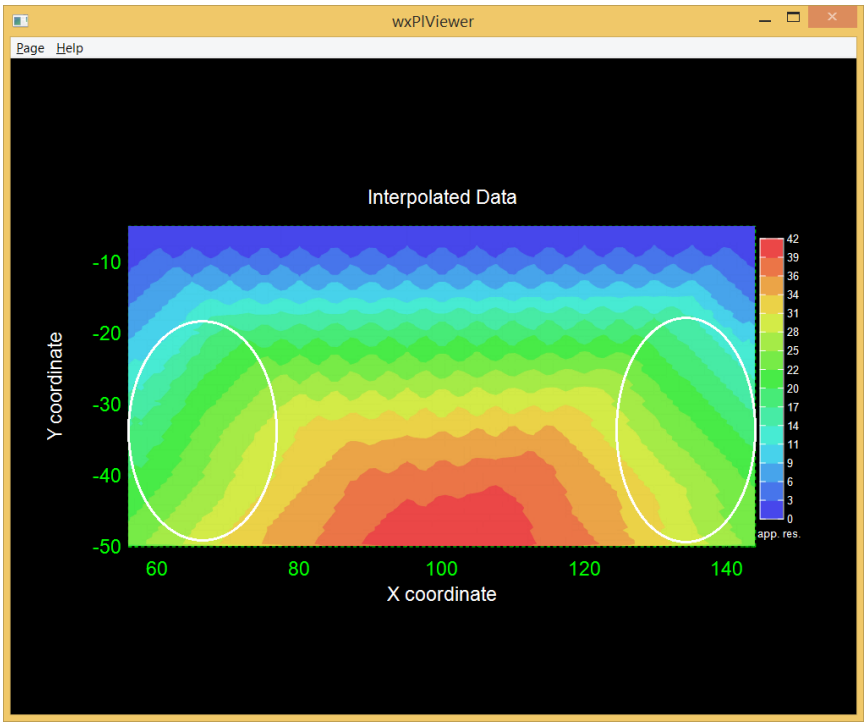
Now we are ready to generate a matrix of a system of linear algebraic equations. Click the “Start PREPROCESSOR (assembling matrix)” button for this.

The matrix is generated, you can start solving a direct problem, it is not necessary to look at the portrait of the matrix for such a simple problem. Suppose that we want to solve all problems at once on all profiles. Enter in the field “Y-coord. of profile:” the value of “x” without quotes, as the first problem — 1, and as the last problem — an arbitrary large number, for example, 100000. Click “START DIRECT solver” button. The process of solving direct problems begins.

After the end of solving direct problems (it will take several minutes), we press the “bring together” solution composition button, after which all the solution files will be in the “my_simple_test” directory (as we called our problem), which in turn is located in the DiInSo program directory. In the folder with the mesh for the direct problem, new files will appear — to solve the inversion problems in 2D and 3D cases. Let us turn to these problems...

Let's solve the two-dimensional problem of inversion, for example, on the sixth profile, for which we select the profile6.bert file by pressing the “DATA FILE:” button. Do not forget to choose the type of problem “2D”. Other settings we will set later.

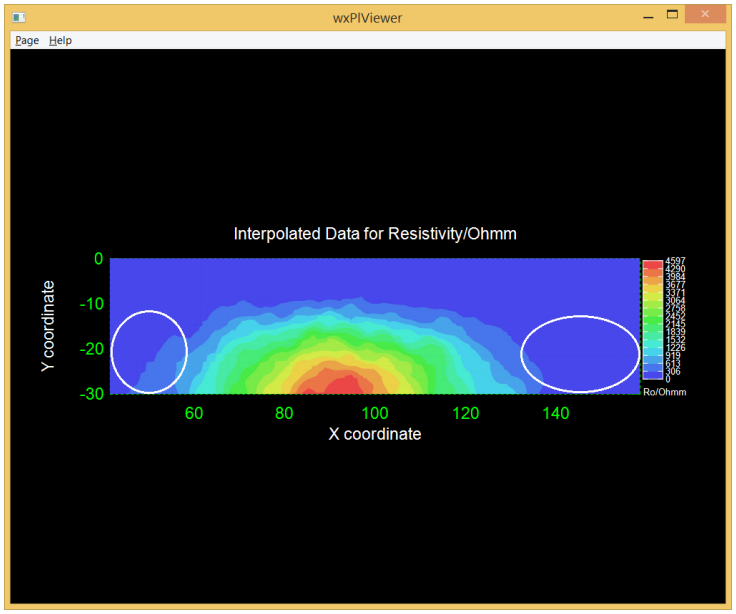
You can see the apparent resistances by clicking the button “vis. app. res.”:



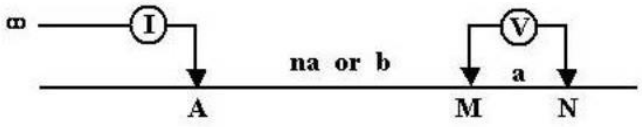
The curvatures highlighted by the ovals are associated with the effect of lack of data, that is, at these points are actually no measurements, and the data in the picture were obtained there by interpolation.

Let us try to solve the inversion problem in two variants — for a small value of the regularization parameter and for a large value of the regularization parameter.

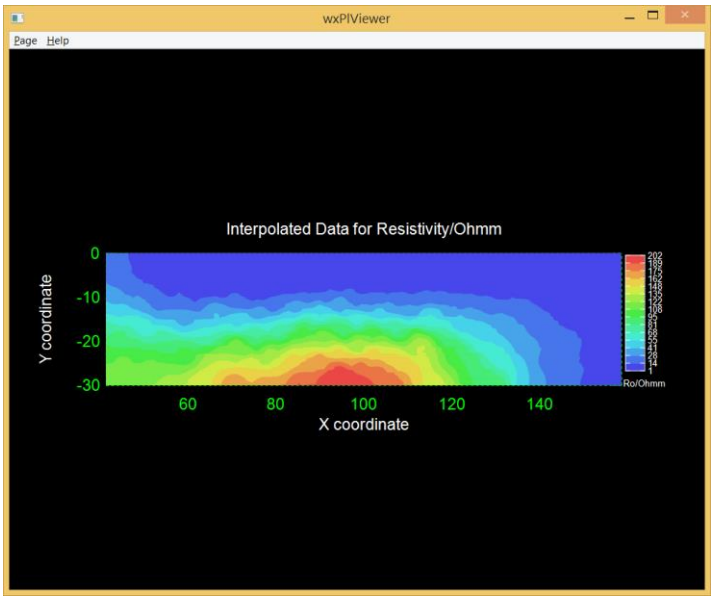
First, we set the following parameters: “size of cells” — 0.25, “mesh growing” — 34.0 (we need medium fine mesh), “model depth” — 30 (deeper than 30 meters, we are not interested in the results), “recalc jacobian?” — 1, “gl reg strength” — 1. Click the “INVERSION” button and see in the console window that the DiInSo program has completed 6 iterations, and the convergence is more or less normal. We can see the results of the inversion by clicking the button “vis. inv.”:



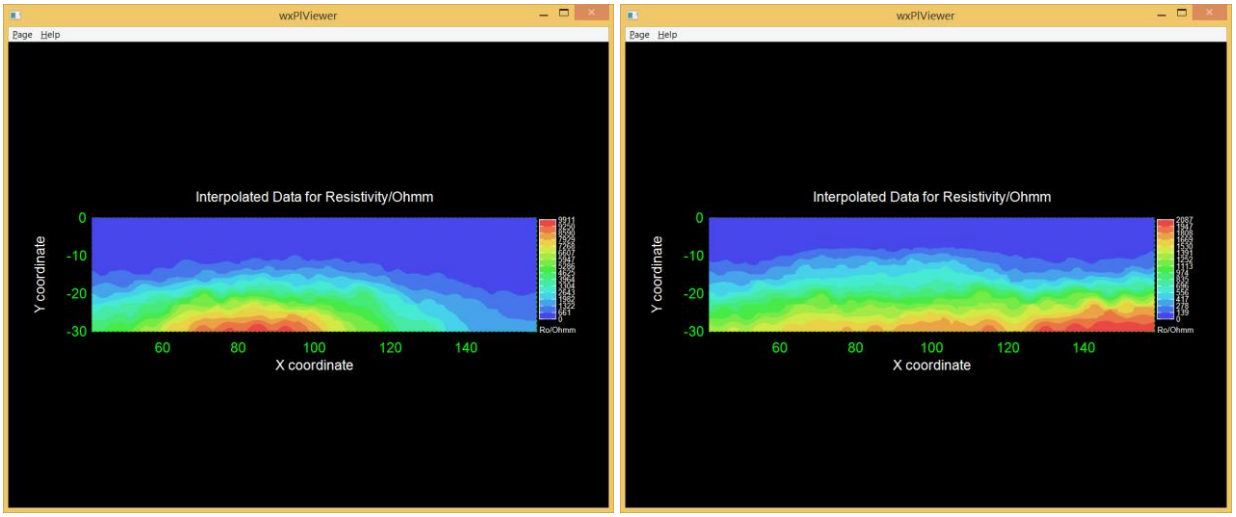
Upper layer, 10 meters thick, stands out quite clearly. The bottom layer stands out no so good, and there is not enough data on the sides of the bottom layer (highlighted by ovals), and in this case the lack of data on the right is felt more acutely. Why is that? To understand this, remember what the pole-dipole array looks like:



This electrical tomography array clearly "lacks" the current electrode on the right, hence the large area of incorrect data on the right. Also, note that with a small value of the regularization parameter, often rather coarse values of the resistances of the regions are obtain. To get more accurate resistance values, try increasing the regularization parameter. Without changing the other settings, we set "gl reg strength" equal to 50. Pressing the "INVERSION" button, we get the solution in 4 iterations with good convergence. We can see the results of the inversion by clicking the button "vis. inv.":



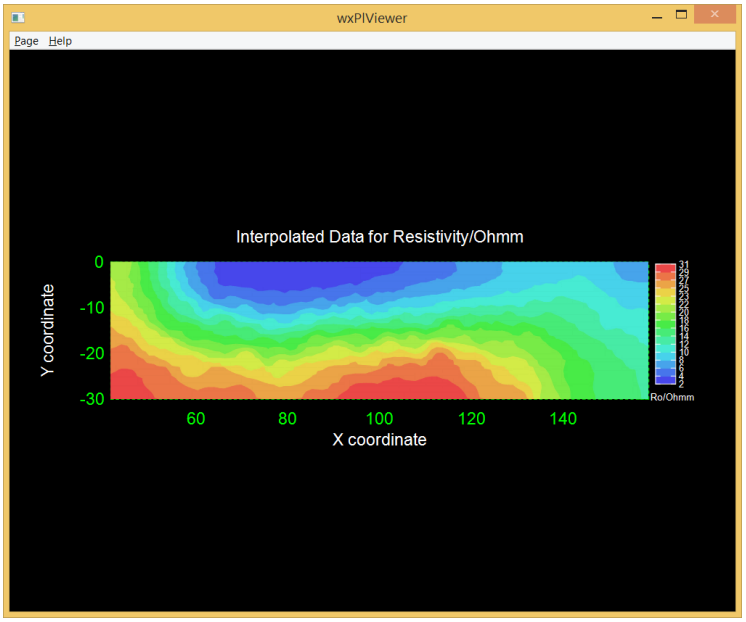
The result was more adequate in every sense of the word. The upper layer stands out quite clearly, there is a “transition layer” between the upper and lower layers, which is always difficult to get rid of. At the same time, the resistance values are much closer to the real ones. You can try to eliminate the distortion of the layered environment by proposing the inversion algorithm to search for vertical boundaries, thereby reducing the blurriness of the results. Recall that the “vert constraints” parameter is configure for this. Set two values for “vert constraints” — 0.1 (left) and 0.01 (right), that is, “blurred vertical boundaries” of the regions and “clear vertical boundaries” of the regions (which are not present):



As we can see, this approach really allows eliminate inaccuracies of geometry, which creates the pole-dipole array, but with loss of accuracy of resistance values. However, we note that sometimes it is better to “cut off” the part of the modeling area where the measured data are missing during demonstration the results of the inversion. If there were any objects inside the area where we have

not enough measuring points, they would not have been detect, but visually the "uncircumcised area" can create a perceived sense of the correctness of the inversion results in the entire area.

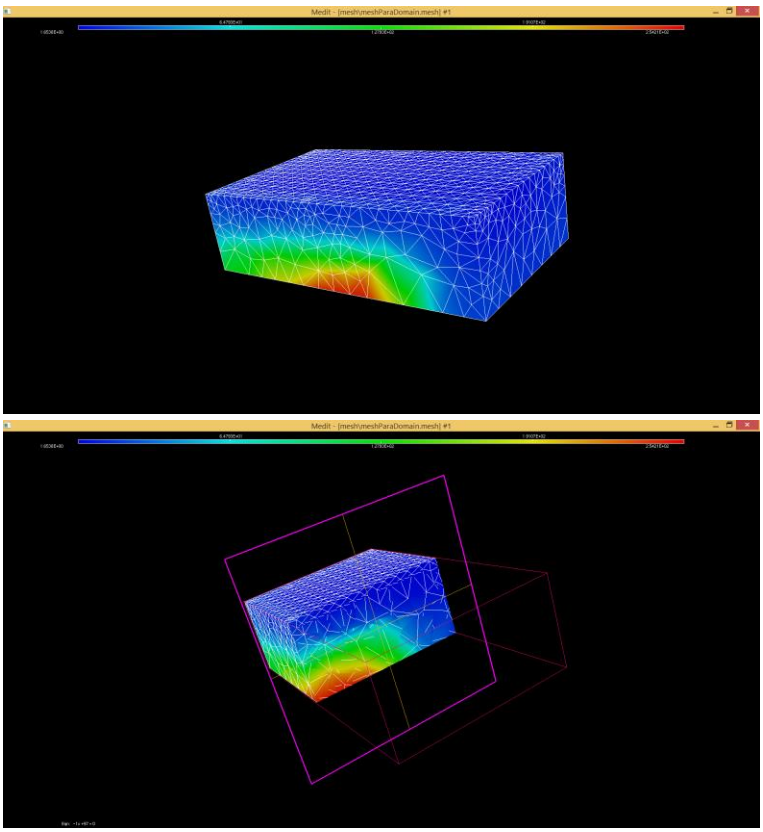
Finally, let's try a very large value of the regularization parameter. Without changing the other settings, we set "gl reg strength" equal to 500. Pressing the "INVERSION" button, we get the solution in 3 iterations with not very good convergence. We can see the results of the inversion by clicking the button "vis. inv.":



Such a result can hardly satisfy us. With the right degree of fantasy, we can understand that we are dealing with two layers, however, the resistance values are not accurate enough, and the layers themselves curved and create a false sense of NOT horizontal boundaries between the layers.

Thus, we have shown how important the correct selection of the value of the regularization parameter is.

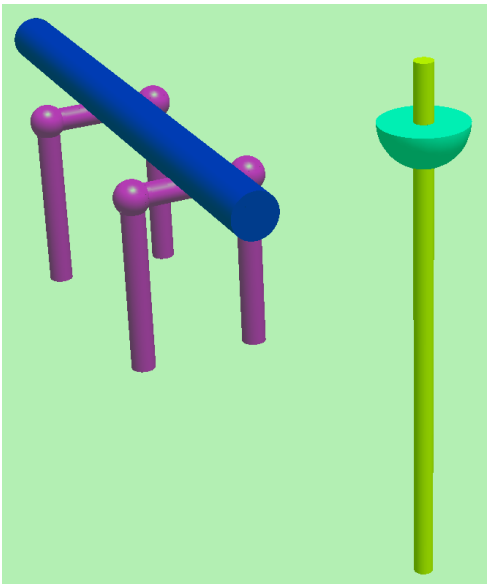
We will solve a similar three-dimensional problem, choosing as the regularization parameter the value that suits us best in the two-dimensional case — 50. We set the inversion type to "3D" and the parameters "quality" — 33, "mesh grow" — 1.2 (we need a relatively good mesh), "recalc jacobian?" — 1, "regular strength" — 50. We will also limit the depth of research to 30 meters, setting the "model depth" to 30. Select the created reverse_data.bert file by pressing the "DATA FILE:" button. We carry out inversion by clicking the "INVERSION" button. The solution converges in 4 iterations with relatively good convergence. We can see the results of the inversion by clicking the button "vis. inv.":



Here we see an interesting effect — the “lack” of the current electrode on the right of the pole-dipole array in the three-dimensional case “cuts off” the bottom layer not only on the right along the X-axis, but also on the right along the Y-axis. Frankly speaking, this is a fairly well known “defect” of pole-dipole array. To “overcome” it is often used “direct” and “reverse” course of the electrical tomography array, that is, first pass the pole-dipole array from left to right, in which the current electrode is on the left, and then pass the pole-dipole array from right to left, in which the current electrode is on the right. Thus, a more realistic result is obtain with the smaller area with not enough data points.

An example of solving a direct problem of electrical tomography with the subsequent solution of the problem of inversion with complex objects using a NetGen mesh generator.

Let’s simulate an electrical tomographic research in the area in which there are metal structures that interfere. Let these structures be two U-shaped metal abutments hammered into the ground, on which a metal pipe rests, as well as a buried well with a metal casing, in the near-surface zone of which there is a thawing zone (area with a higher temperature). As known, when the temperature of the environment changes, its physical properties, including the value of electrical conductivity, can also change; therefore, we will distinguish the thawing zones as a separate object. The construction described above has the form:



For simplicity, we will assume that all metal pipes are NOT hollow, that is, completely filled of metal. Since metal structures are both above ground and underground, we will need to specify an environment such as air, and we will bury the electrodes into the ground to some small depth (2 meters) in order to reduce the numerical noise effect from a strongly contrasting environment — air. We will also assume that below the "air–ground" boundary we have a two-layer zone, the upper layer of which has a thickness of 22 meters or 20 meters below the electrodes. Let the electrodes be arranged in 11 lines (that is, 11 profiles), each of the lines consists of 21 electrodes; the step between the lines (profiles) is 5 meters. As the electrical tomography array, we will take the Wenner-Alpha array, which does not have the drawbacks that the pole-dipole array has, described in the previous example (although this does not mean that this array is better, just each of the arrays is chosen for a specific problem).

First of all, it is necessary to define an area with all objects-environments and electrodes in order to build a tetrahedral mesh. We will do this with the help of the NetGen mesh generator, but you can choose another mesh generator — Gmsh, GiD or Salome. For building geometry file in NetGen we must use primitives for creating entire model. In the first line of geometry file we set special keyword:

```
algebraic3d
```

Next, we set 6 planes, which limit our modelling area (they should be far away from the electrodes — current sources):

```
# Lower plane
solid planev1 = plane (100.0, 100.0, -200.0; 0,0, - 1) -bc = 76;
# Front plane
solid planev2 = plane (100.0, -50.0, 50.0; 0, -1,0) -bc = 76;
# Left plane
solid planev3 = plane (-50.0, 100.0, 50.0; -1,0,0) -bc = 76;
# Upper plane
solid planev4 = plane (100.0, 100.0, 100.0; 0,0,1) -bc = 76;
# Rear plane
solid planev5 = plane (100.0, 200.0, 50.0; 0,1,0) -bc = 76;
```

```
# Right plane
solid planev6 = plane (250.0, 100.0, 50.0; 1,0,0) -bc = 76;
```

Since these are the external boundaries of the modelling area which all far away from the electrodes, it is necessary to specify the zero Dirichlet boundary conditions on them. This is done using the "–bc = 76" parameter, that is, we will assume that the zero Dirichlet boundary conditions are specified using the value "76". We merge the planes into a single "object" and get a region in the form of a "tank":

```
solid bak = planev1 and planev2 and planev3 and planev4 and planev5 and planev6;
```

Further, all objects will be created inside this "tank". We assign the parts of the lower U-shaped abutment with the help of three cylinders (as long as it is NOT the U-shaped abutment itself, these are just its parts):

```
solid truba1 = cylinder (67.5, 62.5, 6.0; 67.5, 62.5, -10.0; 1.0)
and plane ( 67.5, 62.5, 6.0; 0,0,1)
and plane (67.5, 62.5, -10.0; 0,0, -1);
solid truba2 = cylinder (77.5, 62.5, 6.0; 77.5, 62.5, -10.0; 1.0)
and plane (77.5, 62.5, 6.0; 0,0,1)
and plane (77.5, 62.5, -10.0; 0,0, - one);
solid truba3 = cylinder (67.5, 62.5, 5.0; 77.5, 62.5, 5.0; 1.0)
and plane (67.5, 62.5, 5.0; -1,0,0)
and plane (77.5, 62.5, 5.0; 1,0,0) ;
```

At two corners of the U-shaped abutment, we assign two spheres:

```
solid shar1 = sphere (67.5, 62.5, 5.0; 1.5);
solid shar2 = sphere (77.5, 62.5, 5.0; 1.5);
```

Spheres are need to fix the well-known error of the generator NetGen, which at the time of writing this manual is still not fixed. It consists in the fact that cylinders crossed at right angle disturb the work of the mesh generation algorithm and the process of creating the mesh goes into an infinite loop.

Repeat the procedure for the upper U-shaped abutment:

```
solid truba4 = cylinder (67.5, 82.5, 6.0; 67.5, 82.5, -10.0; 1.0)
and plane (67.5, 82.5, 6.0; 0,0,1)
and plane (67.5, 82.5 -10.0; 0.0, -1);
solid truba5 = cylinder (77.5, 82.5, 6.0; 77.5, 82.5, -10.0; 1.0)
and plane (77.5, 82.5, 6.0; 0,0,1)
and plane (77.5, 82.5, -10.0; 0,0, - one);
solid truba6 = cylinder (67.5, 82.5, 5.0; 77.5, 82.5, 5.0; 1.0)
and plane (67.5, 82.5, 5.0; -1,0,0)
and plane (77.5, 82.5, 5.0; 1,0,0) ;
solid shar3 = sphere (67.5, 82.5, 5.0; 1.5);
solid shar4 = sphere (77.5, 82.5, 5.0; 1.5);
```

With the help of the cylinder, we set the metal pipe lying on the abutments:

```
solid bigtruba = cylinder (72.5, 50.0, 7.0; 72.5, 100.0, 7.0; 2.0)
and plane (72.5, 50.0, 7.0; 0, -1.0)
and plane (72.5 , 100.0, 7.0; 0.1.0);
```

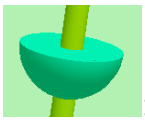
With the help of a hemisphere, we set the thawing zone:

```
solid polushar1 = sphere (102.5, 77.5, 0.0; 4.5) and plane (102.5, 77.5, 0.0; 0.0,1);
```

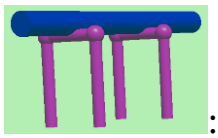
Using the cylinder, we set the bore well:

```
solid trubafull1 = cylinder (102.5, 77.5, -50.0; 102.5, 77.5, 6.0; 1.0)
and plane (102.5, 77.5, 6.0; 0,0,1)
and plane (102.5, 77.5, -50.0 ; 0,0, -1);
```

Next, you need to inform the algorithm about which objects intersect and which ones merge, otherwise we will get several meshes within the same areas.

Let's start with the simplest — the thawing zone does NOT include the bore well  :

```
solid zatepl1 = polushar1 and not trubafull1;
```

Each of the U-shaped abutments combines three support pipes, two spheres at the corners of the U-shaped abutment, and excludes the pipe that lies on top of the abutments  :

```
solid oporaniz = (truba1 or truba2 or truba3 or shar1 or shar2) and not bigtruba;
solid oporaverh = (truba4 or truba5 or truba6 or shar3 or shar4) and not bigtruba;
```

The two U-shaped abutments and the pipe lying on them is a single metal object from the point of view of our model, so we combine them (this object will not be used in the future, so this step can be skipped, but to understand the structure of the object it is left):

```
solid vsetruby = oporaniz or oporaverh or bigtruba;
```

As we remember, our "tank" has several layers — air, the top layer of the earth and the bottom layer of the earth. However, we will define another layer that is inside the top layer of the earth. Electrodes buried by 2 meters will lie on the boundary of this layer. Thus, we have total four layers, the boundaries between which are divide by three planes:

```
solid cutplane1 = plane (100.0, 100.0, 2.0; 0,0,1);
solid cutplane2 = plane (100.0, 100.0, 0.0; 0,0,1);
solid cutplane3 = plane (100.0, 100.0, -20.0; 0,0,1);
```

Now list each of the four layers, remembering that the objects inside the layer must be exclude from the layer:

```
# Air
solid slice1 = bak and not cutplane1 and not trubafull1;
# Layer of earth created for electrodes
and not zbpl1 and not trubafull1;
# Top layer of the earth
solid slice3 = bak and cutplane2, and not cutplane 3 and not trubafull1;
# Bottom layer of the earth
solid slice4 = bak and cutplane3 and not trubafull1;
```

Now we will list all the objects-environments ("tlo"), highlighting the materials of the objects-environments ("-material") separately, including color ("-col") with transparency ("-transparent") and limiting the maximum sizes of the edges of the tetrahedra inside the objects-environments ("-maxh"):

```
tlo slice1 -col = [0,1,0] -transparent -material = slice1 -maxh = 20.0;
tlo slice2 -col = [0,1,0] -transparent -material = slice2 -maxh = 1.5;
tlo slice3 -col = [0,1,0] -transparent -material = slice3 -maxh = 10.0;
tlo slice4 -col = [0,1,0] -transparent -material = slice4 -maxh = 20.0;
tlo oporaniz -col = [1,0,1] -material = oporaniz -maxh = 1.0;
tlo oporaverh -col = [1,0,1] -material = oporaverh -maxh = 1.0;
tlo bigtruba -col = [0,0,1] -material = bigtruba -maxh = 2.0;
tlo zatepl1 -col = [0,1,1] -material = polushar1 -maxh = 3.0;
tlo trubafull1 -col = [1,1,0] -material = trubafull1 -maxh = 1.0;
```

Note that we again "divided" our U-shaped abutments and the pipe lying on them into separate objects. This done in order to set different mesh generation rules

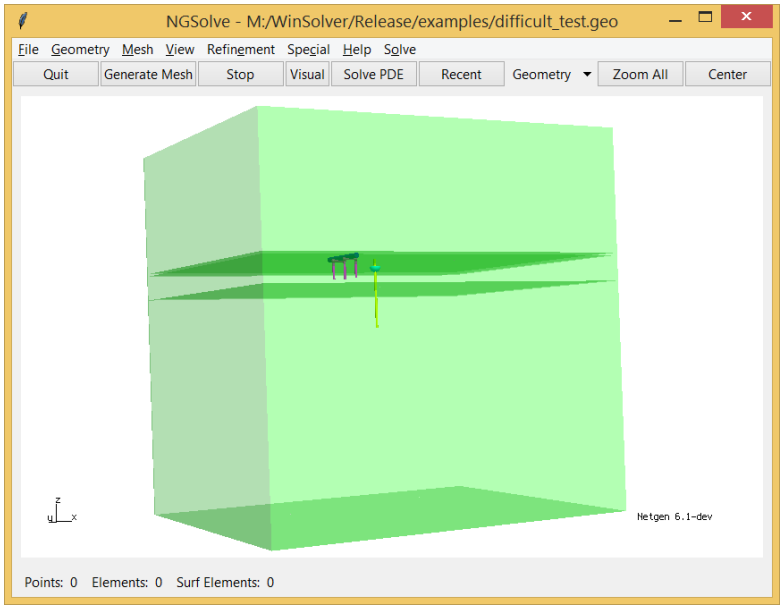
for sub-objects of a single object (different maximum sizes of the tetrahedron edges). The last step is to list all points where the electrodes are located:

```
point (50.0, 50.0, 0.0);
point (55.0, 50.0, 0.0);
point (60.0, 50.0, 0.0);
point (65.0, 50.0, 0.0);
point (70.0, 50.0, 0.0);
point (75.0, 50.0, 0.0);
point (80.0, 50.0, 0.0);
point (85.0, 50.0, 0.0);
point (90.0, 50.0, 0.0);
point (95.0, 50.0, 0.0);
point (100.0, 50.0, 0.0);
point (105.0, 50.0, 0.0);
point (110.0, 50.0, 0.0);
point (115.0, 50.0, 0.0);
point (120.0, 50.0, 0.0);
point (125.0, 50.0, 0.0);
point (130.0, 50.0, 0.0);
point (135.0, 50.0, 0.0);
point (140.0, 50.0, 0.0);
point (145.0, 50.0, 0.0);
point (150.0, 50.0, 0.0);
point (50.0, 55.0, 0.0);
point (55.0, 55.0, 0.0);
point (60.0, 55.0, 0.0);
point (65.0, 55.0, 0.0);
point (70.0, 55.0, 0.0);
point (75.0, 55.0, 0.0);
point (80.0, 55.0, 0.0);
point (85.0, 55.0, 0.0);
point (90.0, 55.0, 0.0);
point (95.0, 55.0, 0.0);
point (100.0, 55.0, 0.0);
point (105.0, 55.0, 0.0);
point (110.0, 55.0, 0.0);
point (115.0, 55.0, 0.0);
point (120.0, 55.0, 0.0);
point (125.0, 55.0, 0.0);
point (130.0, 55.0, 0.0);
point (135.0, 55.0, 0.0);
point (140.0, 55.0, 0.0);
point (145.0, 55.0, 0.0);
point (150.0, 55.0, 0.0);
point (50.0, 60.0, 0.0);
point (55.0, 60.0, 0.0);
point (60.0, 60.0, 0.0);
point (65.0, 60.0, 0.0);
point (70.0, 60.0, 0.0);
point (75.0, 60.0, 0.0);
point (80.0, 60.0, 0.0);
point (85.0, 60.0, 0.0);
point (90.0, 60.0, 0.0);
point (95.0, 60.0, 0.0);
point (100.0, 60.0, 0.0);
point (105.0, 60.0, 0.0);
point (110.0, 60.0, 0.0);
point (115.0, 60.0, 0.0);
point (120.0, 60.0, 0.0);
point (125.0, 60.0, 0.0);
point (130.0, 60.0, 0.0);
point (135.0, 60.0, 0.0);
point (140.0, 60.0, 0.0);
point (145.0, 60.0, 0.0);
point (150.0, 60.0, 0.0);
point (50.0, 65.0, 0.0);
point (55.0, 65.0, 0.0);
point (60.0, 65.0, 0.0);
point (65.0, 65.0, 0.0);
point (70.0, 65.0, 0.0);
point (75.0, 65.0, 0.0);
point (80.0, 65.0, 0.0);
point (85.0, 65.0, 0.0);
point (90.0, 65.0, 0.0);
point (95.0, 65.0, 0.0);
point (100.0, 65.0, 0.0);
point (105.0, 65.0, 0.0);
point (110.0, 65.0, 0.0);
point (115.0, 65.0, 0.0);
point (120.0, 65.0, 0.0);
point (125.0, 65.0, 0.0);
point (130.0, 65.0, 0.0);
point (135.0, 65.0, 0.0);
point (140.0, 65.0, 0.0);
point (145.0, 65.0, 0.0);
point (150.0, 65.0, 0.0);
point (50.0, 70.0, 0.0);
point (55.0, 70.0, 0.0);
point (60.0, 70.0, 0.0);
```

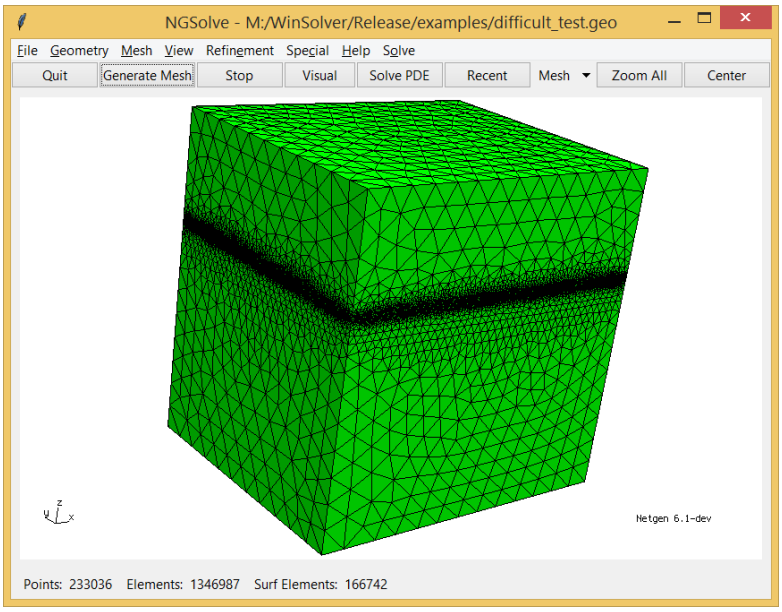
```
point (65.0, 70.0, 0.0);
point (70.0, 70.0, 0.0);
point (75.0, 70.0, 0.0);
point (80.0, 70.0, 0.0);
point (85.0, 70.0, 0.0);
point (90.0, 70.0, 0.0);
point (95.0, 70.0, 0.0);
point (100.0, 70.0, 0.0);
point (105.0, 70.0, 0.0);
point (110.0, 70.0, 0.0);
point (115.0, 70.0, 0.0);
point (120.0, 70.0, 0.0);
point (125.0, 70.0, 0.0);
point (130.0, 70.0, 0.0);
point (135.0, 70.0, 0.0);
point (140.0, 70.0, 0.0);
point (145.0, 70.0, 0.0);
point (150.0, 70.0, 0.0);
point (50.0, 75.0, 0.0);
point (55.0, 75.0, 0.0);
point (60.0, 75.0, 0.0);
point (65.0, 75.0, 0.0);
point (70.0, 75.0, 0.0);
point (75.0, 75.0, 0.0);
point (80.0, 75.0, 0.0);
point (85.0, 75.0, 0.0);
point (90.0, 75.0, 0.0);
point (95.0, 75.0, 0.0);
point (100.0, 75.0, 0.0);
point (105.0, 75.0, 0.0);
point (110.0, 75.0, 0.0);
point (115.0, 75.0, 0.0);
point (120.0, 75.0, 0.0);
point (125.0, 75.0, 0.0);
point (130.0, 75.0, 0.0);
point (135.0, 75.0, 0.0);
point (140.0, 75.0, 0.0);
point (145.0, 75.0, 0.0);
point (150.0, 75.0, 0.0);
point (50.0, 80.0, 0.0);
point (55.0, 80.0, 0.0);
point (60.0, 80.0, 0.0);
point (65.0, 80.0, 0.0);
point (70.0, 80.0, 0.0);
point (75.0, 80.0, 0.0);
point (80.0, 80.0, 0.0);
point (85.0, 80.0, 0.0);
point (90.0, 80.0, 0.0);
point (95.0, 80.0, 0.0);
point (100.0, 80.0, 0.0);
point (105.0, 80.0, 0.0);
point (110.0, 80.0, 0.0);
point (115.0, 80.0, 0.0);
point (120.0, 80.0, 0.0);
point (125.0, 80.0, 0.0);
point (130.0, 80.0, 0.0);
point (135.0, 80.0, 0.0);
point (140.0, 80.0, 0.0);
point (145.0, 80.0, 0.0);
point (150.0, 80.0, 0.0);
point (50.0, 85.0, 0.0);
point (55.0, 85.0, 0.0);
point (60.0, 85.0, 0.0);
point (65.0, 85.0, 0.0);
point (70.0, 85.0, 0.0);
point (75.0, 85.0, 0.0);
point (80.0, 85.0, 0.0);
point (85.0, 85.0, 0.0);
point (90.0, 85.0, 0.0);
point (95.0, 85.0, 0.0);
point (100.0, 85.0, 0.0);
point (105.0, 85.0, 0.0);
point (110.0, 85.0, 0.0);
point (115.0, 85.0, 0.0);
point (120.0, 85.0, 0.0);
point (125.0, 85.0, 0.0);
point (130.0, 85.0, 0.0);
point (135.0, 85.0, 0.0);
point (140.0, 85.0, 0.0);
point (145.0, 85.0, 0.0);
point (150.0, 85.0, 0.0);
point (50.0, 90.0, 0.0);
point (55.0, 90.0, 0.0);
point (60.0, 90.0, 0.0);
point (65.0, 90.0, 0.0);
point (70.0, 90.0, 0.0);
point (75.0, 90.0, 0.0);
point (80.0, 90.0, 0.0);
point (85.0, 90.0, 0.0);
point (90.0, 90.0, 0.0);
point (95.0, 90.0, 0.0);
point (100.0, 90.0, 0.0);
```

```
point (105.0, 90.0, 0.0);
point (110.0, 90.0, 0.0);
point (115.0, 90.0, 0.0);
point (120.0, 90.0, 0.0);
point (125.0, 90.0, 0.0);
point (130.0, 90.0, 0.0);
point (135.0, 90.0, 0.0);
point (140.0, 90.0, 0.0);
point (145.0, 90.0, 0.0);
point (150.0, 90.0, 0.0);
point (50.0, 95.0, 0.0);
point (55.0, 95.0, 0.0);
point (60.0, 95.0, 0.0);
point (65.0, 95.0, 0.0);
point (70.0, 95.0, 0.0);
point (75.0, 95.0, 0.0);
point (80.0, 95.0, 0.0);
point (85.0, 95.0, 0.0);
point (90.0, 95.0, 0.0);
point (95.0, 95.0, 0.0);
point (100.0, 95.0, 0.0);
point (105.0, 95.0, 0.0);
point (110.0, 95.0, 0.0);
point (115.0, 95.0, 0.0);
point (120.0, 95.0, 0.0);
point (125.0, 95.0, 0.0);
point (130.0, 95.0, 0.0);
point (135.0, 95.0, 0.0);
point (140.0, 95.0, 0.0);
point (145.0, 95.0, 0.0);
point (150.0, 95.0, 0.0);
point (50.0, 100.0, 0.0);
point (55.0, 100.0, 0.0);
point (60.0, 100.0, 0.0);
point (65.0, 100.0, 0.0);
point (70.0, 100.0, 0.0);
point (75.0, 100.0, 0.0);
point (80.0, 100.0, 0.0);
point (85.0, 100.0, 0.0);
point (90.0, 100.0, 0.0);
point (95.0, 100.0, 0.0);
point (100.0, 100.0, 0.0);
point (105.0, 100.0, 0.0);
point (110.0, 100.0, 0.0);
point (115.0, 100.0, 0.0);
point (120.0, 100.0, 0.0);
point (125.0, 100.0, 0.0);
point (130.0, 100.0, 0.0);
point (135.0, 100.0, 0.0);
point (140.0, 100.0, 0.0);
point (145.0, 100.0, 0.0);
point (150.0, 100.0, 0.0);
```

The resulting file (available in the “examples” folder of the DiInSo program, named difficult_test.geo) can be open in the NetGen mesh generator:



Remains only to create a mesh by clicking the “Generate Mesh” button (it will take several minutes to create the mesh):



Save the mesh to a file difficult_test.vol via File -> Save Mesh pop-up menu...

Open the DiInSo program. In the field “Name of problem (!):” we enter the name of our problem. Let it be, for example, “my_difficult_test” without quotes. Click the “select file with mesh:” button and select the resulting difficult_test.vol file with mesh. Let’s configure additional input files through the dropdown menu “FILE OPEN”. The first three lines of the current.txt file should look like this:

```
2
-1.0
1.0
```

To understand how to set the conductivity.txt file, you need to remember in which order we enumerated the objects-environments (“tlo”) in the input file for the NetGen mesh generator and understand that NetGen numbers the objects-environments starting from 1. The order was as follows:

1. Air,
2. Layer of earth created for electrodes,
3. Upper layer of earth,
4. Lower layer of earth,
5. Lower U-shaped abutment,
6. Upper U-shaped abutment,
7. Pipe lying on U-shaped abutments,
8. A thawing zone,
9. Bore well.

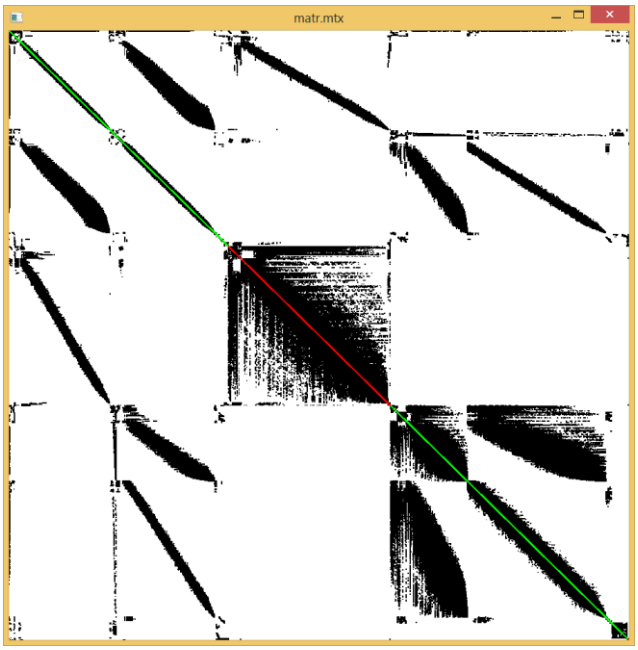
Therefore, conductivity.txt file will contain 10 objects-environments (9 that we need plus “zero” object-environment):

```
10
0.0
```

```
1.0E-12
0.001
0.001
0.1
7690000.0
7690000.0
7690000.0
0.01
7690000.0
```

For “zero” object-environment we enter arbitrary value (because it does not exist), let it be 0.0. Air has a very low electrical conductivity value. However, we can’t simply wright down 0.0 as conductivity (see Maxwell’s equation system), so we will set a very small number: 1.0E-12 S/m. Then there are two layers that actually coincide in their conductivity values, since one of these layers was create for the electrodes. Let the values of electrical conductivity for them be 0.001 S/m. The next object-environment is the bottom layer of earth with a conductivity of 0.1 S/m. After this layer, three metal objects — two U-shaped abutments and a pipe on them. The electrical conductivity of the metal is 7690000.0 S/m. Then there are the thawing zone. Let its electrical conductivity is 0.01 S/m. Next, the metal bore well, so the electrical conductivity here is again 7690000.0 S/m.

Finally, create a new elec_profiles.txt file by clicking the button  . Having chosen the necessary electrical tomography array (in our case it is Wenner-Alpha), we recall how we locate the electrodes and how many of them: X0 = 50, Y0 = 50, Z0 = 0, Number of electrodes in one profile — 21, Number of profiles — 11, X_step — 5, Y_step — 5. Click the “Generate” button and a new elec_profiles.txt file will be create. For the sake of interest, you can open it (via the drop-down menu “FILE OPEN”) and find out that the total number of problems that we have to solve is 693. Make sure that the number of zero Dirichlet boundary conditions have value 76. To do this, open the file null.txt (through the dropdown menu “FILE OPEN”). Now we are ready to generate a matrix of a system of linear algebraic equations. Click the “Start PREPROCESSOR (assembling matrix)” button. After the matrix creation process is completed, you can begin to solve the direct problems, but first we will look at the portrait of the matrix by clicking on the “matrix portrait” button:



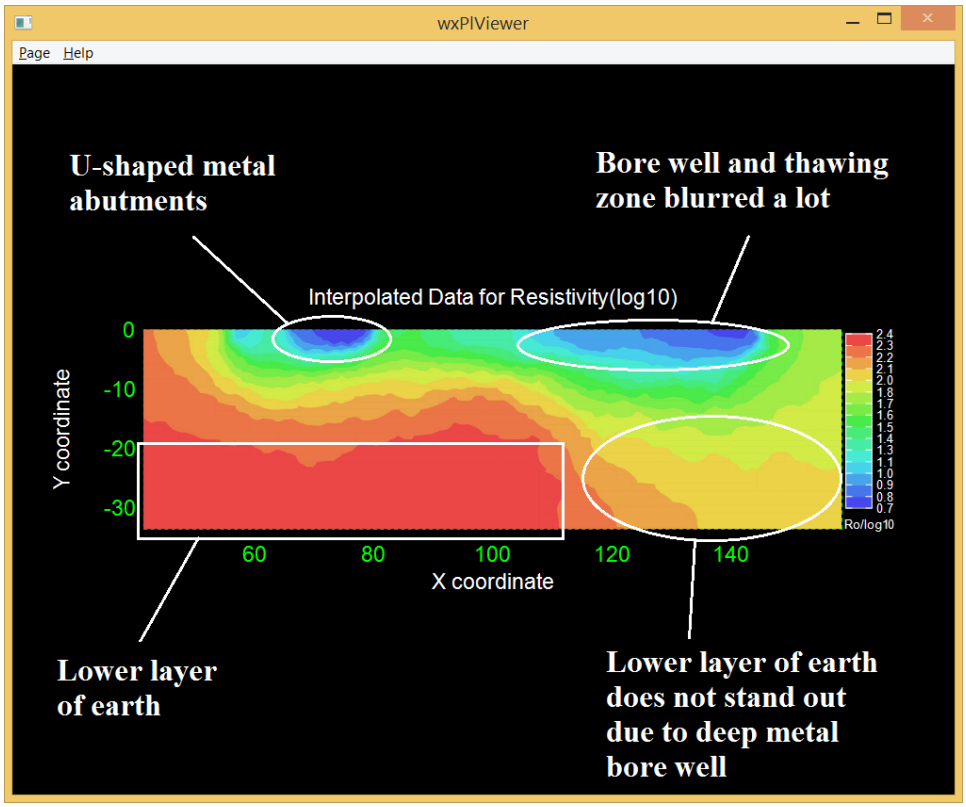
Thus, the numbering of the mesh nodes in NetGen leaves much to be desire. However, there is nothing terrible in such a portrait. The presence of elements close to zero on the diagonal is much more terrible (they are marked in red). Their presence is because there are strongly contrasting objects-environments. However, no matter how bad the portrait of the matrix looked, such a problem must be solve and we can do it.

Suppose that we want to solve all problems at once on all profiles. Enter in the field “Y-coord. of profile:” the value of “x” without quotes, as the first problem — 1, and as the last problem — an arbitrary large number, for example, 100000. Click “START DIRECT solver” button. The process of solving direct problems was initiate.

After the end of the solution of direct problems (it will take several minutes), we press the “bring together” solution composition button, after which all the solution files will be in the “my_difficult_test” directory (as we called our problem), which in turn is located in the program DiInSo directory. In the folder with the mesh for the direct problem, new files will appear — for solving the inversion problem in 2D and 3D cases. Let us turn to these problems...

Let's solve a two-dimensional inversion problem, for example, on the eighth profile, for which we select the file profile8.bert, clicking the “DATA FILE:” button. Do not forget to choose the type of problem “2D”. Set the other settings as follows: “size of cells” — 0.25, “mesh growing” — 35.0 (we need a detailed mesh), “recalc jacobian?” — 1, “optimize reg?” — 1 (let the regularization parameter be selected using the L-curves method), “enh contrasts?” — 1 (highlight contrasting objects), “min app res” — 0, “max app res” — 1000 (exclude unnecessary values of apparent resistances, if any). Press the button

“INVERSION” and we can see in the console window that the DiInSo program has performed 4 iterations, the convergence is rather bad. The reason is in a small number of electrodes on the surface (for such a problem) and in a complex model, where there is a lot of noise which created by metal objects. However, our goal was to evaluate the effect of metal interferences on the result. Let’s press the button “vis. inv.” and see the results... It is better to look at logarithmic results, since we have very strongly contrasting objects and environments:



Analysis of the results shown in the figure. We would not be able to do such an analysis without modeling a direct problem with known metal structures. Such modeling in practice would help us avoid mistakes and wrong conclusions when solving the inversion problem in the “real research territory”.

Then you can see for yourself the results on the other profiles and in the 3D case.

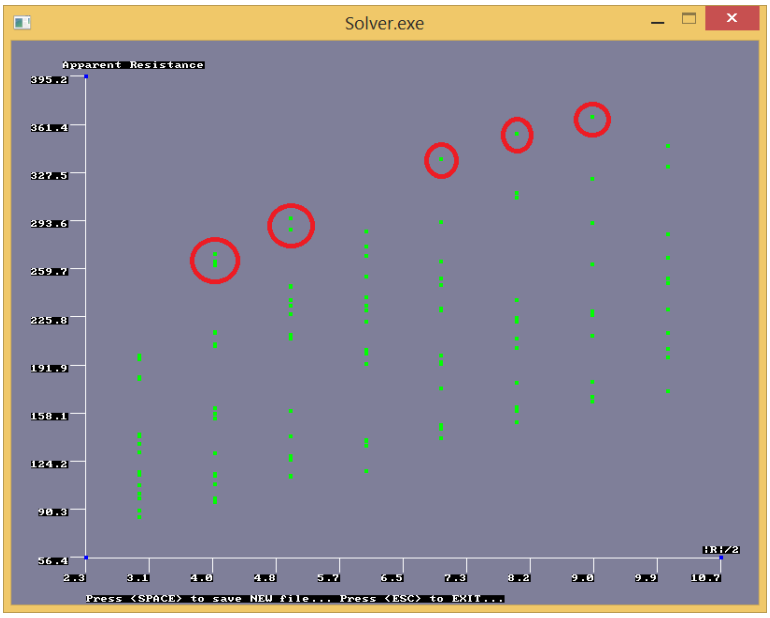
An example of solving simple 2D inversion problem.

These examples taken from the set of examples of the software https://gitlab.com/resistivity-net/bert, while minor changes made to the input data file for better compatibility with all features of the DiInSo program. The input files are in the “examples” folder of the DiInSo program and are call simple_test.bert, simple_testERR.bert, simple_testIP.bert, simple_testIPERR.bert. All these files are intend for 2D inversion. The file simple_test.bert contains simple data for

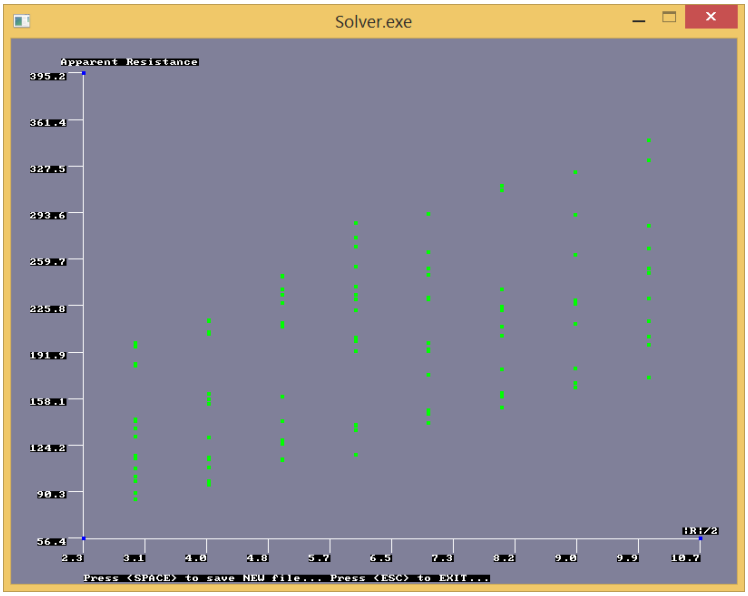
inversion, the file simple_testERR.bert contains data for inversion, where evaluation of measurement errors presented, the file simple_testIP.bert contains additional data of induced polarization, and finally, simple_testIPERR.bert contains both data of induced polarization and evaluation of measurement errors. Since we are considering a simple example, we will not set any additional parameters for 2D inversion at all. Therefore, we set only the type of inversion “2D” and sequentially examine each of the input files by selecting them using the “DATA FILE:” button. So, let's start...

File simple_test.bert:

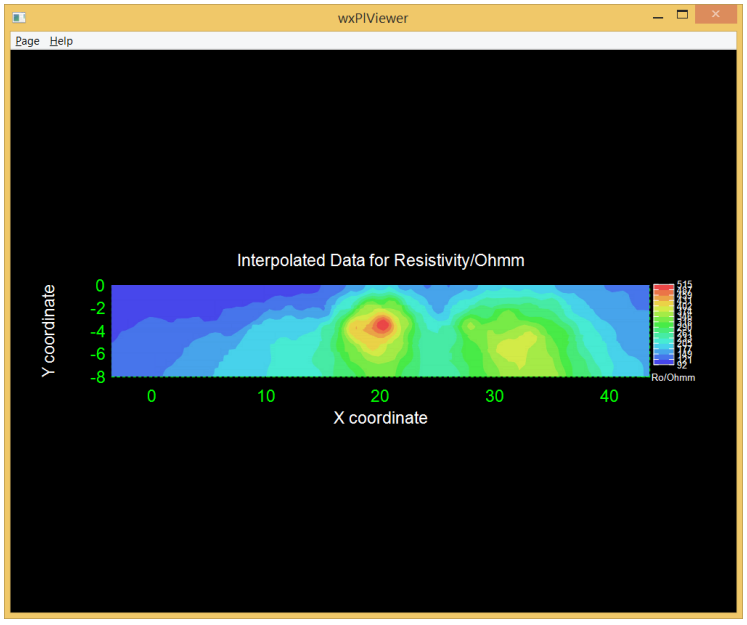
To make it more interesting to solve the inversion problem, let's work with the input data. Let's try, for example, to exclude “unnecessary” data. Select the value “graphic” in the “Exclude arrays from:” field, press the “Exclude” button and you will see the following picture:



Suppose we have some scientific assumptions that allow us to conclude that the points marked with red circles are break out of the overall measurement pattern. We exclude them by surrounding them with closed curves and we get:



Press <SPACE> on your keyboard and get a new file simple_testnewgraph.bert, exit the window of excluding results by pressing <ESC>. Open the new file simple_testnewgraph.bert using the "DATA FILE:" button. We press the button "INVERSION" and sufficiently quick get the result with very good convergence. Let's look at the results of the inversion by pressing the button "vis. inv.":



Next, you can experiment on your own — remove data points, open the resulting .vtk files with the VTK-viewers, enter some parameters for the inversion and so on.

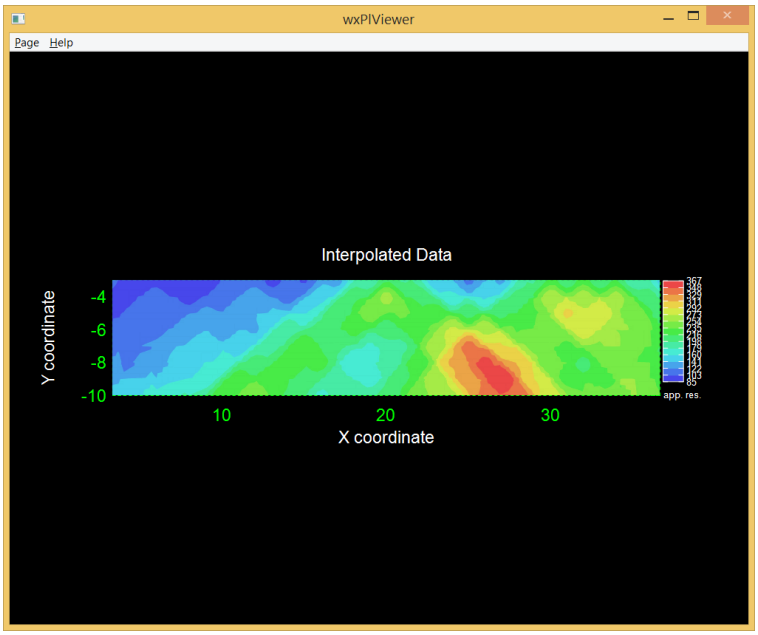
File simple_testERR.bert:

This file is not much different from the previous one; it just has additional data of measurement errors. Let's remember what needs to be done if we do not

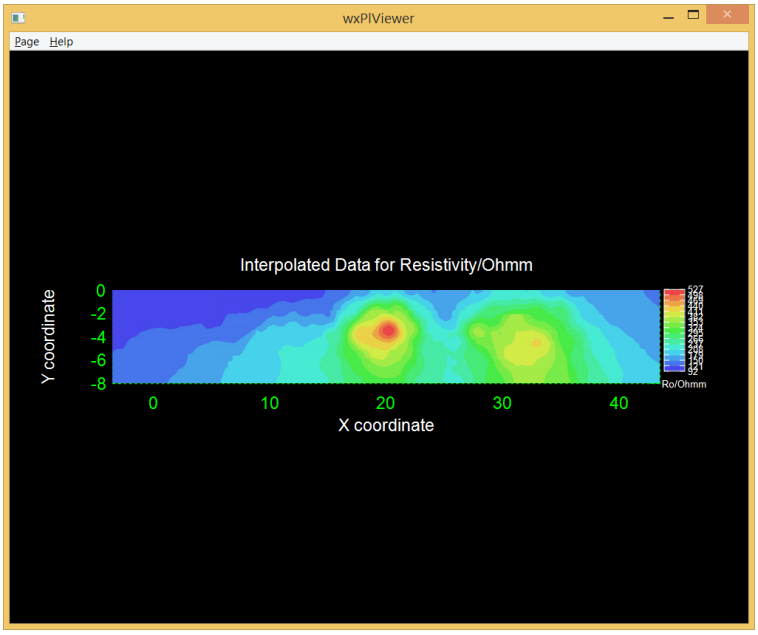
need this data of measurement errors and we want to make new “automatic” estimates of measurement errors. That's right, you need to set the “estim err” parameter to 1, and then the measurement error data recorded in the input file will be recalculated during the operation of the inversion algorithm.

File simple_testIP.bert:

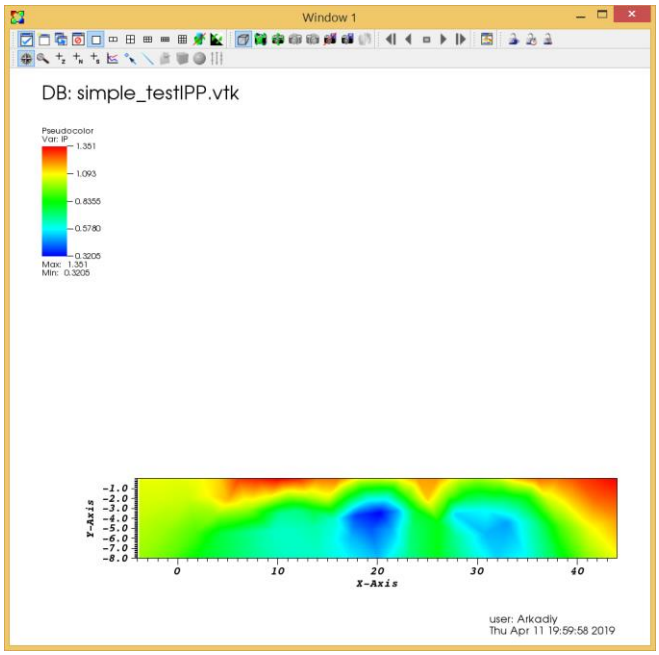
Let's look at the input data graphically by clicking the “vis. app. res.” button:



Note that despite the fact that the input file contains induced polarization (IP) data, on the picture you can see data of apparent resistances. The reason is that IP data are of little interest in such a view, they will interest us like a result of solving inversion problem. Start an inversion process by clicking the “INVERSION” button, and look at the results by clicking the “vis. inv.” button:



Again, you see the results of the inversion with the data of the resistances, not the induced polarization (IP). To see the results of the inversion of the IP, you need to open one of the resulting output files in a third-party visualizer. For example, we will work with *.vtk files, as with the most convenient format, according to many scientists. We use the program VisIt, the link to which is above in the text. Open the file simple_testIPP.vtk, select the button “Add” -> Pseudocolor -> IP in the VisIt menu, press the button “Draw” and look at the result:



For the sake of fairness, our IP data entered into the input file NOT by real values, so no need to try somehow interpreting this result.

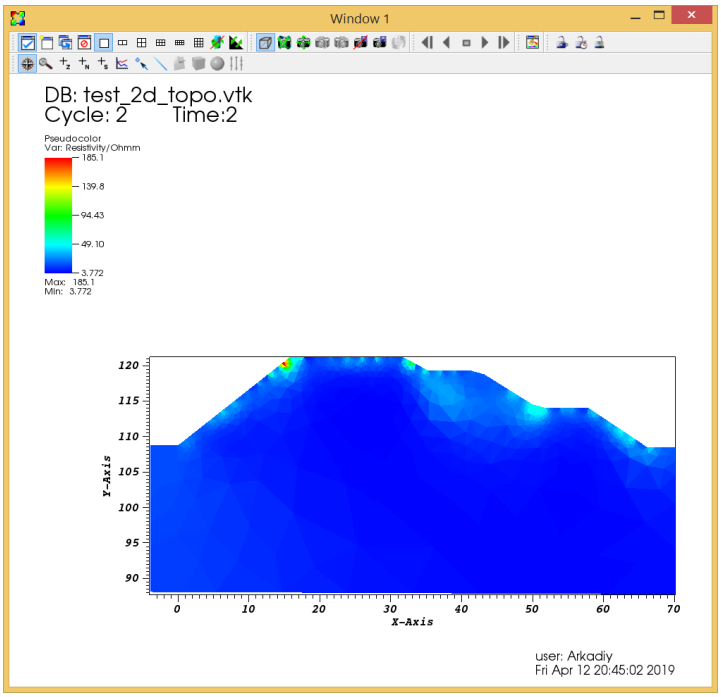
File simple_testIPERR.bert:

This file is an example of the richest data set — here you can see both induced polarization (IP) and measurement errors as additional data. However, the uniqueness of this file ends there. Work with it by yourself.

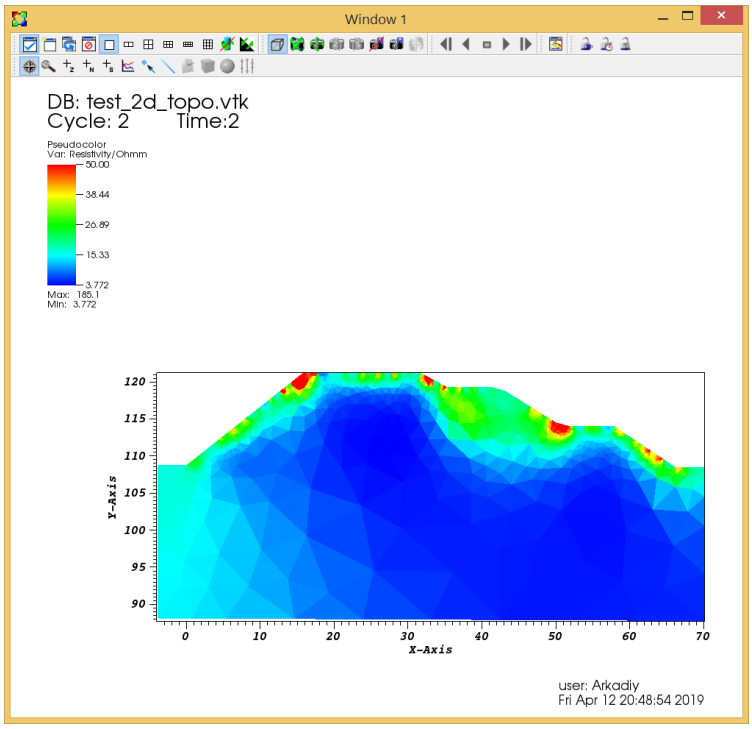
An example of solving a 2D inversion problem with a topographic effect.

This example taken from the set of examples of the software https://gitlab.com/resistivity-net/bert, while minor changes made to the input data file for better compatibility with all features of the DiInSo program. The input file is in the “examples” folder of the DiInSo program and it name is test_2d_topo.bert. This file is intend for 2D inversion. First, set the parameters for 2D inversion. Let “size of cells” — 0.25, “mesh growing” — 34 (we need a rather detailed mesh), “recalc jacobian?” — 1 (we will recalculate Jacobian at each iteration step). Next, select the input file using the “DATA FILE:” button. The attentive reader noticed that we did not set the topography parameter “topography”. Yes, we really did not

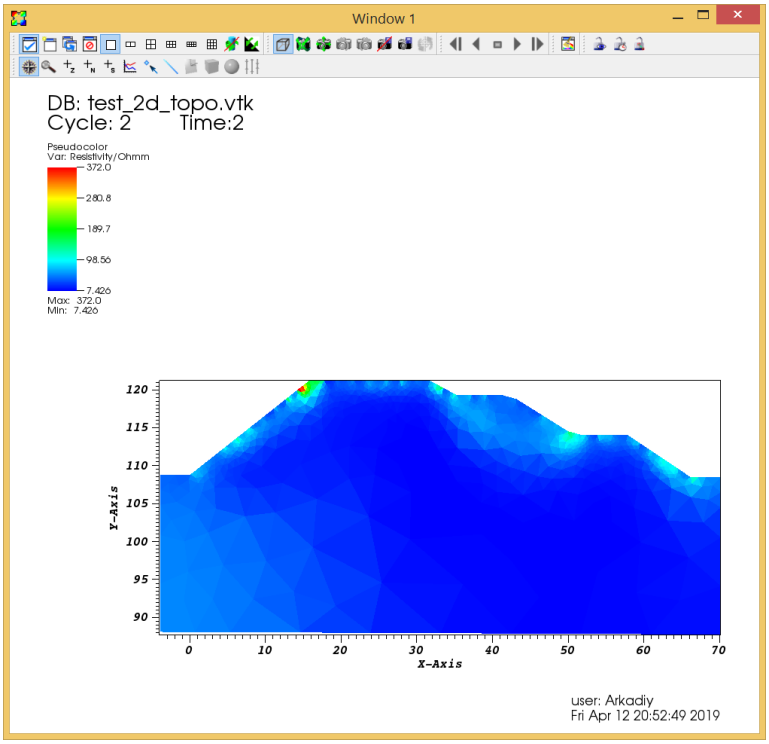
do this and want to see the result without taking into account the topography effect. So, press the “INVERSION” button and wait for the end of the inversion process. In the console window, among other things, there will be the inscription “TOPOGRAPHY FOUND BUT TOPOGRAPHY FLAG DISABLED!”. That is why it is sometimes important to read the information in the console. The program reports a possible user error, but the inversion process continues, because taking into account the topography effect in this case is not critically necessary. Next, we can see the results by clicking the “vis. inv.”, but it is better not to do it and now you know why. The visualizer built into the program ALWAYS depicts a rectangular area; therefore, the data will be interpolate to those parts of the rectangle area where they are actually absent due to the topography effect. Therefore, when we deal with topography effect, it is better to use more professional visualization packages, for example, VisIt, which was mention above. Let’s open in VisIt the test_2d_topo.vtk file, select the “Add” -> Pseudocolor -> Resistivity/Ohmm button in the VisIt menu, press the “Draw” button and look at the result:



Not so good representation of the results due to the small area with higher resistances. Open the menu “Pseudocolor — Resistivity/Ohmm” by clicking on the arrow to the left and click twice on the word “Pseudocolor” that appears. Let’s limit the upper limit of values, for example, to 50. Now the picture has become much more informative:

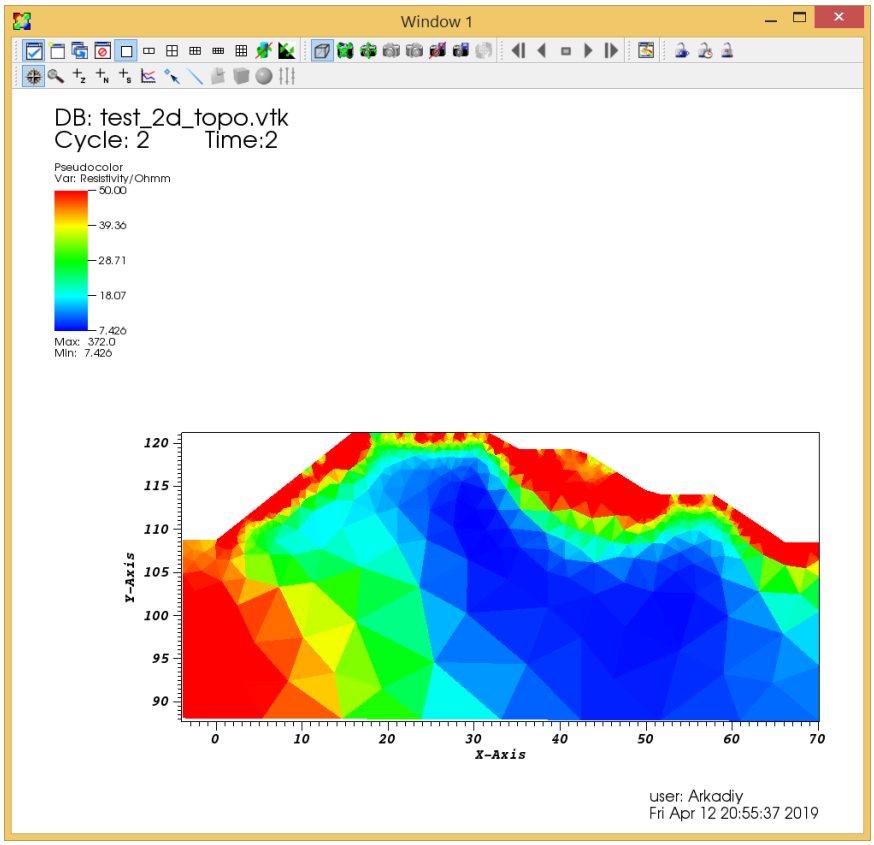


However, we repeat, on this graph did not take into account the topography effect, even though the area looks like the area with topography. Let's check can you trust these results or not. To do this, we solve the problem with a topographic effect, setting the parameter “topography” to 1. Click “INVERSION” button and then repeat all the procedures described above. We got the following result:



Once again, it is not good representation of the results due to the small area with higher resistances; however, the maximum value of resistance is now much

bigger. Let's limit the upper limit of the values to the same number 50 and get next results:

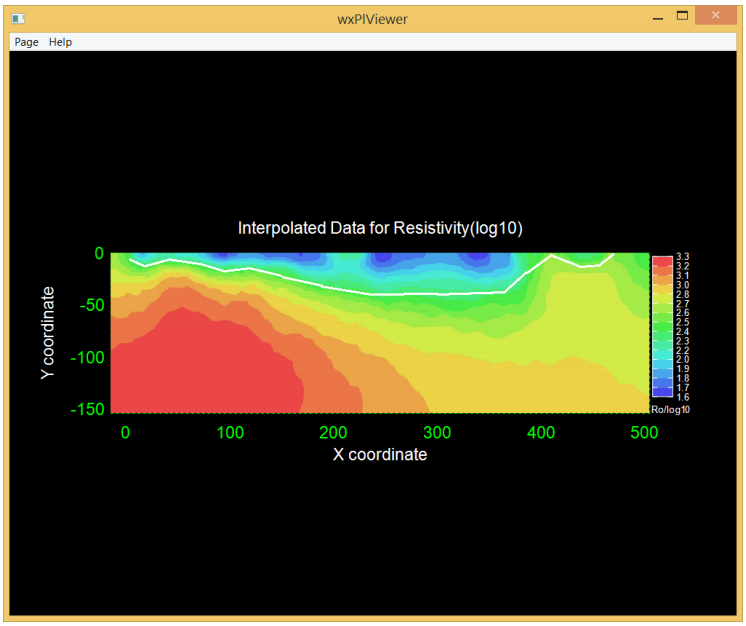


We can see that from the point of view of the subdomain geometries our results were similar in the case of taking into account and NOT taking into account the effect of topography, but from the point of view of the resistance values, the differences are quite significant, even though that we used the same settings for the inversion (except for the setting for the topography effect). In fact, you can find other examples that show that not taking into account the effect of topography leads to a noticeable curvature of the subdomains, but, as we have shown, there are other cases. For example, for the problem above: if the geometry is more important to you than the values of resistances, then forgetting about the topography you will not lose much. However, once again, this is not always true!

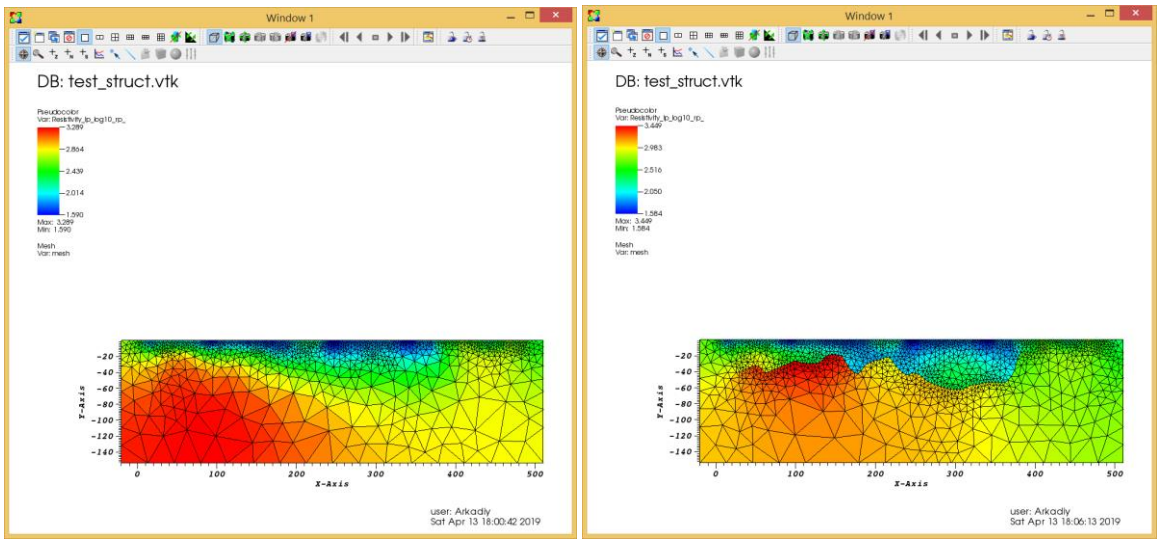
An example of solving a 2D inversion problem with the selection of the boundaries of regions with different resistances

This example taken from the set of examples of the software https://gitlab.com/resistivity-net/bert, while minor changes made to the input data file for better compatibility with all features of the DiInSo program. The input file is in the "examples" folder of the DiInSo program and it name is test_struct.bert. This file is intend for 2D inversion. First, set the parameters of 2D inversion. Let "size of cells" be 0.25, "mesh growing" — 34 (we need medium detail mesh),

“recalc jacobian?” — 1 (we will recalculate Jacobian at each iteration step). Next, select the input file using the “DATA FILE:” button. This test differs from the previous ones in that we know the complex boundaries of the regions (in this case we mean the boundaries of the continental rock), which are contained in the file test_struct.xz. Select this file using the “topo/line file:” button. First, we will not activate a checkbox from the left of the “topo/line file:” button and look at the result of the inversion without information about the boundaries of the regions. Let’s press the button “INVERSION”, then press the button “vis. inv” and look at the result (we will look at logarithmic values so that the boundaries of the regions are more clearly visible):



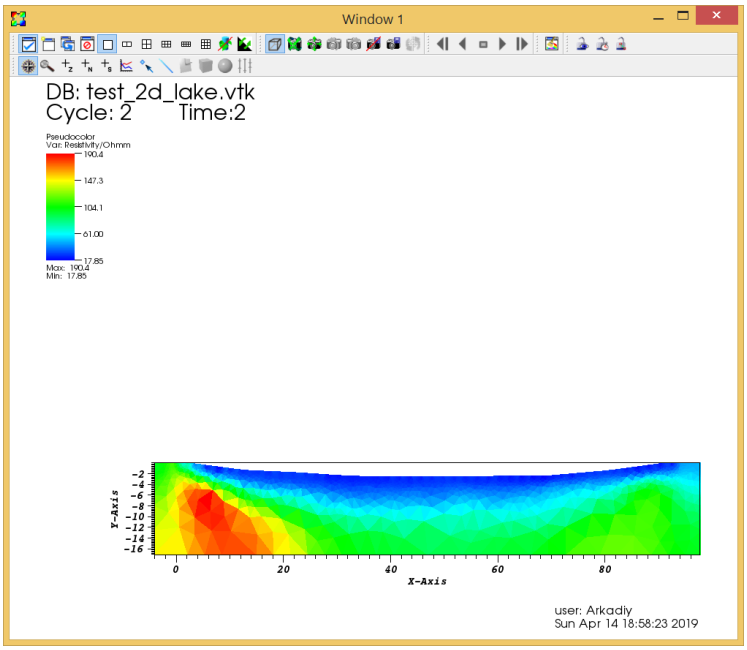
In the picture with the results, we can roughly distinguish two distinct subdomains even without applying our knowledge about boundaries (we selected the approximate boundary between the regions by the white line). Now we solve the inversion problem with information about the boundaries of the regions and compare the two results by looking at them in the VisIt visualizer (in VisIt it is more convenient to compare, since it can simultaneously view the results and the mesh: Add -> Pseudocolor -> Resistivity_lp_log10_rp_, and after that — Add -> Mesh -> mesh). So, on the left — logarithmic values without information about the boundaries, on the right — logarithmic values with information about the boundaries (it is better to open the file WITHOUT the “P” postfix and look at the values associated with cells):



We can see that the selection of file with the boundaries of regions with different conductivities helped us not only divide subdomains much more obvious, but it also helps to get rid of smooth transitions between boundaries. In fact, we have a result with relatively sharp transitions of resistance values, which is almost impossible to achieve if we do not use preliminary data with the boundaries of subdomains.

An example of solving a 2D inversion problem with well-known information about model subdomain.

This example taken from the set of examples of the software https://gitlab.com/resistivity-net/bert, while minor changes made to the input data file for better compatibility with all features of the DiInSo program. The input file is in the "examples" folder of the DiInSo program and it name is test_2d_lake.bert. This file is intend for 2D inversion. First, set the parameters of 2D inversion. Let "size of cells" — 0.25, "mesh growing" — 34 (we need a rather detailed mesh), "recalc jacobian?" — 1 (we will recalculate Jacobian at each iteration step). Next, select the input file using the "DATA FILE:" button. In this example, the measurements were carry out on an uneven bottom of the lake. Therefore, we need to set the topography parameter "topography" as 1. First, we solve the problem without taking into account the fact that there is a layer of water above the electrodes. We press the "INVERSION" button and look at the results in the VisIt program (recall that the data with the topography is better to look in program VisIt):

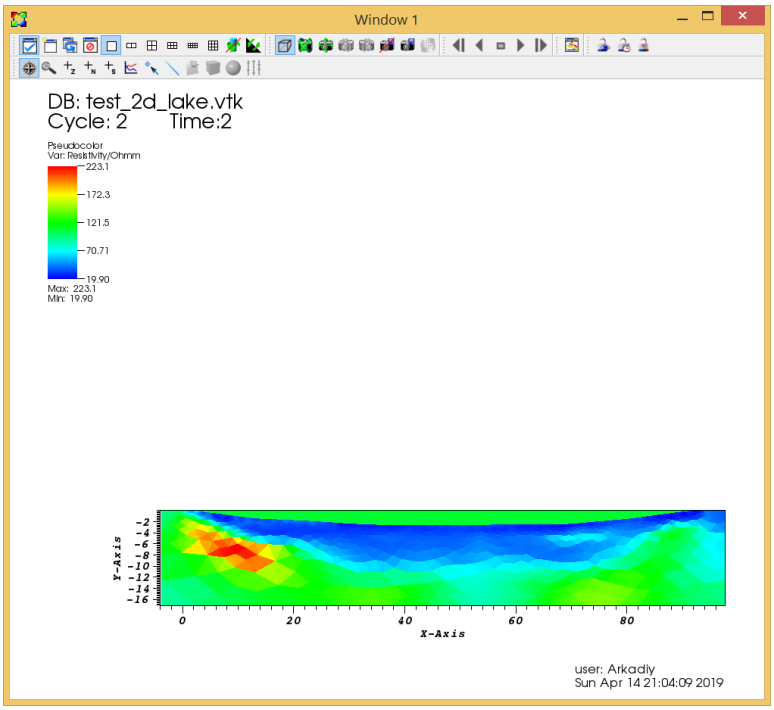


Now we take into account that there is a layer of water above the electrodes. To do this, we need to make small changes to the geometry of the model, but first we need to get the geometry itself. Click the button “geom+mesh” and get the file test_2d_lakeGEOM.poly. Open it in a text editor. To set the top layer of water, you need to connect the extreme points with a depth of zero, in our case these are the points “4 2.0000000000000000e+00 0.00000000000000e+00 -99” and “184 9.17452000000000e+01 0.00000000000000e+00 -99”. Having studied the format *.poly we know that we can connect points by adding one new edge. Thus, we increase the number of edges from 198 to 199, and at the end of the list of edges we add the line “198 4 184 -1” without quotes. We used the edge marker “-1” because we added an upper border. At the same time, it is not necessary to change the markers of the other edges with the value “-1”, since this is the boundary of the subdomain inside the solution area (recall the *.poly format). Now mark our new subdomain. Let us increase the list of areas by 1, that is, they will now be 3 (the area where the solution is of interest, the area for far away the external boundaries where the solution is NOT of interest to us and water layer). As we remember, the area is marked with an internal point, so we add the following entry to the list of areas: “2 50 -1 3 0.0”. The marker for the new area will be 3, as markers 1 and 2 are already occupied. A new geometry file is ready, and we will to use it in future. Select the file with geometry created by us by clicking the “geometry file:” button; activate checkbox to the left of this button.

Now we will define a regions information file with the extension *.control. Create it manually using any text editor and give any convenient name with the extension *.control. Write the following parameters to the file:

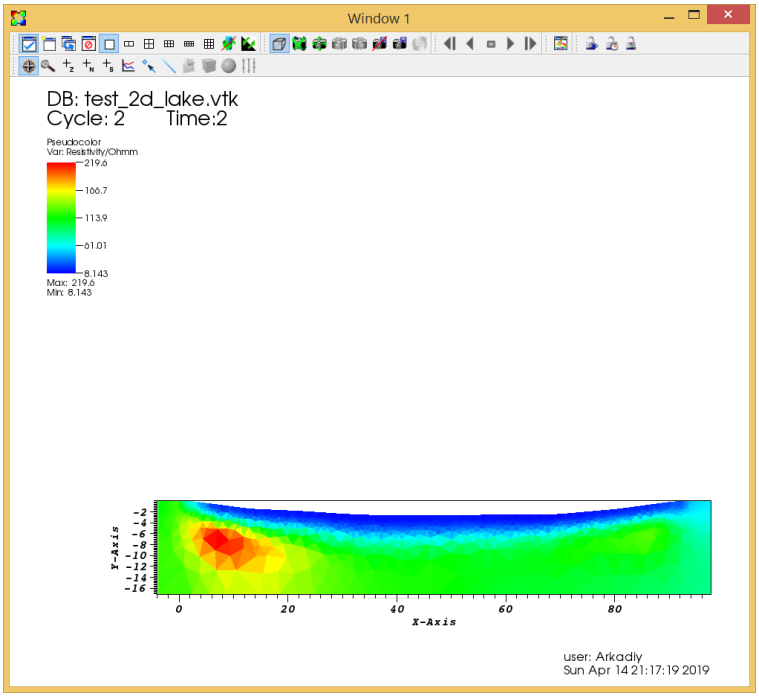
```
#No start Trans zWeight Ctype lBound uBound
2   100   log   0.1     1     10     1000
#No single start
3   1       22.5
#No background
1   1
```

That is, we set the main region as “moderately horizontal”, but containing some objects as inclusions. The range of resistance of the region extends from 10 to 1000 Ohm*m with a starting value of 100 Ohm*m. We mark the water layer as an area with a constant resistance value, which starts from a value of 22.5 Ohm*m (true water resistance value). Region number 1 is the area where the solution does not interest us. Select the file with regions created by us by pressing the “region file:” button; activate checkbox to the left of this button. Let’s start an inversion by clicking the “INVERSION” button and look at the result in VisIt program:

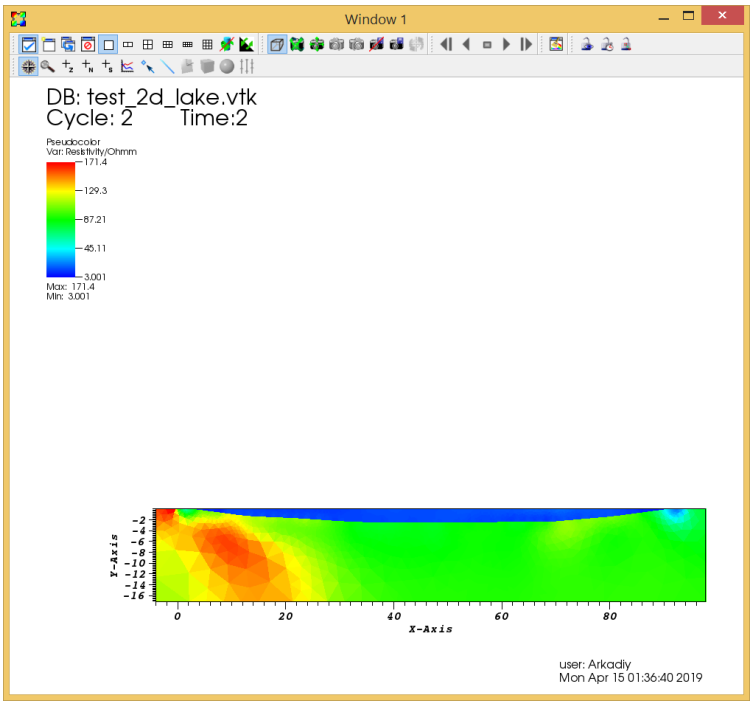


We can see that the inversion algorithm gave a slightly different distribution of resistances below the bottom of the lake, but the resistance value of the lake itself was far from real (22.5 ohm*m). What could be the reason? Oddly enough, the reason for this is that we have not disabled the topography option. Above, we discussed that the option which enable topography recounts the geometric factors of electrical tomography arrays and, accordingly, apparent resistances. However, it sometimes happens that the researchers did it earlier, because the topography was not complicated and the input apparent resistances is already correct and does not require recalculation. Thus, enable the topography option only spoiled the result.

Let's go back to the original problem, when we did not take into account the water layer and recalculate everything anew without the topography effect (“topography” — 0):

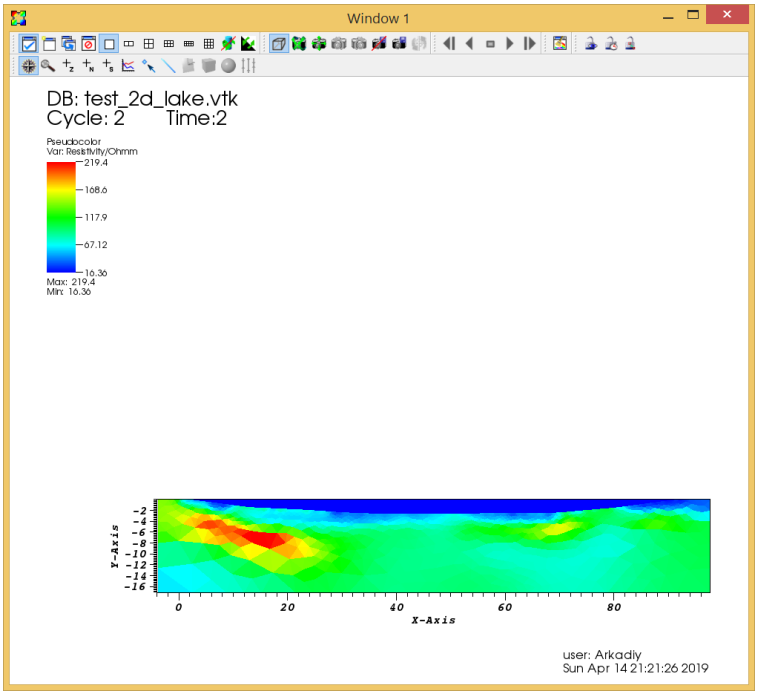


The result is noticeably different. What will happen if you solve an inversion problem without knowledge of the areas, but knowing the boundary between water layer and the remaining area? We obtain the following result:



The inversion algorithm correctly determined the homogeneity of the upper aquatic environment, although it underestimated the values of water resistance, but

let’s check the results below the bottom. To do this you need to try to solve the inversion problem with information about the water layer. After we take into account water layer as it was describe above, the following result will be obtain:



First of all, we can see that water resistance has become much closer to reality. In addition, information about the water layer allowed us to obtain a significantly altered (in this case, more adequate) model below the bottom. Thus, we have seen an example where the presence of one region with known characteristics not only clarifies the entire model, but also allows us to understand that the input data is inadequate, or that we work with them incorrectly. In general, the more we know about the regions with which we work — the more chances to solve the inversion problem correctly. It would seem an obvious idea, but often forgotten by researchers.

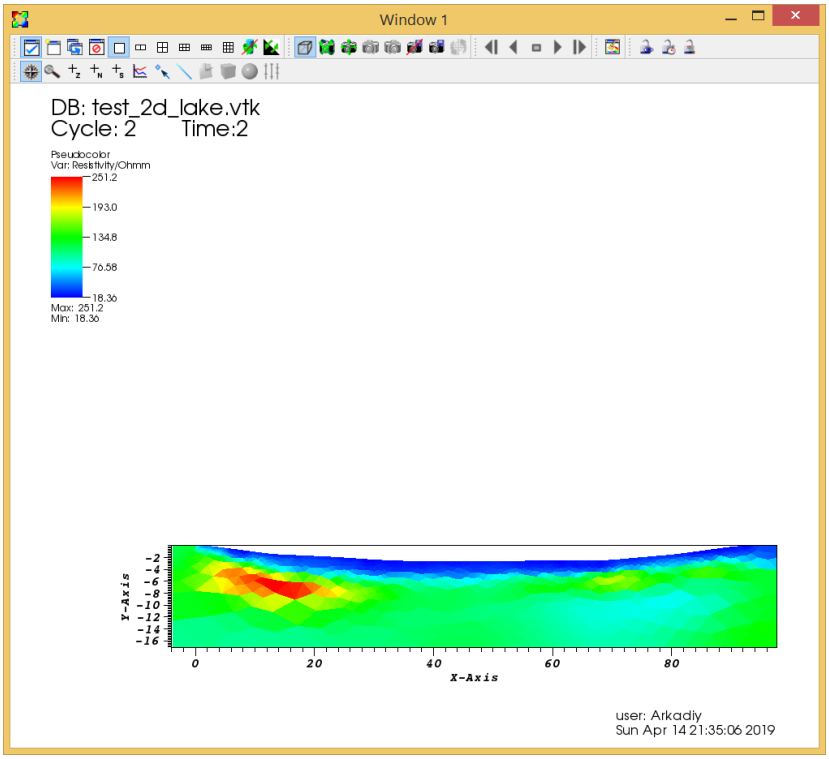
What other options are there for setting the water layer using the *.control file? For example, such:

```
#No single start Trans lBound uBound
3   1      22.5  log   22     23
```

That is, we set the region with clear boundaries of the resistances that interest us. Another one variant:

```
#No fix
3   22.5
```

Recall that with this approach, we “inform” the inversion algorithm fixed resistance value of the region (22.5 Ohm*m), and the calculations themselves will take place without calculations inside this region (the values are strictly fixed and do not change). Since water is exclude from calculations, there will be no water layer in the results:



The result turned out to be somewhat similar to that when we performed calculations with the water layer excluded, but it has a number of important differences in the size and shape of the anomalies. Finally, you can set the rules for the transition between the water layer and the remaining area, because this transition is usually occur (water -> wet underlying zone -> dry underlying zone):

```
#No single start
3   1       22.5
#Inter-region
2 3 0.5
```

In addition, we can assume that the water layer has a heterogeneous structure (for example, due to a silt layer or a variable temperature) and specify this layer as we set the base layer, and also use another regularization parameter for calculations inside the water layer, like this:

```
#No start Trans zWeight Ctype lBound uBound MC
2   25    log   0.1     1     10     1000   1
3   22.5  log   0.01    1     10     30     3
```

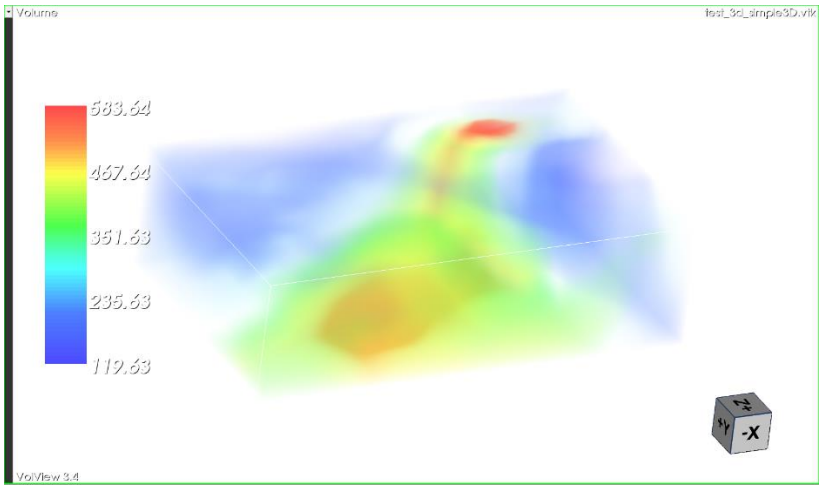
```
#Inter-region
2 3 0.5
```

Try these variants by yourself.

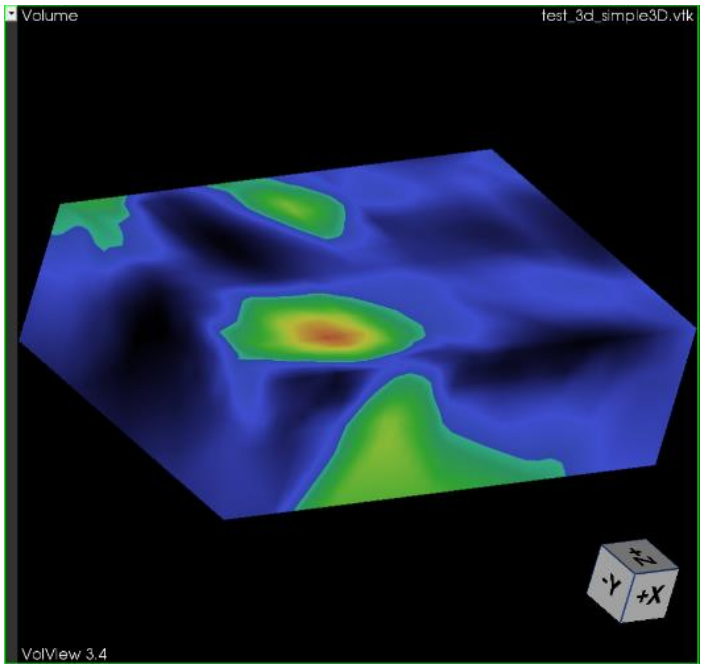
In other words, there can be a great many ideas for the *.control file, the main thing is to understand what we want to get and have some information about the region where the measurements were taken.

An example of solving a simple 3D inversion problem and a 3D inversion problem with well-known information about the boundaries of subdomains.

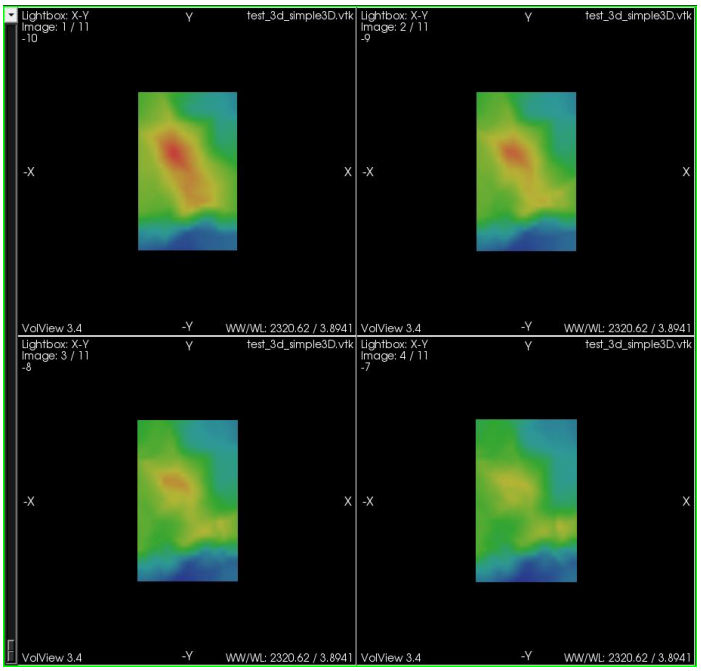
This example taken from the set of examples of the software https://gitlab.com/resistivity-net/bert, while minor changes made to the input data file for better compatibility with all features of the DiInSo program. The input file is in the “examples” folder of the DiInSo program and it name is test_3d_simple.bert. This file is intend for 3D inversion. First, set the parameters of the 3D inversion. Let “quality” — 30, “mesh grow” — 2 (we will not create too detailed mesh to save memory), “model depth” — 10 (determine the depth of research on the assumption of some knowledge), “recalc jacobian?” — 1 (we will recalculate Jacobian at each iteration step), “regular strength” — 5 (determine the regularization parameter independently). Next, select the input file using the “DATA FILE:” button. The number of profiles in this three-dimensional problem is 9, there are 14 electrodes in each profile, the number of array positions is 753. You can start the inversion. Press the button “INVERSION”, the solution will be obtain in 5 iterations with a fairly good convergence. I suggest, for a variety, to look at the results of the solution in the VolView program, which was discuss above, by opening the file with the results test_3d_simple3D.vtk. In VolView, select the “Scientific/General data” data type. By adjusting the “Scalar Opacity Mapping” and “Scalar Color Mapping” parameters, you can highlight the objects you need and, due to the transparency settings, as well as the model rotation, see the full picture of the distribution of layers/objects:



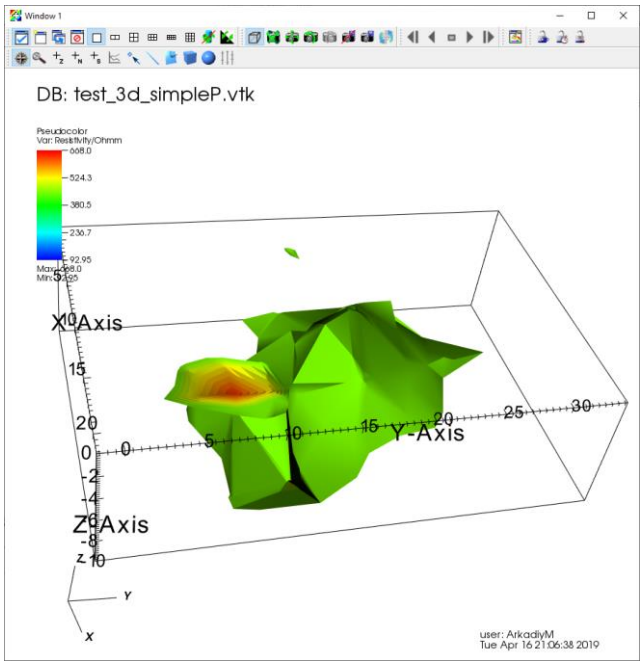
If you cannot adjust the transparency, then most likely, you are using an old or built-in (simple) video card, in this case you will see something like:



You can also view the results in the different sections:



Opening the results in VisIt, you can see surfaces of equal level (Operators -> Slicing -> Isosurface). For example, if we are interested in surfaces of equal level above the value of 450 Ohm*m, then they will look like this:

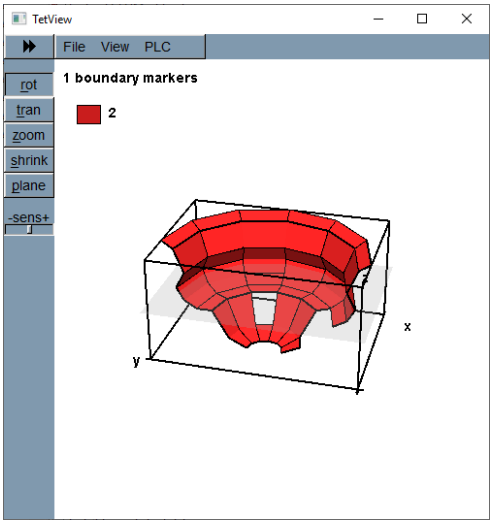


Now let's assume that we have information about the boundary of some internal area and we need to add this boundary to the model. Suppose this boundary is rather complicated and we built it in some three-dimensional editor, for example, in the free opensource “Blender” editor (https://www.blender.org/). We will not practice creating three-dimensional models, but simply use the already prepared object from the Blender tutorial located at: https://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/Creating_a_Simple_Hat

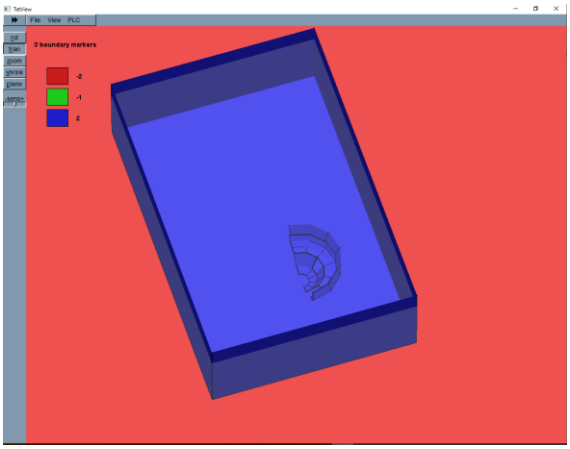
Using this example, you can create a similar object and, saving it, for example, in the format *.obj, and then convert it to the format we need — *.poly, using  button. If the position, size and rotation of the object were not take into account in the program of 3D modeling, then all these manipulations available here in DiInSo. Markers of nodes and faces (first two fields) of our “boundary of object” we are set as 0 (internal node number) and 2 (number that coincides with the boundary number of the area where the solution is searching), since these are the rules for working with the *.poly format for TetGen and these rules have been described above.

Next, create the geometry of the main modeling area by clicking the “geom+mesh” button and receiving the file with the geometry test_3d_simpleGEOM.poly. It would seem that now it is possible to combine our geometries by pressing a button  , but, unfortunately, not everything is so simple. The problem is that many three-dimensional objects generators do not pay attention to the problem of mesh consistency. In other words, some polygons contain, for example, coincident points or, even worse, points that lie very close to each other and almost coincide. In this case, there may be an intersection of polygons and, as a result, an even greater inconsistency of the mesh. Frankly speaking, it is desirable to correct such polygonal models even from the point of view of computer graphics (in the case of computer graphics, to reduce the size of the model and save PC memory), but this is not always done. One of the programs for working with ready-made meshes, including fixing them, is the freely distributed “MeshLab” program: http://www.meshlab.net/. Usually (but not always) to fix the mesh, it is enough to realize a few simple steps in the MeshLab program. Open the mesh with the “Import Mesh” button (or use the Ctrl+I keyboard keys combination), select the menu item Filters -> Clearing and Repairing -> Remove Duplicate Faces — delete the matching faces, then Filters -> Clearing and Repairing -> Remove Duplicate Vertices — delete the matching nodes, then Filters -> Clearing and Repairing -> Merge Close Vertices — merge closely located nodes (you can leave the default settings, or you can work with the settings) and, finally, Filters -> Clearing and Repairing -> Remove Unreferenced Vertices — delete nodes which are NOT part of the object. You can check the created object for correctness by selecting the menu item Filters -> Clearing and Repairing -> Select Self Intersecting Faces — select intersecting faces. Obviously, there should be NO such faces; otherwise, the object is NOT correct in our sense. Then, if everything is normal, we save the mesh, for example, in the *.ply format using the “Export Mesh As” button (or by pressing Ctrl+Shift+E keyboard keys

combination). And the newly received *.ply file we will convert into the required *.poly format with the button  . As mentioned above, we will not consider the subtleties of 3D modeling here; you will have to learn them by yourself. However, the fact remains fact: there is no correct mesh — there is no working final model. The hand-made corrected (not ideal way) object model *.poly can be find in the already known to us folder “examples” and has the name test_3d_object.poly. This object can be view, for example, in the “TetView” program (http://wias-berlin.de/software/tetgen/tetview.html), which was discuss above:

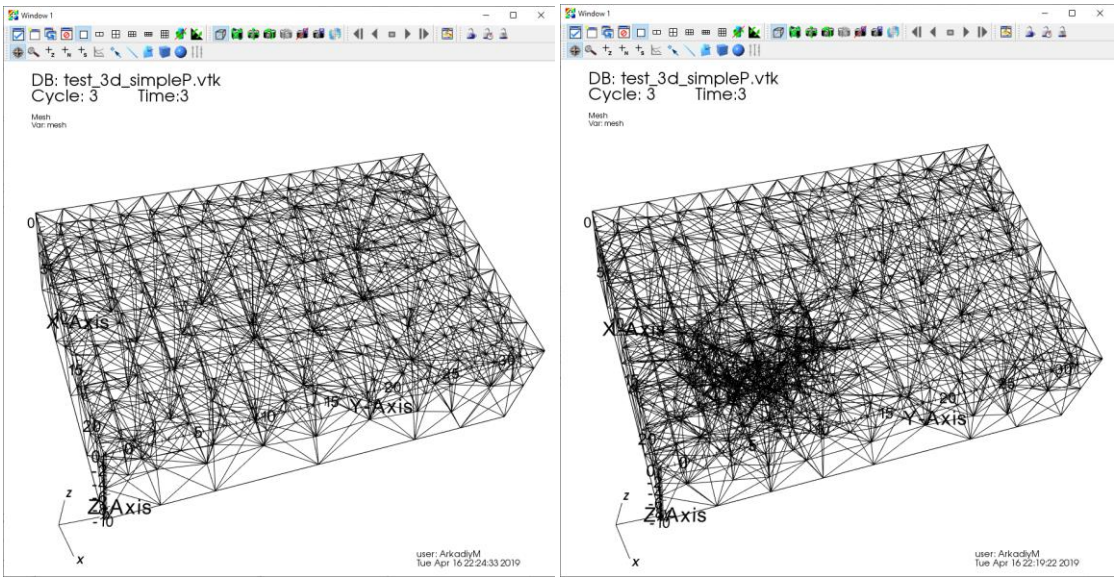


We will combine two geometries — test_3d_simpleGEOM.poly and test_3d_object.poly using button  . A new file will name, for example, my_new_geom.poly. Let's look with the help of TetView, what final model we got, having increased the area with the included object:

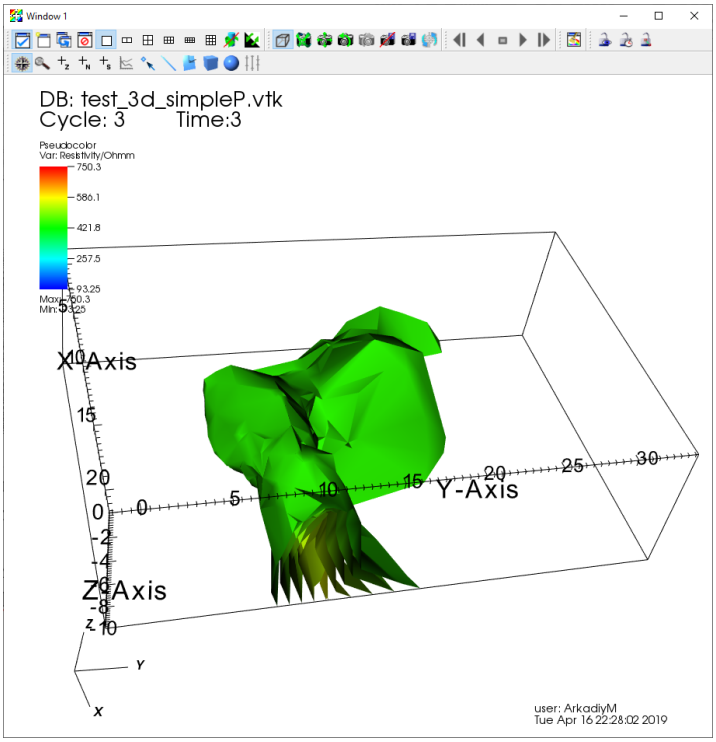


So, we have included our object–“boundary” inside the modeling area. Now select the my_new_geom.poly file using the “geometry file:” button, without forgetting to activate the checkbox to the left of the button. Let's carry out modeling with the help of a new geometry. Press the button “INVERSION” and

wait for the end of the inversion process (note that the convergence has become better). First, we compare the two meshes of the resulting solution by opening them in the VisIt program (Add -> Mesh -> mesh):



We can see how the included object–"boundary" is clearly highlight on the right picture. This means that the inclusion of the boundary of the region has passed correctly. Now let's look at surfaces of equal level above the value of resistance 450 Ohm*m that we observed earlier (Operators -> Slicing -> Isosurface):



As we can see, the result is slightly different at the area where our object–"boundary" located. This is primarily due to the fact that a smaller, and therefore

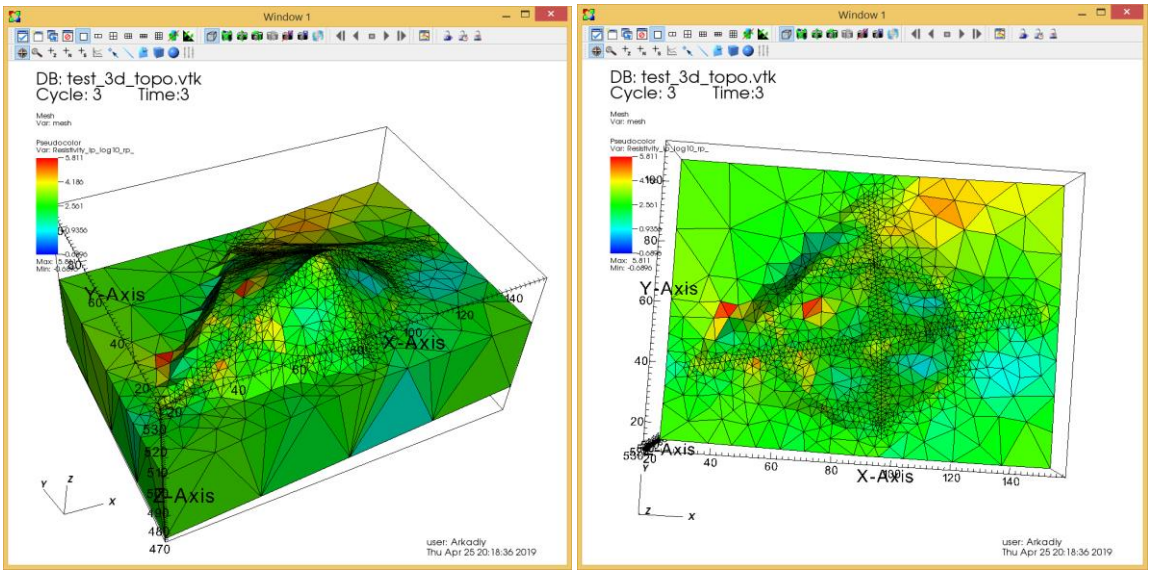
more detailed mesh was generated in this area, as can be seen in the mesh picture above (this is why the convergence became better in the case of the included object–"boundary" and was not so good without them — the mesh was too rough without an object). In this case, our object–"boundary" has no other effect, because it was chose in a completely arbitrary manner and was include in an arbitrary location.

In the case when we want to include in the modeling area not an object-"boundary", but an integral (filled) three-dimensional object, we would act in the same way, with the only difference that in the case of a three-dimensional filled object we would need a closed polygonal model; it was necessary to know the coordinates of any point inside the filled object to set it when the button  is pressed or to enter it into the *.poly file (remember the *.poly format for TetGen) and, finally, we would need a *.control file similar to the one used in 2D example. In all other respects, the simulation procedure is identical.

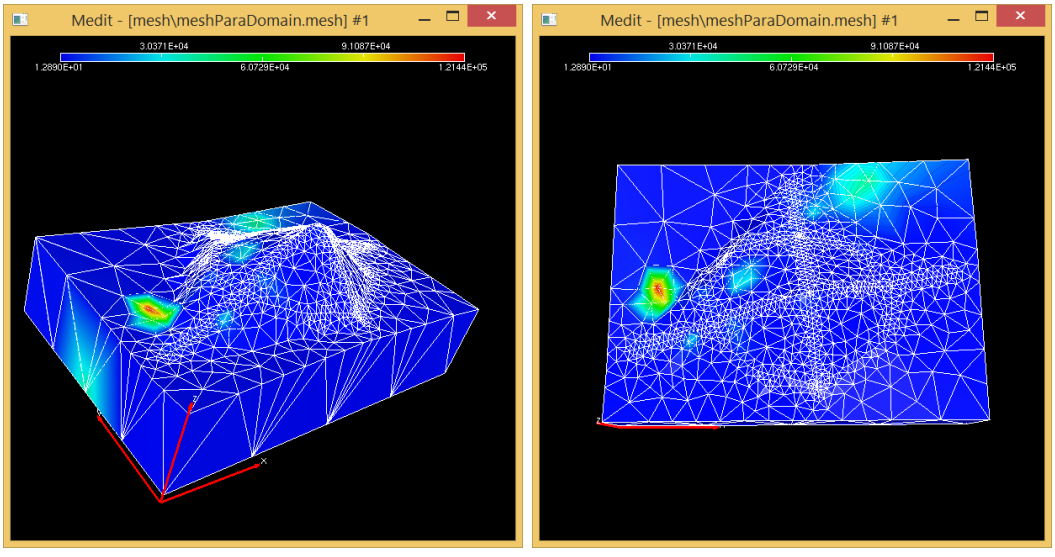
An example of solving a 3D inversion problem with topography in the source and additional files.

This example taken from the set of examples of the software https://gitlab.com/resistivity-net/bert, while minor changes made to the input data file for better compatibility with all features of the DiInSo program. The input file is in the "examples" folder of the DiInSo program and it name is test_3d_topo.bert. This file is intend for 3D inversion, the electrodes in the file are located at different heights, so we will deal with the topographic effect. First, set the parameters of the 3D inversion. Let "smooth?" — 1 (we need a smoothed mesh on the surface of the modeling area), "quality" — 30, "mesh grow" — 2 (let's not set the mesh too detailed to save PC memory), "recalc jacobian?" — 1 (we will recalculate Jacobian at each iteration step), "topography" — 1 (we will take into account the topographic effect). It is IMPORTANT to REMEMBER that taking into account the topographic effect in the three-dimensional case is a much more complicated task than in the two-dimensional case; therefore in the presence of a complex modeling surface the user MUST SET the option "topography" — 1, otherwise the mesh may be inconsistent with the positions of the electrodes and the DiInSo program will close abnormally. Next, select the input file using the "DATA FILE:" button. The number of electrical tomography arrays positions for this problem is quite large — 2424, so the inversion process may take a bit more time. Now we are ready to begin the inversion. Let's press the button "INVERSION", the solution will be obtain in 7 iterations with not very good convergence (our mesh is too coarse). In this example, we will not pay much attention to the inversion results

inside the modeling area, especially since we have a rather coarse mesh, but look at the results on the surface of the modeling area and on the surface itself, which has a topographical effect. Open the file with the inversion results test_3d_topo.vtk in the VisIt program and carefully look at the uneven surface (Add -> Mesh -> mesh, Add -> Pseudocolor -> Resisitivity_lp_log10_rp_, Draw):



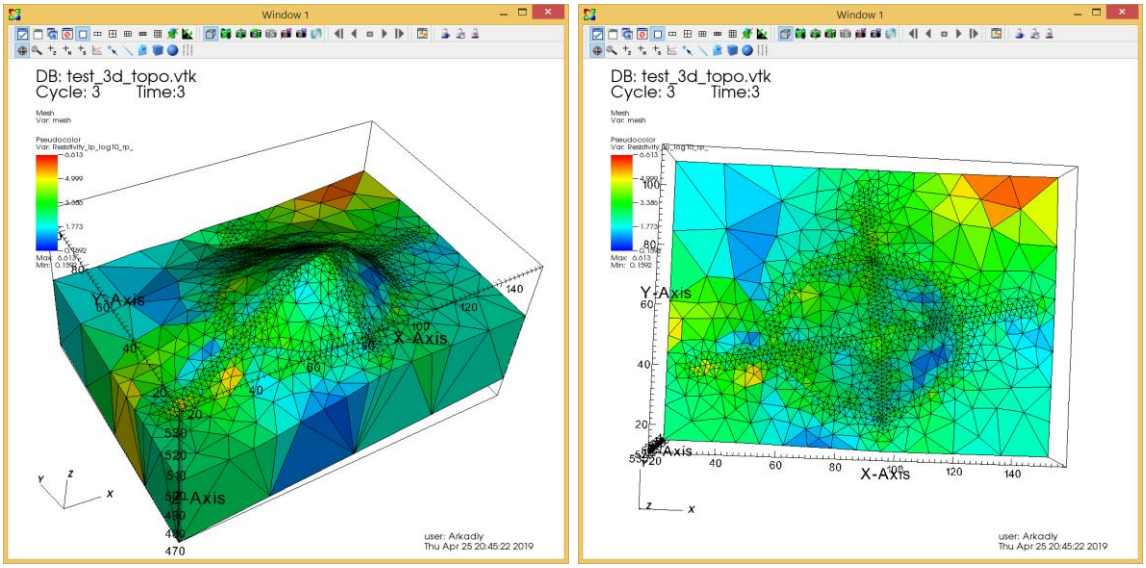
Or you can look at the results using the built-in visualizer by clicking the “vis. inv.” button (although in this case the concept of the surface will be slightly worse due to a different implementation of shading in the embedded visualizer), and then, for example, press “m” (data), “o” (contour), “A” (axes coordinates) and “c” (“add surface color”) keyboard keys:



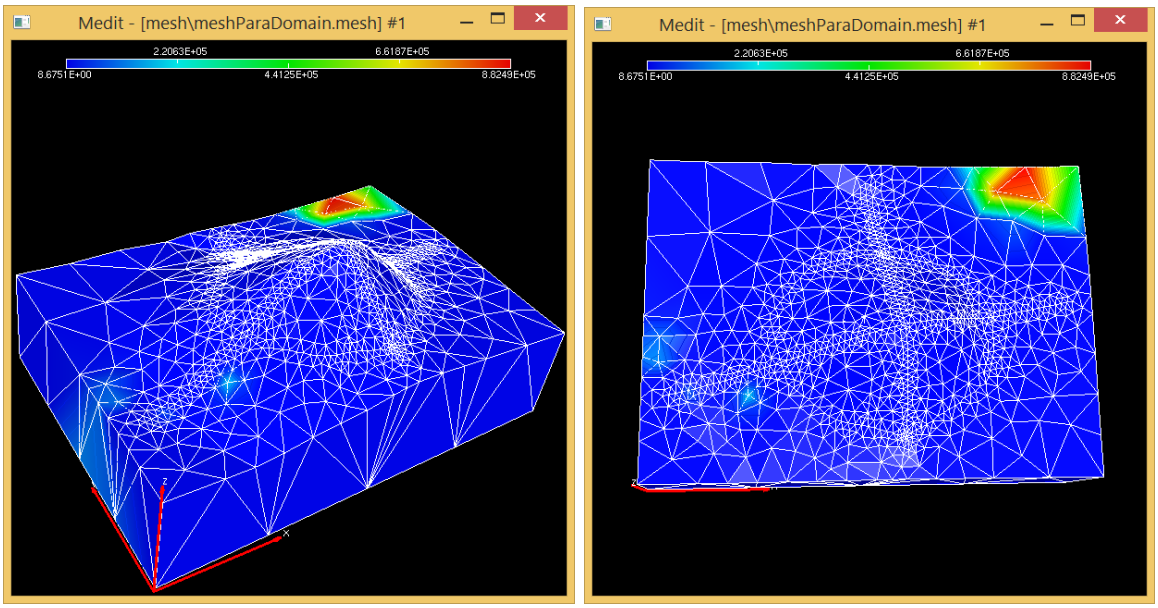
The apparent difference in the distribution of resistances when opening inversion results in the VisIt program and in the embedded visualizer is because in VisIt we opened a file where results are associated with mesh cells (tetrahedra), and the built-in visualizer shows results associated with nodes of the mesh. Recall

that the first presentation is more natural and “correct” from the point of view of the inversion algorithm, and the second one is view that is more pictorial. The finer the mesh — the less different in the representations associated with cells (tetrahedra) and nodes. By the way, if we opened the test_3d_topoP.vtk file in VisIt program, we would see the same pattern of resistance distribution as in the built-in visualizer.

On the surface, we can observe quite sharp transitions in height and even acute angles, which are unusual for real geological surfaces. They are especially noticeable during approaching the “hill” and the lines of electrodes located crosswise. Obviously, errors when specifying a complex surface can lead to errors in the results of inversion. That is why it is necessary to “refine” such surfaces. We can do this with the help of additional points specified in a separate file. In our case, such a file is in the “examples” folder of the DiInSo program and it name is test_3d_topo.xyz. We leave all previous settings unchanged, but we add the test_3d_topo.xyz file, selecting it using the “topo/line file:” button and activating checkbox to the left of this button. Let’s solve an inversion problem by pressing the “INVERSION” button. This time the solution is obtained in 8 iterations, the convergence is still not very good (but better than in the previous case), since the internal mesh has not been improved. Open the newly received file test_3d_topo.vtk with the results and, like last time, pay attention only to the surface of the modeling area:



Or you can look at the results using the built-in visualizer by clicking the “vis. inv.” button (although in this case the concept of the surface will be slightly worse due to a different implementation of shading in the embedded visualizer), and then, for example, press “m” (data), “o” (contour), “A” (axes coordinates) and “c” (“add surface color”) keyboard keys:



About the apparent difference in the distribution of resistances when opening the inversion results in the VisIt program and in the embedded visualizer, we have already explained above.

We can see that the surface has become more natural, sharp transitions and sharp corners minimized. If desired, it would be possible to more accurate define the surface by adding additional points. The rule here is very simple: more supporting points — better model. However, overdoing here is also not necessary, since our goal is not to model the ideal surface, but to eliminate gross flaws that can affect the results of the inversion. By the way, about the results. We see that after the refinement of the surface, results changed somewhat, although we did not change other inversion settings. This clearly demonstrates that the complex surface in model requires special attention from the researcher. It is not enough just to conduct an electrical tomographic measurements and remembering the points where the electrodes are located. We also need additional supporting points for accurate modeling of the surface where the measurements taken.

An example of solving a 3D inversion problem using the data obtained for another program for solving inversion problems (Res2DInv / Res3DInv).

In order to use the input data for the following example, we need the installed version of the “Res2Dinv” program. To do this, you can download the demo version of Res2DInv x32 or Res2DInv x64 from the official website: https://www.geotomosoft.com/downloads.php

After installing Res2DInv, its example files are located in the same folder as the program itself. Our goal is to take one of the sample files *.dat, convert it to the *.bert format and solve the inversion problem. Recall that the DiInSo converter works only with the “general array data” format of the Res2DInv / Res3DInv

programs and full support is not guaranteed, therefore any *.dat file from the folder will not work for us. You can learn more about the “general array data” format of Res2DInv / Res3DInv programs from the official documentation that is install with the programs. Let's look at an example with the name MIXED.DAT. Simple conversion from Res2DInv to *.bert for 2D inversion and from Res3DInv to *.bert for 3D inversion is nothing complicated and interesting (do it yourself), so we’ll consider converting from several Res2DInv files to one *.bert file for 3D inversion. One “MIXED.DAT” file will act as “several” files, copies of which are not necessary to create. We recall that for such conversion you need to create in any text editor a file in the following format which was describe above:

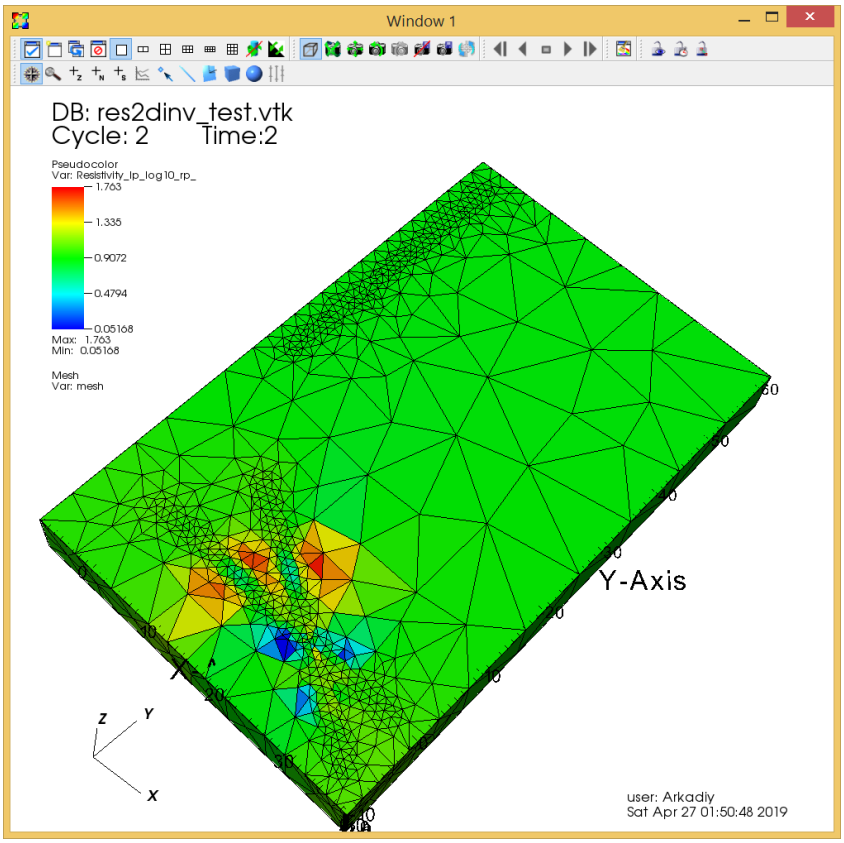
```
<Res2DInv file name with FIRST profile>
<X displacement of NEXT profile> <Y displacement of NEXT profile relative to the FIRST profile (i.e., relative to zero)> <rotation angle (in degrees) of NEXT profile clockwise>
<Res2DInv file name with NEXT profile>
<X displacement of NEXT profile> <Y displacement of NEXT profile relative to the FIRST profile (i.e., relative to zero)> <rotation angle (in degrees) of NEXT profile clockwise>
<Res2DInv file name with NEXT profile>
...
```

Create a file with any name, say, res2dinv_test.txt and write to it, for example, the following lines:

```
MIXED.DAT
5 10 30
MIXED.DAT
-5 25 270
MIXED.DAT
```

Thus, in the resulting *.bert file we want to see three identical profiles from the MIXED.DAT file, the second of which is displaced relative to the first one by 5 meters in X, by 10 meters in Y and rotated 30 degrees clockwise around the first point of the profile; the third profile is displaced relative to the first one by –5 meters in X, by 25 meters in Y (or by 15 meters in relation to second profile) and rotated 270 degrees clockwise around the first point of the profile. Save the file in the same directory as the file MIXED.DAT and open it with the DiInSo converter, called by the button  . Conversion type choose as “List of Res2DInv's to BERT 3D”. Press the “Convert” button and get a *.bert file-result for 3D inversion with the name res2dinv_test.bert. In this file, the number of electrical tomography arrays positions is 1221, which is three times more than in the MIXED.DAT file and this is logical. Select the res2dinv_test.bert file by clicking on the “DATA FILE:” button. First, set the parameters of the 3D inversion. Let “quality” — 34, “mesh grow” — 1.5 (we will set the average detail mesh), “recalc jacobian?” — 1 (we will recalculate Jacobian at each iteration step). Now we are ready to begin the inversion. Press the button “INVERSION”, the solution will be obtain in 4 iterations with very good convergence. In this example, we will not pay much attention to the inversion results inside the modeling area, especially because our test is fictional, but look at the results on the surface of the modeling area and on

the surface itself, where the profiles placed by us in a rather unusual way. Open the file res2dinv_test.vtk with the results of inversion in VisIt program and carefully look at the surface (Add -> Mesh -> mesh, Add -> Pseudocolor -> Resisitivity_lp_log10_rp_, Draw):



On the surface clearly stand out three profiles, two of which are crossed in the form of the letter “X”, and the third is strongly moved away from the rest (since it was rotated 270 degrees clockwise). Of course, in real-life situations, such an arrangement of profiles is quite rare, but the goal of this example was to show that we could assemble an arbitrarily complex configuration for 3D inversion from a set of 2D profiles.

Many more examples of different complexity can be contrive, but we will stop here. If you have an interesting example and you would like to see it in the updated help file, send it to my email address with a detailed description: marinenkoarkady@yandex.ru

That's all! Enjoy using the program!