PNNL-SA-23783

E4D: A distributed memory parallel electrical geophysical modeling and inversion code

User Guide - Version 1.0

September 2014

TC Johnson

U.S. DEPARTMENT OF
ENERGY

Prepared for the U.S. Department of Energy
under Contract **DE-AC05-76RL01830**

PNNL-SA-23783

E4D: A distributed memory parallel electrical geophysical modeling and inversion code

TC Johnson

September 2014

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99352

Acknowledgments

E4D development was jointly funded by the U.S. Department of Energy - Office of Environmental Management and Office of Science, and the U.S. Department of Defense under the Environmental Security Technology and Certification Progam. Development of the infrastructure modeling and inversion modeling capabilities was funded by the Pacific Northwest National Laboratory, with support for testing provided by the U.S. Department of Energy - Office of River Protection.

Contents

Figures

1.0 Overview

E4D is a three-dimensional (3D) modeling and inversion code designed for subsurface imaging and monitoring using static and time-lapse 3D electrical resistivity (ER) or spectral induced polarization (SIP) data. To address the computational demands of inverting large scale 3D and 4D data sets, E4D was designed specifically to run on distributed memory parallel high performance computing systems. However, E4D will run on any system with at least two processing cores.

E4D executes by reading a number of user-created ascii text input files, executing a particular run mode (e.g., mesh generation, ER forward simulation, ER inversion, SIP inversion), and reporting results. Run-time options made available through input files are designed to enable flexibility and a high level of customization for a particular problem, making E4D suitable for both advanced research applications, as well as more common applications.

A critical component of E4D is visualization. The E4D distribution includes the utility program bx, which places E4D simulation results into an EXODUS II finite element data model formatted file. EXODUS II files are then visualized using the VisIt visualization program, an open source visualization code developed by the Lawrence Livermore National Laboratory. However, all of the files necessary for users to develop and operate other visualization codes are provided by E4D. Guidance concerning the use of VisIt is provided on the VisIt website, and is not included in this user guide.

The intended audience for this user guide includes researchers and other practitioners familiar with electrical geophysical methods and deterministic geophysical inversion. Guidance is given herein concerning how to enable and use the capabilities provided by E4D, with specific examples illustrated in a series of tutorials at the end of each chapter. Justifications for using a particular approach or capability are left to the user.

2.0 General Capability Description

E4D executes according to one of several run modes chosen by the user. Run modes can be classified generally as mesh generation modes, forward run modes, static inversion modes, and time-lapse inversion modes. Required input files, and the formats of those files, vary somewhat depending on the run mode. This user guide is divided into chapters that describe input file and other requirements specific to each run mode, including tutorials designed to illustrate the general use for each mode. The run modes are as follows:

Run Mode	Function	Description
1	ER Mesh Generation Mode	Mesh generation for electrical resistivity modeling and inversion
2	ER Forward Run Mode	Direct current potential field and electrical resistivity survey simulation
3	ER Static Inversion Mode	Electrical resistivity inversion for a single survey
4	ER Time-lapse Inversion Mode	Electrical resistivity inversion for multiple time-lapse surveys
21*	SIP Mesh Generation Mode	Mesh generation for spectal induce polarization modeling and inversion
22*	SIP Forward Run Mode	Complex potential field and spectral induced polarization survey simulation
23*	SIP Static Inversion Mode	Spectral induced polarization inversion for a single survey
31*	ER Tank Mesh Generation Mode	Mesh generation for electrical resistivity modeling and inversion within a tank
32*	ER Tank Forward Run Mode	Direct current potential field and electrical resistivity survey simulations for tank scale imaging
33*	ER Tank Static Inversion Mode	Electrical resistivity inversion for a single survey within a tank
34*	ER Tank Time-lapse Inversion Model	Electrical resistivity inversion for multiple time-lapse surveys within a tank
41*	SIP Tank Mesh Generation Mode	Mesh generation for spectral induced polarization modeling and inversion within a tank
42*	SIP Tank Forward Run Mode	Complex potential field and spectral induced polarization survey simulations for tank scale imaging
43*	SIP Tank Static Inversion Mode	Spectral induced polarization inversion for a single survey within a tank
*	Not provided in the open source version of E4D	

E4D provides conditional solution constraint options using the method of Iteratively Reweighted Least Squares (IRLS). Model constraint options are intended to be flexible, providing users numerous options for incorporating a priori information into the inverse problem, thereby improving imaging resolution (Johnson, submitted). In addition, E4D executes on an unstructured tetrahedral mesh, enabling users to accurately incorporate complex subsurface structures with known dimension and location. Each of the forward run and inversion modes has the optional capability of modeling the effects of metallic inclusions with arbitrary shape and size, including non-point electrodes, and accounting for (and removing) those effects in the inversion. The module providing this capability is the Infrastructure Modeling and Inversion (IMI) module, and is not included as part of the open source distribution. Information concerning the availability of E4D capabilities that are not provided in the open source release version can be found on the E4D website at https://e4d.pnnl.gov .

3.0 Installation

Most distributed memory high performance computing systems operate on Unix/Linux-based operating systems. Because such systems vary in configuration, users will need to compile E4D for each individual system. E4D is provided with an installation script designed to automatically download missing software libraries and compile all of the components necessary to execute E4D in parallel. This includes the E4D executable *e4d*, the Message Passing Interface executable *mpirun*, the open source mesh generation programs triangle and tetgen, and the utility program bx. Because E4D downloads missing libraries, an internet connection is required for compilation. VisIt binaries are available for several operating systems, and must be downloaded separately. E4D requires the fortran90/95 compiler gfortran, which is freely available for Unix/Linux operating systems. Other fortran90/95 compilers may be used by modifying the make files provided with the E4D distribution. Also, E4D requires compilation on a 64-bit architecture.

3.1 Linux Installation

1. Download the E4D tarball from https://e4d.pnnl.gov
2. Extract E4D and third party codes:

   tar -xzvf e4d_qc.tgz

   This will unpack the directory **e4d_qc**.
3. Goto the directory **e4d_qc/src**.
4. Execute the build script **build.bsh** script with:

   ./build.bsh

The script build.bsh will compile and install the third party libraries necessary to build E4D (except *gfortran)* and E4D utility programs including triangle, tetgen, petsc, netcdf, and exodus. Executables necessary to run E4D are copied to e4d_qc/bin, including:

- **triangle**: triangular mesh generation software used by E4D to build the surface boundary of the computational mesh,
- **tetgen**: tetrahedral mesh generation software used by E4D to build the unstructured computation mesh,
- **bx**: E4D utility file used to insert simulation and inversion results into an exodus file for visualization,
- **mpirun**: script used to call E4D with an allocation of processors, and
- **e4d**: main E4D program.

3.2 Windows Compilation

Compiling E4D on Windows requires the cygwin package, and is somewhat more involved than compiling on a linux/unix operating system. Cygwin is a collection of tools and libraries that enable linux functionality on a windows operating system. E4D is provided with a set of bash scripts and make files that are used to compile E4D with cygwin in the same manner as the linux compilation described above.

To install cygwin, download the appropriate cygwin install script from https://cygwin.com/install.html (note the 64-bit version is required for E4D). Review the installation instructions and execute the install script. The script will initiate a graphical user interface (gui) that provides options for the installation directory, the website from which to download packages and libraries, and a list of optional packages to install. For the installation directory, choose a directory without spaces in the name, (i.e., C:\cygwin). E4D requires (at a minimum) the following packages to be installed with cygwin:

- gcc-core
- gcc-fortran
- gcc-g++
- make - the GNU version of the make utility
- diffutils
- netcdf-fortran
- netcdr-fortran-devel
- python

Each of these packages can be located by entering the package names given above in the search dialog box of the package selection section of the installation gui. After choosing the packages, the installation script indicates a list of dependencies necessary for the selected list. Simply click “next” to accept the dependencies and begin the installation. When the cygwin installation completes, the install script will ask whether a cygwin icon should be place on the desktop or start menu. Choose one or both of these to simplify initiating the cygwin terminal window.

- Once cygwin is installed, download the E4D tarball from https://e4d.pnnl.gov into your cygwin home directory. For example, if cygwin was installed in C:/cygwin, your home directory is located in C:\cygwin\home\<your_user_name>.
- Open a cygwin terminal by double-clicking the icon cygwin created during installation (either on your start menu or desktop or both). If you are unfamiliar with the linux terminal window, it is similar to the dos command prompt. You should automatically be in your home directory within the terminal when it is opened.
- Decompress the E4D tarball by typing "tar -xvzf e4d_qc.tgz"
- Within the terminal, go to the directory <e4d_dir>/eq4_qc/src, where <e4d_dir> is the directory created from which the tarball was decompressed in the previous step.
- The cygwin build script is located in <e4d_dir>/e4d_qc/src/build_cygwin.bsh. To execute the build script, type "./build_cygwin.bsh" from on the cygwin command line.

At this point the build script will download and compile the required third party libraries, utility routines, and the e4d executable, and place them in <e4d_dir>/e4d_qc/bin.

Add the following lines to the file .bashrc, which is located in your home directory.

```
alias ls='ls -l -hF --color=tty'
alias mpirun="<e4d_dir>/e4d_qc/third_party/petsc-3.4.3/arch-mswin-c-opt/bin/mpirun"
export PATH=$PATH:<e4d_dir>/e4d_qc/ bin
```

Now type the command "source ~/.bashrc" from the cygwin terminal. Each of the programs necessary to execute e4d should now be visible from the command line, and will be visible with each new initiation of a cygwin terminal.

Note that all of the files necessary to execute e4d may be created using windows tools and file managers. However, all e4d programs described in the forthcoming documentation must be executed as described from the cygwin terminal command line. Visualization may then be conducted with the windows version of VisIt, downloadable from the VisIt website.

Note that some windows text editing programs leave hidden characters that corrupt ascii files as far as cygwin is concerned. It is important to use a linux compatible text editor when creating e4d input files. TextPad is one example of a compatible text editor.

3.3 Custom Compilation

Many Linux-based high performance computing systems provide customized libraries, conveniently loaded and made available as modules. Because the compilation process on such systems varies, no specific instruction for compilation is given herein. However, the commands found in the build script (*build.bsh*) and corresponding make files provide information concerning which compilers and libraries are needed for each program.

4.0 Running e4d

4.1 Standard Execution

E4D is executed from the command line by typing:

<e4d_dir>/e4d_qc/bin/mpirun -np <num_proc> <e4d_dir>/e4d_qc/bin/e4d

where <*e4d_dir*> is the directory containing E4D (i.e. e4d_qc), and *num_proc* is the number of processors *e4d* will use during execution. Alternatively, *<e4d_dir>/e4d_qc/bin* can be placed in the executable path so that E4D is called by:

mpirun -np <num_proc> e4d

Note that some systems install a version of MPI by default, and set system variables so that the *mpirun* command defaults to the version installed by the system. It is important to ensure that E4D is called using *<e4d_dir>/e4d_qc/bin/mpirun* so that the correct run-time libraries are used.

Upon execution, E4D will read the file *e4d.inp* to obtain the names of other input files and run instructions chosen by the user, all of which are described in the forthcoming documentation for each run mode.

4.2 Batch Submission

Most high performance computing systems use batch submission software to queue and prioritize parallel computation jobs. Job submission scripts and associated commands are system independent. Users should refer to system documentation and/or administrative help to run *e4d* in batch submission mode.

5.0 Mode 1: ER Mesh Generation Mode

5.1 Introduction

Mesh generation refers to the process E4D uses to generate the unstructured tetrahedral computational mesh used in forward and inverse simulations. Depending on user input, E4D can create a mesh that conforms to known subsurface boundaries such as borehole boundaries, water table boundaries, or the boundaries of buried tanks and other structural features. E4D can also create distinct zones within the mesh, which are the primary units whereby model constraints are defined to regularize or otherwise inform the inverse problem. Known boundaries incorporated into the mesh facilitate accurate ER and SIP data simulation in forward modes, and inform the inverse operation in inverse modes.

To create the mesh, E4D:

1. reads mesh construction options from a user supplied mesh configuration file,
2. builds an input file for the third party open source program triangle,
3. calls triangle to construct a high quality triangular mesh that will be used to define the surface boundary of the tetrahedral mesh,
4. loads the surface mesh constructed by triangle,
5. builds an input file for the open source tetrahedral mesh generator tetgen,
6. calls tetgen to build the tetrahedral mesh, and
7. if directed in the mesh configuration file, calls bx to build the exodus file used for mesh visualization.

5.2 Run Configuration File: e4d.inp

Upon execution, e4d reads the instructions provided in the run configuration file *e4d.inp* in order to determine which mode to run and which input files should be used to execute that mode. In ER mesh generation mode (mode 1), *e4d.inp* requires only two lines of instruction as described below, where each line in the table represents a line in the text file, as follows:

Input	Description
1	Run Mode
mesh_configuration_filename	Name of the mesh configuration file

The mesh configuration file (described below) name must end with the suffix *.cfg*, which E4D uses to delineate between mesh configuration files and other mesh files as described below. For example, if the name of the mesh configuration file is *test_mesh.cfg*. then *e4d.inp* would have the following two lines of text:

<begin e4d.inp> this line is not included in the file

1
test_mesh.cfg

<end e4d.inp> *this line is not included in the file*

Note that all E4D input files, including *e4d.inp*, allow comments to be placed after required inputs on a given line, and/or after all of the required lines of inputs have been entered. For example, the file given above can be annotated as follows (see Appendix A for formatting rules):

<begin e4d.inp> this line is not included in the file

1 text is allowed here ... this line specifies the run mode
test_mesh.cfg text is allowed here ... this line specifies the name of the mesh configuration file

text is also allowed here since there are no further lines of input required in mode 1.

<end e4d.inp> *this line is not included in the file*

5.3 Mesh Files

As described above, E4D executes mesh generation when mode = 1 in *e4d.inp*. Upon execution, E4D reads the mesh configuration file and attempts cursory checks for formatting errors. To aid users in identifying mesh configuration file errors, E4D reports values read from the mesh configuration file to the file *mesh_build.log*. If there are errors in the mesh configuration file that cause triangle or tetgen to fail, E4D will print an error message and exit. In this case, users should check *mesh_build.log* for help in identifying the source of the error. If the mesh build executes successfully, five files are produced that describe the the mesh, each of which are used in forward run or inverse modes of E4D. Assuming the mesh configuration file is named *<e4d_mesh>.cfg,* where *<e4d_mesh>* is the user chosen mesh configuration file name, the following five files are produced:

File Name	Produced By	Description
<e4d_mesh>.1.node	tetgen	Provides locations of nodes and boundary markers for each node in the mesh. (see tetgen documentation)
<e4d_mesh>.1.ele	tetgen	Describes how nodes are connected to form elements, and assigns each element a zone number. (see tetgen documentation)
<e4d_mesh>.1.neigh	tetgen	Provides a list of neighboring elements for each element in the mesh. (see tetgen documentation)
<e4d_mesh>.1.face	tetgen	Provides a list of faces that constitute boundaries defined in the mesh configuration file. (see tetgen documentation)
<e4d_mesh>.trn	E4D	Provides the horizontal (x and y) and vertical (z) coordinate translations used internally by E4D to optimize numerical precision.
For reference, E4D calls tetgen with the following command line options: tetgen -pnq<*m_qual*>a<*max_evol*>aAA <e4d_mesh>.poly where *m_qual* and *max_evol* are defined below, and <e4d_mesh>.poly is the tetgen input file created by E4D using options specified in the mesh configuration file.		

Although it is not necessary to understand the format of the mesh files for standard applications in E4D, understanding the mesh file formats facilitates more advanced uses of E4D.

5.4 Elements and Zones

E4D closely follows the conventions used by tetgen to build the mesh. Namely, the mesh boundaries (both external and internal) are described by a set of planes (called piecewise linear complexes in the tetgen documentation) that define one or more distinct 'watertight' 3D volumes, which are called zones in E4D. Zones are used, for example, to define known subsurface structures such as wellbores or geologic facies, or to define a specific volume for visualization. Each tetrahedral element in the mesh is assigned a unique zone, and the elements comprising a zone are contiguous. Zone assignments are specified in the mesh configuration file.

5.5 Nodes and Boundaries

Each element is defined by a set of four nodes connected to form a tetrahedron. Node placement and element generation is done automatically by tetgen, with guidance from E4D through the mesh configuration file. For example, the mesh configuration file must specify that a node be placed at each point electrode location. Nodes are used to define mesh boundaries and form zones, and can also be used to refine the mesh in a specific location (i.e., around electrodes). Each specified node location within the mesh configuration file is called a control point in E4D. Connections between control points are used to form boundaries, which are used to define zones, as specified in the mesh configuration file.

5.6 Mesh Configuration File

The purpose of the mesh configuration file is to provide E4D with instructions concerning how to build the mesh. The mesh configuration file is comprised of five blocks, each of which are described below. As with all E4D input files, blank lines are allowed. Comments are also allowed on each line where an input value is specified, at any position on the line after the input value (see Appendix A for formatting rules). As described above, the mesh configuration file must be named *<ert_mesh>.cfg*, where *<ert_mesh>* is a user chosen mesh file name.

5.6.1 General Block

The general block provides instructions concerning mesh quality, lower boundary elevation, and the locations of the tetgen and triangle. The format of the general block is provided in the table below, with each row of the table representing a corresponding row in the mesh generation file. The only record required on each line is the specified variable (i.e., the first column), and comments may be added after each variable.

Variable	Type	Description
m_qual	real number	The maximum radius-to-edge ratio of any element in the mesh (see tetgen documentation). Recommended values are in the range of 1.3 to 1.5, with 1.3 specifying a higher quality mesh
max_evol_def	real number	Default maximum volume of all elements in the mesh. Maximum volumes for each zone are specified using *mz_vol* (see zone configuration block below)
m_bot	real number	Elevation of the bottom of the computational mesh.
tet_build_flag	integer (0 or 1)	If 0 is specified, E4D will build the .poly input file for tetgen, but will not call tetgen to build the mesh. If 1 is specified, E4D will call tetgen to build the mesh
tet_loc	String	This variable specifies the location of the tetgen executable, and must be enclosed in single quotes. For example, if tetgen is located in */usr/e4d_qc/bin/tetgen*, then *tet_loc* is '*/usr/e4d_qc/bin/tetgen*'. If */usr/e4d_qc/bin* is in the executable path, then 'tetgen' will suffice
tri_loc	string	The variable specifies the location of the triangle executable, and be enclosed in single quotes. For example, if triangle is located in */usr/e4d_qc/bin/triangle*, then *tri_loc* is '*/usr/e4d_qc/bin/triangle/*'. If */usr/e4d_qc/bin* is in the executable path, then 'triangle' will suffice

5.6.2 Control Points Block

Those points that are required to define mesh geometry, including surface topography, electrode locations, internal boundaries (e.g., non-point electrodes, buried infrastructure, known geologic contacts), and mesh refinement points are specified as control points within the control points block. Each control point will become a mesh node in the final tetrahedral mesh. There are three types of control points:

1. Surface points are used to specify points of known elevation and/or surface electrode locations on the mesh, and are given a flag of 1. E4D, triangle, and tetgen use these points to define surface topography and add nodes at surface electrode locations. Surface points cannot be placed on or beyond the outer boundary of the mesh (see Figure 5.1), which is specified by a set of surface boundary points, as described below.

2. Surface boundary points are used to define the outer boundary of the computational mesh, and are given a flag of 2 (see Figure 5.1). E4D uses these nodes to construct the outer vertical boundaries of the computational mesh by connecting adjacent points and constructing a series of vertical planes down to the elevation *m_bot*, which is specified in the general block.

3. Internal control points are used to define internal boundaries, buried electrode locations, and mesh refinement points, and are given a flag of 0 unless they are used to define a metallic boundary. Internal control points that are used to define a metallic boundary are assigned a negative integer, with each unique metallic boundary (point, line, plane, or volume) having a unique negative flag. All internal control points must be placed within the mesh boundaries formed by the surface boundary, vertical external boundaries, and bottom boundary (Figure 5.1).

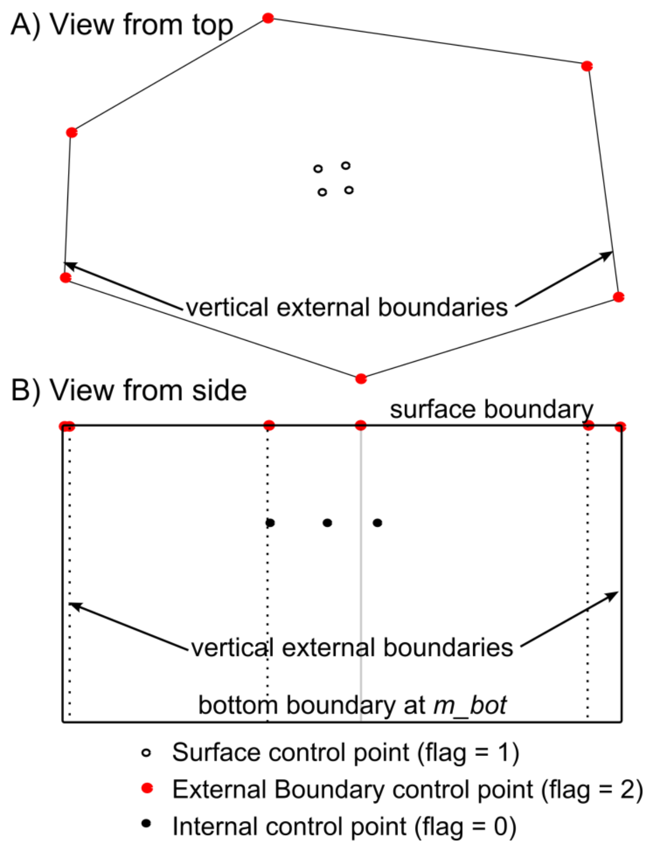

A) View from top
vertical external boundaries
B) View from side
surface boundary
vertical external boundaries
bottom boundary at *m_bot*
Surface control point (flag = 1)
External Boundary control point (flag = 2)
Internal control point (flag = 0)

Figure 5.1. Diagram of the three types of control points used to define a mesh

It is best practice to place a control point at every electrode location, although it is not explicitly required by E4D. E4D 'snaps' each electrode to the nearest node in the mesh. If the nearest node is far from the actual electrode location, modeling errors may result (E4D will print a warning message in modes 2 and higher if this occurs).

The format of the control point block is as follows:

Variable	Type	Description
n_cpts	positive integer	specifies the total number of control points
cp_num x y z b_flag	see description	*cp_num* is a positive integer value that gives the index of the control points specified on this line. The *cp_num* values should be numbered consecutively from 1 to *n_cpts*. *cp_num* can be used to identify a particular control point used in a boundary definition as described in the internal boundary control block below. *x y* and *z* are real valued numbers that give the easting, northing and elevation positions of control point *cp_num* respectively. *b_flag* is an integer value that provides the boundary flag for the control point specified on this line as described above. **There will be *n_cpts* of these lines in the control points block, one for each control point.**

It is generally beneficial to refine the mesh near electrodes in order to improve modeling accuracy. For buried electrodes, this can be done by placing a second control point near each electrode control point. By so doing, tetgen will refine around the two points, as is required, to maintain a high quality mesh. The mesh can be refined around surface electrodes in the same manner. Another approach to refine around surface electrodes is to place each just under the surface with a boundary flag of *b_flag* = 0. By doing so, tetgen will automatically refine around each surface electrode. This practice also facilitates high quality surface mesh generation by removing the requirement to have a node at every electrode location on the surface (i.e., the electrodes are placed just beneath the surface).

5.6.3 Internal Boundary Configuration Block

Internal boundaries consist of one or more user-defined internal planes (called piecewise linear complexes in the tetgen documentation) that separate one region of the mesh from another. Internal boundaries that close to form a 'watertight' structure form a zone within the mesh. Boundaries are specified by describing how selected control points are connected to form planes within the internal boundary configuration block as follows:

Variable	Type	Description
n_plc	non-negative integer	total number of user-defined piecewise linear complex's (i.e. planes)
np b_num	see description	*np* is the number of control points used to define this plane (there will be one *np* value listed for each of the *n_plc* planes defined) *b_num* is the user specified boundary number for this plane. This integer can be any value except 0, 1, or 2, which are reserved within E4D. Set *b_num >2* unless this boundary is an infinite conductivity (i.e. metallic) boundary, in which case *b_num* should be set to the same negative integer as *b_flag,* where *b_flag* is the boundary flag specified for each of the *np* control points used to defined this plane, as listed on the next line. All boundaries sharing the same negative *b_num* value are modeled as infinite conductivity and electrically connected structures within E4D. **This line is the beginning of a plane definition. There will be n_plc of lines in the internal boundary configuration block.**

Variable	Type	Description
p1 p2 ... p_np	non-negative integer	p* is the *th control point defining this plane. This value references the control point indexes specified in the control points block. Control points must be specified in order to define a plane such that no segment connecting two consecutive control points crosses any other segment connecting two consecutive control points (i.e list the control points either clockwise or counter-clockwise). *E4D does not check this condition prior to executing tetgen, and tetgen will fail if this condition is violated.* Also, each of the points specified must lie in the same plane according to numerical precision or tetgen will fail. ***This line is the end of a plane definition. There will be n_plc of lines in the internal boundary configuration block.***

5.6.3.1 Critical boundary definition rules

The most common source of error in mesh generation is the improper specification of internal boundaries. If tetgen fails and the source of the error is not obvious from the *mesh.log* file, make sure the boundary definitions in the boundary configuration block adhere to the following requirements.

1) Do not define planes on any external boundary (e.g., surface, bottom, or side boundaries). E4D will construct these automatically. It is appropriate and required to explicitly specify segments of subsurface planes that intersect the surface boundary within the boundary definition (see item 5 below).

2) Ensure that boundaries do not intersect, except along common segments. Such intersections will cause tetgen to fail, and E4D does not check for this condition.

3) All control points existing within a defined plane, or along any segment of of a defined plane, must be included in the definition of that plane.

4) Each of the control points used in a plane definition must lie upon the same plane. This is important if four or more points are used to define a plane.

5) There are several import considerations when constructing boundaries that intersect the surface.

   - The segments of any plane that intersect the surface boundary must be explicitly specified in the plane definition. In other words, any plane segment that intersects the surface boundary must not be defined by the first and last points specified in the plane definition (i.e. the points defining segments on the surface must be listed sequentially in the plane definition).

   - It is often useful to specify a series of planes that form a closed boundary (thereby forming a zone) whose upper margin is the surface of the mesh, which is automatically generated by E4D (see item 1 above). In this case E4D must correctly connect the segments of planes intersecting the surface such that those segments define the upper boundary of the zone. To do so, E4D takes the list of control points defining the segments that intersect the surface, and connects each point to the next nearest point. With this in mind, it is important to define the planes that intersect the surface in such a manner that E4D correctly connects the surface segments. Attention to this issue is only required for boundaries whose surface segments form convex angles.

It is generally helpful and highly recommend to draw a picture with relevant control points labelled to ensure that boundary definitions do not violate these conditions.

5.6.4 Hole Configuration Block

Any zone defined within E4D can be turned into a hole or void in the mesh. To create a hole, tetgen begins at one point within the subject zone and successively removes all of the elements enclosed within the zone boundary. The hole configuration block specifies which zones should be hollowed to form a hole by specifying a point within the zone as follows:

Variable	Type	Description
n_holes	non-negative integer	specifies the number of holes in the mesh
hn xh yh zh	see description	*hn* is the index for this hole. *xh yh* and *zh* are real valued numbers indicating the easting, northing, and elevation of any point within this hole **There are *n_holes* of these lines, one for each hole, and *hn* should range from 1 to *n_holes*.**

Three-dimensional metallic structures should be defined as holes in E4D, so that only the shell of the structure exists in the mesh. This is done by specifying a hole point inside of a zone constructed using planes with the same negative boundary flag as described above.

5.6.5 Zone Configuration Block

The zone configuration block assigns maximum element volumes, conductivity values, and zone numbers to each zone.

The format of the zone configuration block is as follows:

Variable	Type	Description
n_zones	positive integer	number of user-defined zones
zn xz yz zz mz_vol zcond	see description	*zn* is the zone number assigned to this zone. *xz yz zz* is the position of any point within this zone *mz_vol* is the maximum volume of any point within this zone *zcond* is the conductivity of this zone **There are *n_zones* of these lines, one for each zone in the mesh. These lines should be listed consecutivelyfrom *zn* = 1 to *zn* = *n_zones*.**

Zones that are not configured will assume a maximum volume constraint of *max_evol_def* (see general block) and a default conductivity value. However, it is best practice to provide configuration options for each zone defined in the mesh.

5.6.6 Visualization Options Block

The visualization options block provides the option to automatically produce an exodus file for visualization once the mesh is generated as follows:

Variable	Type	Description
bx_flag	integer value (0 or 1)	specifies whether E4D should build an exodus for mesh visualization (1) or not (0)
bx_loc	string	specifies the location of the utility program bx. For example, if bx is located in /usr/e4d_qc/bin/, then *bx_loc* is '/usr/e4d_qc/bin/bx' (quotes are required). If /usr/e4d_qc/bin is in the executable path, then 'bx' will suffice.
mtr_opt	integer (0 or 1)	*mtr_opt* is the mesh translation option. If *mtr_opt* is set to 1 then the mesh is translated internally to preserve numerical precision. If *mtr_opt* is set to 0, the mesh translation is not applied. Mesh translation occurs internally to E4D, and does not affect user interface in general use (i.e. results are always reported in the original coordinates). Mesh translation is particularly important when using global coordinate systems, and is generally recommended.

5.7 ER Mesh Generation Tutorial 1.1: Two Buried Blocks

5.7.1 Conceptual Diagram

In this example, we build a mesh comprising surface electrodes, two subsurface boxes, a foreground zone defined for convenient visualization, and a boundary zone that extends from the foreground zone to the external boundaries. A conceptual diagram of the mesh, including control points and zone assignments to be defined in the mesh configuration file, is shown in Figure 5.2. The mesh consists of three lines of surface electrodes, where line 1 includes control points 1 through 16, line 2 includes control points 17 through 32, and line 3 includes control points 33 through 48. Control points 49 through 52 and 57 through 60 define the foreground region, which is assigned to be zone 1. Control points 61 through 68 define the western-most box, which is assigned to be zone 2. Control points 69 through 76 define the eastern-most box, which is assigned to be zone 3. The region extending from the foreground to the outer boundaries is not shown, but is assigned to be zone 4. The mesh configuration file for Figure 5.2 (two_blocks.cfg) is shown below with annotations denoting the corresponding mesh configuration file variables described above. It is also included with the E4D distribution under *<e4d_dir>tutorial/mode_1/two_blocks/two_blocks.cfg.*

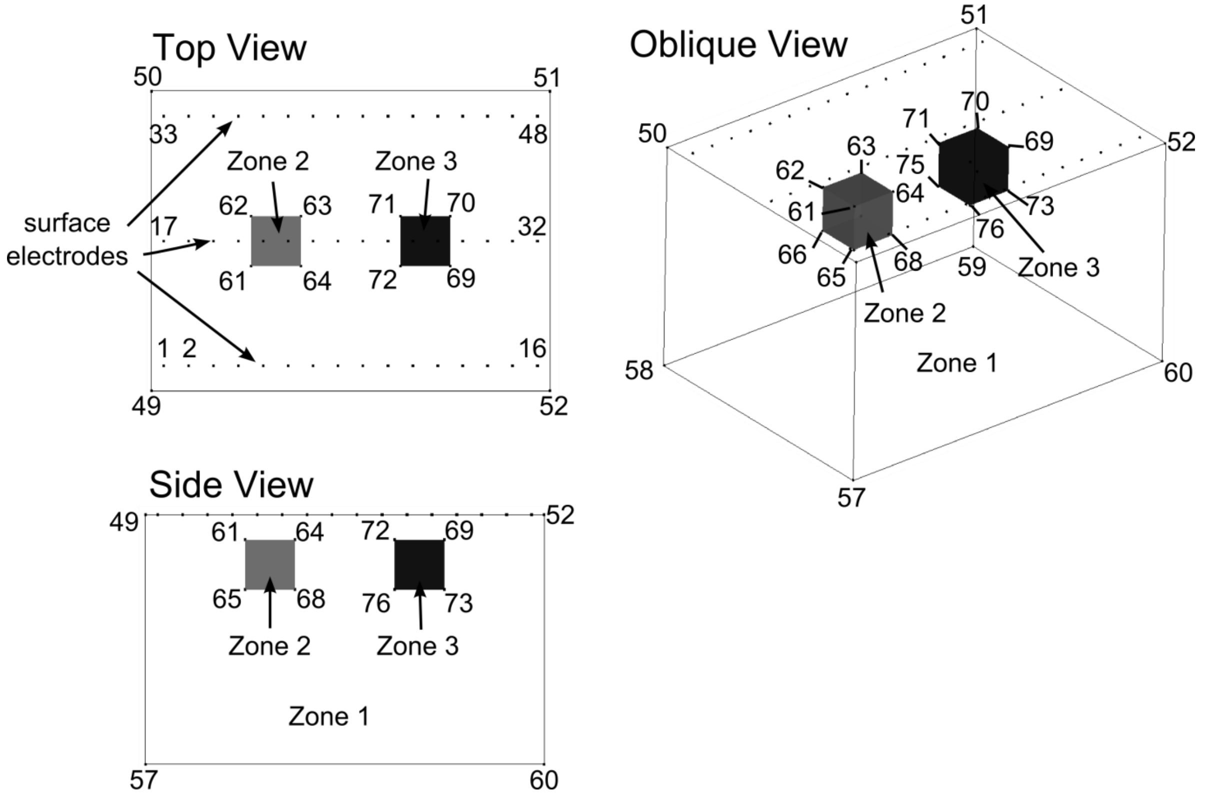


Figure 5.2. Control point and zone map for mesh configuration file two_blocks.cfg. External boundary points 54-56 and internal refine points 77-124 are not shown. The part of the domain extending from zone 1 to the external boundaries (zone 4) is also not shown.

5.7.2 Building the Mesh

In addition to the mesh configuration file, the run configuration file *e4d.inp* is needed to build the mesh. In mode 1, the only file that must be listed is the mesh configuration file. The run configuration file for this example is shown below.

<begin run configuration file e4d.inp> (this line is not included in the file)

```
1
two_blocks.cfg
```

<end run configuration file e4d.inp> (this line is not included in the file)

To build the mesh, execute the command:

<e4d_dir>/bin/mpirun -np 1 <e4d_dir>/bin/e4d

or

mpirun -np 1 e4d

if *<e4d_dir>/bin* is in the executable path, where *<e4d_dir>* is the E4D installation directory. E4D builds the input files for triangle and tetgen, and calls each to build the mesh. E4D then builds a conductivity file for the mesh as specified in the mesh configuration file, and calls bx to build an exodus file for visualization. The files generated include:

File	Generated by	Description
two_blocks.poly	e4d	Tetgen input file generated by E4D. Not used in run modes > 1.
two_blocks.1.node	tetgen	Tetgen node file. Gives the nodal coords of the mesh.
two_blocks.1.ele	tetgen	Tetgen element file. Describes how nodes are connected to form elements.
two_blocks.1.face	tetgen	Tetgen face file. Describes which nodes form boundary faces.
two_blocks.1.neigh	tetgen	Tetgen neighbor file. Provides neighbors of all elements.
two_blocks.sig	e4d	Conductivity file. Describes element conductivities and is written to the exodus file.
two_blocks.exo	bx	Exodus file used to visualize mesh and conductivity distribution.

5.7.3 Two_Block Mesh Visualization

Figure 5.3 shows a visualization of the mesh produced using VisIt with the exodus file *two_blocks.exo*.

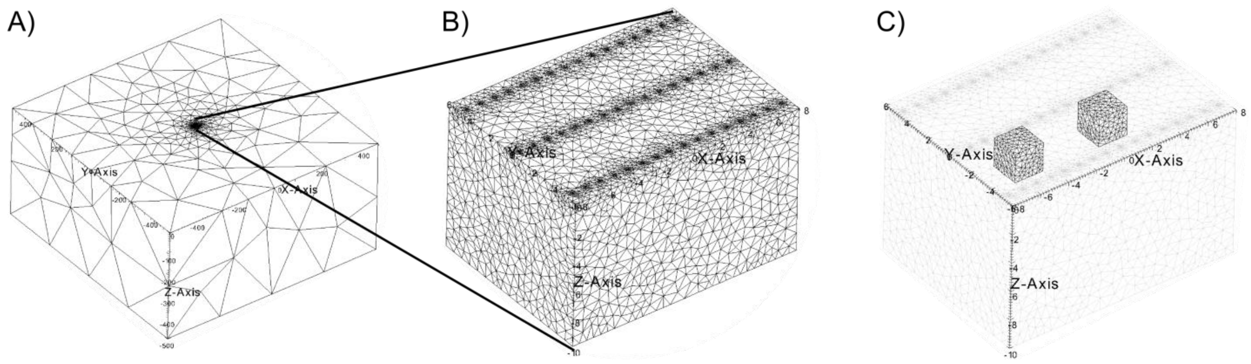


Figure 5.3. A) Oblique view of mesh show outer boundaries of zone 4 and inner refined region comprising the electrodes and box bounded by zone 1. B) Oblique view showing other boundaries of zone 1 and mesh refinement at electrode locations. C) Oblique view with zone 1 boundaries shown in transparency to reveal zone 2 and zone 3 boundaries.

5.7.4 Mesh Configuration File: *two_blocks.cfg*

<begin file two_blocks.cfg>   this line is not included in two_blocks.cfg

```
1.3 1e12                  mesh quality (m_qual) max volume (max_evol_def)
-500                      bottom of mesh elevation (m_bot)
1                         flag to build mesh (tet_build_flag)
"tetgen"                  command to run tetgen (tet_loc)
"triangle"                command to run triangle (tri_loc)
124                       number of control points (n_cpts)
1 -7.5 -5 0 1             1 x1 y1 z1 b_flag_1
2 -6.5 -5 0 1             2 x2 y2 z2 b_flag_2
3 -5.5 -5 0 1             .
4 -4.5 -5 0 1             .
5 -3.5 -5 0 1             .
6 -2.5 -5 0 1
7 -1.5 -5 0 1
8 -0.5 -5 0 1
9 0.5 -5 0 1
10 1.5 -5 0 1
11 2.5 -5 0 1
12 3.5 -5 0 1
13 4.5 -5 0 1
14 5.5 -5 0 1
15 6.5 -5 0 1
16 7.5 -5 0 1
17 -7.5 0 0 1
18 -6.5 0 0 1
19 -5.5 0 0 1
20 -4.5 0 0 1
21 -3.5 0 0 1
22 -2.5 0 0 1
23 -1.5 0 0 1
24 -0.5 0 0 1
25 0.5 0 0 1
26 1.5 0 0 1
27 2.5 0 0 1
28 3.5 0 0 1
29 4.5 0 0 1
30 5.5 0 0 1
31 6.5 0 0 1
32 7.5 0 0 1
33 -7.5 5 0 1
34 -6.5 5 0 1
35 -5.5 5 0 1
36 -4.5 5 0 1
37 -3.5 5 0 1
38 -2.5 5 0 1
39 -1.5 5 0 1
40 -0.5 5 0 1
41 0.5 5 0 1
42 1.5 5 0 1
43 2.5 5 0 1
44 3.5 5 0 1
45 4.5 5 0 1
46 5.5 5 0 1
47 6.5 5 0 1
48 7.5 5 0 1
```

```
49 -8 -6 0 1            upper control points for foreground (zone 1)
50 -8 6 0 1
51 8 6 0 1
52 8 -6 0 1

53 -500 -500 0 2      boundary control points
54 -500 500 0 2
55 500 500 0 2
56 500 -500 0 2

57 -8 -6 -10 0          lower control points for forground (zone 1)
58 -8 6 -10 0
59 8 6 -10 0
60 8 -6 -10 0

61 -4.0 -1.0 -1.0 0    Upper control points for left block (zone 2)
62 -4.0 1.0 -1.0 0
63 -2.0 1.0 -1.0 0
64 -2.0 -1.0 -1.0 0

65 -4.0 -1.0 -3.0 0    lower control points for left block (zone 2)
66 -4.0 1.0 -3.0 0
67 -2.0 1.0 -3.0 0
68 -2.0 -1.0 -3.0 0

69 4.0 -1.0 -1.0 0    upper control points for right block (zone 3)
70 4.0 1.0 -1.0 0
71 2.0 1.0 -1.0 0
72 2.0 -1.0 -1.0 0

73 4.0 -1.0 -3.0 0    lower control points for right block (zone 3)
74 4.0 1.0 -3.0 0
75 2.0 1.0 -3.0 0
76 2.0 -1.0 -3.0 0

77 -7.5 -5 -0.05 0    additional points for electrode refinement
78 -6.5 -5 -0.05 0
79 -5.5 -5 -0.05 0
80 -4.5 -5 -0.05 0
81 -3.5 -5 -0.05 0
82 -2.5 -5 -0.05 0
83 -1.5 -5 -0.05 0
84 -0.5 -5 -0.05 0
85 0.5 -5 -0.05 0
86 1.5 -5 -0.05 0
87 2.5 -5 -0.05 0
88 3.5 -5 -0.05 0
89 4.5 -5 -0.05 0
90 5.5 -5 -0.05 0
91 6.5 -5 -0.05 0
92 7.5 -5 -0.05 0
93 -7.5 0 -0.05 0
94 -6.5 0 -0.05 0
95 -5.5 0 -0.05 0
96 -4.5 0 -0.05 0
```

```
97 -3.5 0 -0.05 0
98 -2.5 0 -0.05 0
99 -1.5 0 -0.05 0
100 -0.5 0 -0.05 0
101 0.5 0 -0.05 0
102 1.5 0 -0.05 0
103 2.5 0 -0.05 0
104 3.5 0 -0.05 0
105 4.5 0 -0.05 0
106 5.5 0 -0.05 0
107 6.5 0 -0.05 0
108 7.5 0 -0.05 0
109 -7.5 5 -0.05 0
110 -6.5 5 -0.05 0
111 -5.5 5 -0.05 0
112 -4.5 5 -0.05 0
113 -3.5 5 -0.05 0
114 -2.5 5 -0.05 0
115 -1.5 5 -0.05 0
116 -0.5 5 -0.05 0
117 0.5 5 -0.05 0
118 1.5 5 -0.05 0
119 2.5 5 -0.05 0
120 3.5 5 -0.05 0
121 4.5 5 -0.05 0
122 5.5 5 -0.05 0
123 6.5 5 -0.05 0
124 7.5 5 -0.05 0

17                      number of internal planes (n_plc)
4 10                    number of points in plc 1 (np1), boundary number(b_num_1)
49 50 58 57             control points in plc 1: western boundary of zone 1
4 10                    np2 b_num_2
50 51 59 58             control points in plc 2: northern boundary of zone 1
4 10
51 52 60 59             eastern boundary of zone 1
4 10
52 49 57 60             southern boundary of zone 1
4 10
57 58 59 60             bottom boundary of zone 1
4 11
61 62 66 65             western boundary of zone 2
4 11
62 63 67 66             northern boundary of zone 2
4 11
63 64 68 67             eastern boundary of zone 2
4 11
64 61 65 68             southern boundary of zone 2
4 11
61 62 63 64             upper boundary of zone 2
4 11
65 66 67 68             lower boundary of zone 2
4 12
69 70 74 73             eastern boundary of zone 2
4 12
70 71 75 74             northern boundary of zone 2
```

```
4 12
71 72 76 75                 western boundary of zone 2
4 12
72 69 73 76                 southern boundary of zone 2
4 12
69 70 71 72                 upper boundary of zone 2
4 12
73 74 75 76                 lower boundary of zone 2

0                                     number of holes(n_holes)

4                                     number of zones(n_zones)
1 0.0 0.0 -5.0 0.1 0.002              zone1  xz_1 yz_1 zz_1 mz_vol_1 cond_1
2 -2.5 0.0 -2.5 0.01 0.002            zone2  xz_2 yz_2 zz_2 mz_vol_2 cond_2
3 2.5 0.0 -2.5 0.01 0.002             .
4 0.0 0.0 -20.0 1e12 0.002             .

1                                     flag to build exodus file (bex_flag)
'bx'                                  command to build exodus file (bex_loc)
1                                     translate the mesh internally to preserve numerical precision
```

<end file two_blocks.cfg> this line is not included in two_blocks.cfg

5.8 ER Mesh Generation Tutorial 1.2: Buried Metallic Box, Sheet, and Line

5.8.1 Overview

In this example, we demonstrate how to build a mesh with infinite conductivity inclusions. This example builds upon concepts presented in ER Mesh Generation Tutorial 1.1; users are encouraged to review and understand that example prior to reviewing this example.

5.8.2 Conceptual Diagram

In this example, we build a mesh comprising surface electrodes, three buried infinite conductivity structures (a box, a sheet, and a line), a foreground zone defined for convenient visualization, and a boundary zone that extends from the foreground zone to the external boundaries. A conceptual diagram of the subsurface is shown in Figure 5.4. The mesh consists of three lines of surface electrodes, where line 1 includes control points 1 through 16, line 2 includes control points 17 through 32, and line 3 includes control points 33 through 48. Control points 49 through 52 and 57 through 60 define the foreground region, which is assigned zone 1. Control points 61 through 68 define the conductive box. Control points 69,70,73 and 74 through 76 define the conductive sheet, and points 125-148 define the conductive line. The region extending from the foreground to the outer boundaries is not shown, but is assigned zone 2. The mesh configuration file for Figure 5.4 (mbsl.cfg) is shown below, with annotations denoting the corresponding mesh configuration file variables described in the mesh generation section of the user guide. It is also included with the E4D distribution under *<e4d_dir>/tutorial/mode_1/metal_box_sheet_line/mbsl.cfg.*

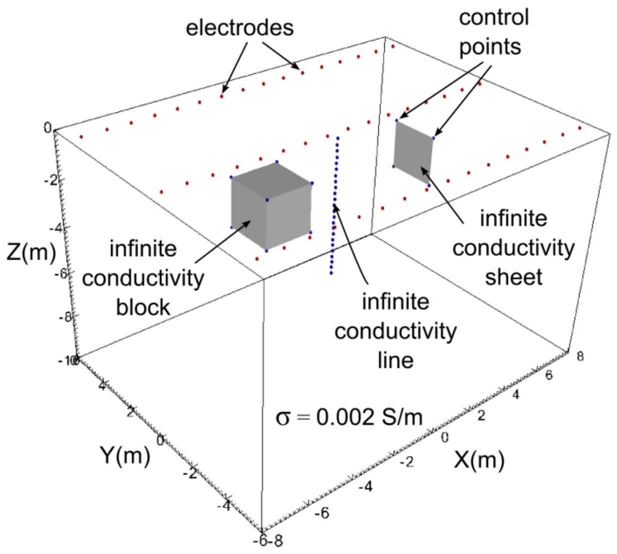

electrodes
control points
infinite conductivity block
infinite conductivity sheet
infinite conductivity line
σ = 0.002 S/m
Z(m)
Y(m)
X(m)

Figure 5.4. Conceptual diagram of surface electrodes with a subsurface containing an infinite conductivity box, sheet, and line embedded in a background medium with a conductivity of 0.002 S/m. Each of the control points and electrode points shown are listed in the mesh configuration file *mbsl.cfg*.

5.8.3 Building the mbsl Mesh

In addition to the mesh configuration file, the run configuration file *e4d.inp* is needed to build the mesh. In mode 1, the only file that must be listed is the mesh configuration file. The run configuration file for this example is shown below.

<begin run configuration file e4d.inp> (this line is not included in the file)

```
1
mbsl.cfg
```

<end run configuration file e4d.inp> (this line is not included in the file)

To build the mesh, execute the command:

<e4d_dir>/bin/mpirun -np 1 <e4d_dir>/bin/e4d

or

mpirun -np 1 e4d

if *<e4d_dir>/bin* is in the executable path, where *<e4d_dir>* is the E4D installation directory. E4D builds the input files for triangle and tetgen, and calls each to build the mesh. E4D then builds a conductivity file for the mesh as specified in the mesh configuration file, and calls bx to build an exodus file for visualization. The files generated include:

File	Generated by	Description
mbsl.poly	e4d	Tetgen input file generated by E4D. Not used in run modes > 1.
mbsl.1.node	tetgen	Tetgen node file. Gives the nodal coords of the mesh.
mbsl.1.ele	tetgen	Tetgen element file. Describes how nodes are connected to form elements.
mbsl.1.face	tetgen	Tetgen face file. Describes which nodes form boundary faces.
mbsl.1.neigh	tetgen	Tetgen neighbor file. Provides neighbors of all elements.
mbsl.sig	e4d	Conductivity file. Describes element conductivities and is written to the exodus file.
mbsl.exo	bx	Exodus file used to visualize mesh and conductivity distribution.

5.8.4 mbsl Mesh Visualization

Figure 5.5 shows a visualization of the mesh produced using VisIt with the exodus file *mbsl.exo*.

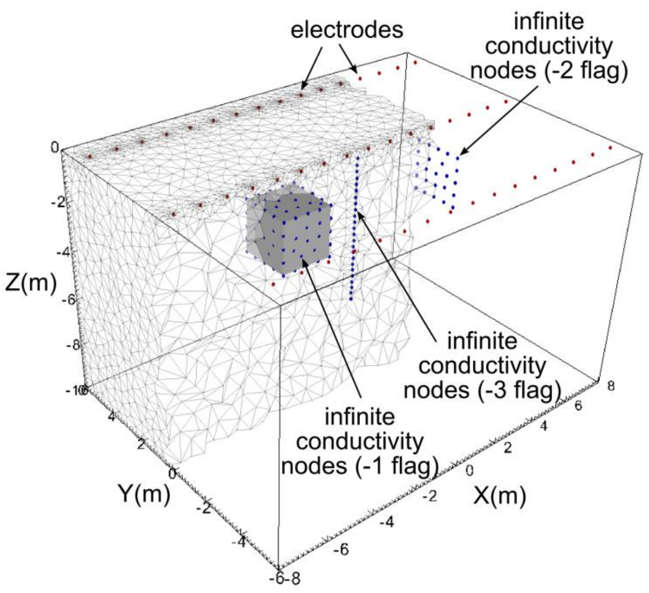

electrodes
infinite conductivity nodes (-2 flag)
infinite conductivity nodes (-3 flag)
infinite conductivity nodes (-1 flag)
Z(m)
Y(m)
X(m)
0
-2
-4
-6
-8
-10
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8

Figure 5.5. Cut-out view of the mbsl mesh showing nodes added by tetgen for the three infinite conductivity boundaries. Note that the grey region representing the infinite conductivity box is shown only for reference. The mesh is hollow in the interior of the box; only the outer boundary of the box is included in the mesh.

5.8.5 Mesh Configuration File: *mbsl.cfg*

<begin mesh configuration file mbsl.cfg>   this line is not included in two_blocks.cfg

```
1.3 1e12                                mesh quality (m_qual), max volume (max_evol_def)
-500                                    bottom of mesh elevation (m_bot)
1                                       flag to build mesh (tet_build_flag)
"tetgen"                                command to run tetgen (tet_loc)
"triangle"                              command to run triangle (tri_loc)

148                                     number of control points (n_cpts)
1 -7.5 -5 0 1                           1 x1 y1 z1 b_flag_1
2 -6.5 -5 0 1                           2 x2 y2 z2 b_flag_2
3 -5.5 -5 0 1                           .
4 -4.5 -5 0 1                           .
5 -3.5 -5 0 1                           .
6 -2.5 -5 0 1
7 -1.5 -5 0 1
8 -0.5 -5 0 1
9 0.5 -5 0 1
10 1.5 -5 0 1
11 2.5 -5 0 1
12 3.5 -5 0 1
13 4.5 -5 0 1
14 5.5 -5 0 1
15 6.5 -5 0 1
16 7.5 -5 0 1
17 -7.5 0 0 1
18 -6.5 0 0 1
19 -5.5 0 0 1
20 -4.5 0 0 1
21 -3.5 0 0 1
22 -2.5 0 0 1
23 -1.5 0 0 1
24 -0.5 0 0 1
25 0.5 0 0 1
26 1.5 0 0 1
27 2.5 0 0 1
28 3.5 0 0 1
29 4.5 0 0 1
30 5.5 0 0 1
31 6.5 0 0 1
32 7.5 0 0 1
33 -7.5 5 0 1
34 -6.5 5 0 1
35 -5.5 5 0 1
36 -4.5 5 0 1
37 -3.5 5 0 1
38 -2.5 5 0 1
39 -1.5 5 0 1
40 -0.5 5 0 1
41 0.5 5 0 1
42 1.5 5 0 1
43 2.5 5 0 1
```

```
44 3.5 5 0 1
45 4.5 5 0 1
46 5.5 5 0 1
47 6.5 5 0 1
48 7.5 5 0 1

49 -8 -6 0 1                        upper control points for foreground (zone 1)
50 -8 6 0 1
51 8 6 0 1
52 8 -6 0 1

53 -500 -500 0 2                    boundary control points
54 -500 500 0 2
55 500 500 0 2
56 500 -500 0 2

57 -8 -6 -10 0                      lower control points for forground (zone 1)
58 -8 6 -10 0
59 8 6 -10 0
60 8 -6 -10 0

61 -4.0 -1.0 -1.0 -1                Upper control points conductive block
62 -4.0 1.0 -1.0 -1
63 -2.0 1.0 -1.0 -1
64 -2.0 -1.0 -1.0 -1

65 -4.0 -1.0 -3.0 -1                lower control points conductive block
66 -4.0 1.0 -3.0 -1
67 -2.0 1.0 -3.0 -1
68 -2.0 -1.0 -3.0 -1

69 4.0 -1.0 -1.0 -2                 control points 69-70 define the upper edge of the metal sheet
70 4.0 1.0 -1.0 -2
71 2.0 1.0 -1.0 0
72 2.0 -1.0 -1.0 0

73 4.0 -1.0 -3.0 -2                 control points 73-74 define the lower edge of the metal sheet
74 4.0 1.0 -3.0 -2
75 2.0 1.0 -3.0 0
76 2.0 -1.0 -3.0 0

77 -7.5 -5 -0.05 0                  additional points for electrode refinement
78 -6.5 -5 -0.05 0
79 -5.5 -5 -0.05 0
80 -4.5 -5 -0.05 0
81 -3.5 -5 -0.05 0
82 -2.5 -5 -0.05 0
83 -1.5 -5 -0.05 0
84 -0.5 -5 -0.05 0
85 0.5 -5 -0.05 0
86 1.5 -5 -0.05 0
87 2.5 -5 -0.05 0
88 3.5 -5 -0.05 0
89 4.5 -5 -0.05 0
90 5.5 -5 -0.05 0
91 6.5 -5 -0.05 0
```

```
92 7.5 -5 -0.05 0
93 -7.5 0 -0.05 0
94 -6.5 0 -0.05 0
95 -5.5 0 -0.05 0
96 -4.5 0 -0.05 0
97 -3.5 0 -0.05 0
98 -2.5 0 -0.05 0
99 -1.5 0 -0.05 0
100 -0.5 0 -0.05 0
101 0.5 0 -0.05 0
102 1.5 0 -0.05 0
103 2.5 0 -0.05 0
104 3.5 0 -0.05 0
105 4.5 0 -0.05 0
106 5.5 0 -0.05 0
107 6.5 0 -0.05 0
108 7.5 0 -0.05 0
109 -7.5 5 -0.05 0
110 -6.5 5 -0.05 0
111 -5.5 5 -0.05 0
112 -4.5 5 -0.05 0
113 -3.5 5 -0.05 0
114 -2.5 5 -0.05 0
115 -1.5 5 -0.05 0
116 -0.5 5 -0.05 0
117 0.5 5 -0.05 0
118 1.5 5 -0.05 0
119 2.5 5 -0.05 0
120 3.5 5 -0.05 0
121 4.5 5 -0.05 0
122 5.5 5 -0.05 0
123 6.5 5 -0.05 0
124 7.5 5 -0.05 0
125 0 0 -0.25 -3                    points 125 - 126 define a metallic line (e.g. well casing)
126 0 0 -0.5 -3
127 0 0 -0.75 -3
128 0 0 -1 -3
129 0 0 -1.25 -3
130 0 0 -1.5 -3
131 0 0 -1.75 -3
132 0 0 -2 -3
133 0 0 -2.25 -3
134 0 0 -2.5 -3
135 0 0 -2.75 -3
136 0 0 -3 -3
137 0 0 -3.25 -3
138 0 0 -3.5 -3
139 0 0 -3.75 -3
140 0 0 -4 -3
141 0 0 -4.25 -3
142 0 0 -4.5 -3
143 0 0 -4.75 -3
144 0 0 -5 -3
145 0 0 -5.25 -3
146 0 0 -5.5 -3
```

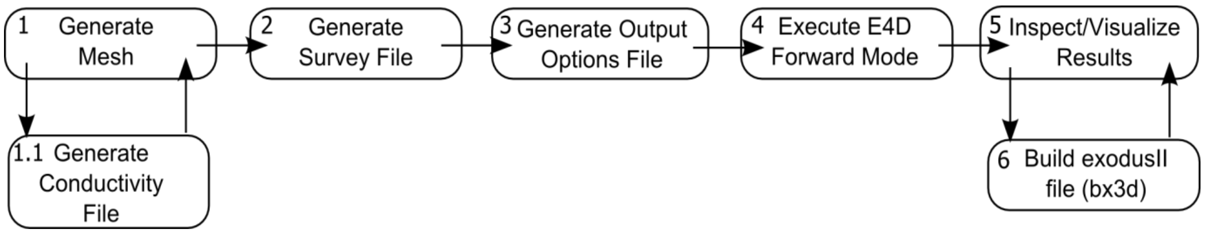
```
147 0 0 -5.75 -3
148 0 0 -6 -3

12                                     number of internal planes (n_plc)
4 10                                   number of points in plc 1 (np1), boundary number(b_num_1)
49 50 58 57                            control points in plc 1: western boundary of zone 1
4 10                                   np2 b_num_2
50 51 59 58                            control points in plc 2: northern boundary of zone 1
4 10                                   .
51 52 60 59                            eastern boundary of zone 1
4 10                                   .
52 49 57 60                            southern boundary of zone 1
4 10                                   .
57 58 59 60                            bottom boundary of zone 1
4 -1
61 62 66 65                            western boundary of metal box
4 -1
62 63 67 66                            northern boundary of metal box
4 -1
63 64 68 67                            eastern boundary of metal box
4 -1
64 61 65 68                            southern boundary of metal box
4 -1
61 62 63 64                            upper boundary of metal box
4 -1
65 66 67 68                            lower boundary of metal box
4 -2
69 70 74 73                            metal sheet

1                                      number of holes ... 1 for the metal box
2 -2.5 0.0 -2.5                        point inside the metal box

2                                      number of zones ... 2
1 0.0 0.0 -5.0 0.1 0.002               1 xz_1 yz_1 zz_1 mz_vol_1 cond_1
2 0.0 0.0 -20.0 1e12 0.002

1                                      flag to build exodus file (bex_flag)
'../../../bin/bx'                      command to build exodus file (bex_loc)
1                                      translate the mesh internally to preserve numeric precision
```

<**end mesh configuration file mbsl.cfg**>   this line is not included in two_blocks.cfg

6.0 Mode 2: ER Forward Run Mode

6.1 Introduction

Forward modeling is defined herein as the process whereby E4D simulates and reports the subsurface electrical potential generated (real and/or complex) per ampere of transmitted current, given the subsurface conductivity distribution specified in the conductivity file listed in *e4d.inp*. Results are reported as simulated measurements and/or simulated potential distributions according to specifications givenin the output options file listed in *e4d.inp*. The general flow for forward modeling is shown below.


1 Generate Mesh
1.1 Generate Conductivity File
2 Generate Survey File
3 Generate Output Options File
4 Execute E4D Forward Mode
5 Inspect/Visualize Results
6 Build exodusII file (bx3d)

1. Generate the computational mesh
   The first step in forward modeling is generating the computational mesh. Mesh generation is described in the previous chapter. The basic steps for generating the mesh include constructing the mesh configuration file, executing the mesh generation with E4D, and visualizing the results using bx and VisIt. A conductivity file is automatically generated as part of the mesh generation process, where the conductivity value is specified for each zone defined in the mesh. It is also possible to construct user-defined conductivity files outside of E4D (e.g., using stochastic simulation software) for forward modeling. Examples are provided in Sections 7.9.4 and 9.1.2.

2. Generate the survey file
   Survey files give the locations of electrodes, define how those electrodes are used to generate a survey, and specify observed measurements and standard deviations. In forward modes, measurement results and standard deviations are not used, but placeholders must be included in the survey file.

3. Generate the output options file
   The output options file specifies how simulated results should be reported, including simulated measurements and/or specified subsurface potential distributions for visualization. If specified in the output options file, simulated survey results are written to the simulated data file, and potential distributions are written to potential files.

4. Once the mesh has been generated, and the survey file and output option files have been constructed, the *e4d.inp* file is constructed and E4D is executed using the command:

   mpirun -np <num_proc> e4d

   where *num_proc* is the number of processors E4D should use during execution, which must be at least 2 and no more than *ne*+1 where *ne* is the number of electrodes plus the number of infinite conductivity (i.e., metallic) inclusions in the simulation.

5. Upon successful forward execution, E4D generates a number of output files as specified in the output options file. If potential distributions are generated, they may be investigated using bx to build a

corresponding exodusII file for visualization with VisIt. Examples are provided at the end of this chapter.

6.2 File Format for *e4d.inp*

The *e4d.inp* format requirements for mode 2 are described below. Each row in the "Variable" column describes the required input for the corresponding row of *e4d.inp***.**

Variable	Type	Description
run_mode	integer (string for analytic mode, see description)	For an ER forward simulation the *run_mode* should be set to either 2 for a numerical solution, or "ANALYTIC","Analytic", or "analytic" for an analytic solution.
mesh_file	string	The *mesh_file* variable should set to <*mesh_prefix*>.1.node, or <*mesh_prefix*>.1.ele, where <*mesh_prefix*> is the prefix of the mesh files. These files are produced in mesh generation mode using a mesh configuration file name <*mesh_prefix*>.cfg.
survey_file	string	*survey_file* is the name of the user-created file that describes electrode locations and survey configuration as described in Appendix A.
conductivity_file	string	*conductivity_file* specifies the name of the conductivity file, which provides the conductivity of each element in the mesh. Conductivity files are produced by E4D during mesh generation and inverse computations. Conductivity files may also be constructed outside of E4D to create customized conductivity distributions.
out_opts_file	string	Specifies the name of the output options file.

6.3 Analytic Solutions

If a mode is specified as '*analytic*'**,** '*ANALYTIC*', or '*Analytic*' in *e4d.inp*, E4D will compute the analytic forward solution for a homogeneous halfspace, and output results as specified in the output options file. In this case, the mesh should be constructed with a flat surface (i.e., no surface topography, constant surface elevation) and a homogeneous conductivity. E4D will use the average surface elevation of the mesh as the halfspace boundary, and the average conductivity specified in the conductivity file as the homogeneous subsurface conductivity. Both surface and subsurface electrodes may be used in this mode.

6.4 Numeric Solutions

In all run modes except '*analytic*', subsurface potential distributions are computed by a weak-form finite element solution to the Poisson equation. See the E4D theory guide for additional details.

6.5 Mode 1 Reporting and Output

6.5.1 Screen Output

All E4D modes generate information describing the current status and progress of computations, and report to standard output (e.g., the computer screen). When E4D is executed on the command line, screen output is printed to the terminal from which E4D was executed. In batch mode, screen output is directed to whatever the operating system specifies as the standard output. Most parallel computing queuing systems write to a default or user specified file, which is continuously updated by E4D to indicate the status of computations.

6.5.2 E4D Log File: *e4d.log*

All E4D modes record execution information to the E4D log file, *e4d.log***.** In addition to the date and time of execution, the log file records all information required to reproduce the simulation, including the names of all input files and corresponding summary statistics. The log file also produces summary information concerning the number of processors used for the simulation, and the computational load on each processor. For inversion modes, *e4d.log* provides convergence information and other inversion progress parameters at each outer iteration. The *e4d.log* file is overwritten with each new execution of E4D. If necessary, users should rename or move *e4d.log* prior to additional E4D computations.

6.5.3 Simulated Data and Survey File

If specified in the output options file, E4D will produce a simulated data file and a simulated survey file. The simulated survey file has the same format as a standard survey file, except that the simulated measurements are entered in place of the observed measurements. This file is meant to aid synthetic imaging studies, whereby simulated data collected within a known conductivity distribution are used to investigate aspects of the inversion. The simulated survey file is named after the conductivity file used in the forward simulation, appended by the ".srv" suffix. For example, if the conductivity file is named *two_blocks.sig*, then a corresponding simulated survey named *two_blocks.sig.srv* is constructed by E4D at the end of the forward run. Note that the standard deviation of each measurement is entered as 5% of the transfer resistance magnitude as a placeholder, but noise is not added to the simulated data.

6.5.4 Potential Files

If specified in the output options file, potential files are output at the end of the simulation. Potential files may be used to visualize the potential field simulated for a given measurement by entering the corresponding potential file into an exodus file using bx, and visualizing the field using VisIt.

6.6 ER Forward Run Tutorial 2.1: Two Buried Blocks

6.6.1 Overview

This example builds upon the mesh generation example shown in ER Mesh Generation Tutorial 1: Two Buried Blocks, and is presented in two sections. In the first section, mode 2 is used to investigate numeric solution accuracy for the mesh by producing both the analytic and numeric solutions to the homogeneous halfspace problem. In the second section, a simulated survey is generated for a heterogeneous halfspace, and a selected potential field simulated for one of the survey measurements is visualized using VisIt.

6.6.2 Using Mode 2 to Investigate Numeric Solution Accuracy

The objective of this section is to demonstrate the use of mode 2 by comparing simulated survey results produced by the analytic and numeric solutions. We begin with the computational mesh and electrode configuration generated in ER Mesh Generation Tutorial 1: Two Buried Blocks, which is shown in Figure 6.1 below.

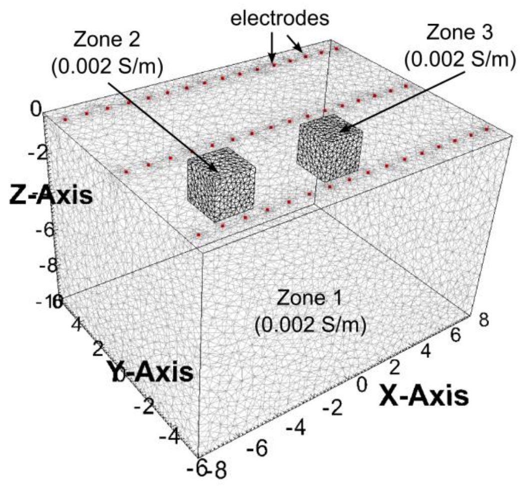


Figure 6.1. Zones 1-3 of computational mesh and electrode locations for the homogeneous conductivity case. Zone 4 extends to the outer boundary and is not shown,

After mesh construction, the files describing the mesh are located in *two_blocks.1.node, two_blocks.1.ele, two_blocks.1.face, two_blocks.1.face*, and *two_blocks.trn*. The conductivity file *two_blocks.sig* and the exodus file *two_blocks.exo* were also produced by E4D as specified in the mesh configuration file for this example.

6.6.3 Survey File: *two_blocks.srv*

With the mesh and conductivity files constructed, the next step is to generate the survey file. The text below shows parts of the survey file used for this example, which includes a series of inline and cross-line dipole-dipole measurements. The full survey file is included with the E4D distribution at *<e4d_dir>/tutorials/mode_2/two_blocks/two_blocks.srv***.**

<begin two_blocks.srv> (this line is not included in the file, see survey file format section for reference)

```
48                number of electrodes
1 -7.5 -5 0 1      eind ex ey ez eflag
2 -6.5 -5 0 1
3 -5.5 -5 0 1
.
.
.
46 5.5 5 0 1
47 6.5 5 0 1
48 7.5 5 0 1

2070                number of measurements
1 1 2 3 4 10.0 1.0    m_ind a b m n v_obs v_std
2 1 2 4 5 10.0 1.0    note: the v_obs and v_std are arbitrary placeholders in this example
3 1 2 5 6 10.0 1.0
.
.
.
1449  23    26    27    30    10.0 1.0   this potential distribution will be visualized in the next section
.
.
.
2068 30 46 31 47 10 1
2069 30 46 32 48 10 1
2070 31 47 32 48 10 1
```

<end two_blocks.srv> (this line is not included in the file)

6.6.4 Output Options File: *two_blocks.out*

Mode 2 also requires an output options file. The text below shows the output options used for this example, which is also included the E4D distribution under *<e4d_dir>/tutorials/mode_2/two_blocks/two_blocks.out***.** For this example, we instruct E4D to produce a simulated data file, and to name that file *two_blocks.dpd*. We also instruct E4D to build the potential distribution file for measurement 1449.

<begin two_blocks.out> (this line is not included in the file_

```
1                          dp_flag = 1 to print simulated data file
two_blocks.dpd             dp_file (name of simulated data file)
1                          number of potential distribution files to construct
1449                       index of measurement for which to produce potential distribution file
```
<end two_blocks.out> (note: this line is not included in the file)

6.6.5 Mode 2 e4d.inp

The final step before executing E4D is to construct *e4d.inp***.** As demonstrated below, we will first execute E4D in analytic mode, and then produce the numeric solution. For the analytic mode, *e4d.inp* is:

<begin e4d.inp> (this line is not included in the file)
```
'analytic'                  run mode
two_blocks.1.node           mesh node file name
two_blocks.srv              survey file name
two_blocks.sig              conductivity file name
two_blocks.out              output options file name
```

<end e4d.inp> (this line is not included in the file)

The strategy here is to first run E4D in analytic mode to produce a simulated survey file that includes the analytic solution to the measurements specified in *two_blocks.srv***.** The simulated survey file is then used for the the numeric solution. Because the survey file used for the numeric simulation contains the analytic solution results, the simulated data file produced by the numeric forward run will contain both the analytic and numeric simulation results for comparison. As directed in the output options file, the simulated data file name will be *two_blocks.dpd.* E4D will also produce the potential distribution file *an_potential.1449* in analytic mode, *potential.1449* in mode 2, the E4D log file *e4d.log,* and output to screen.

6.6.6 Analytic Simulation

With the input files created, E4D is executed by typing on the command line:

```
mpirun -np 1 e4d.
```

Note that slave processors are not utilized in analytic mode; the simulation is computed exclusively by the master process. Thus, only one processor is specified on the command line. If analytic mode is executed with more than one processor, the extra processors are not utilized.

Upon completion, E4D outputs several files as described above. A portion of the log file *e4d.log* is shown below for reference.

<begin e4d.log> (this line is not included in the file)

```
***************** RUNNING IN ERT ANALYTIC MODE ******************
Mode: Analytic
Mesh file: two_blocks.1.node
```

```
Survey configuration file: two_blocks.srv
Conductivity file: two_blocks.sig
Output options file: two_blocks.out
SURVEY FILE SUMMARY
Number of electrodes: 0000048
Number of measurements: 0002070
CONDUCTIVITY FILE SUMMARY
Number of conductivity values: 0000106405
MESH SUMMARY
Number of nodes: 0000018606
Number of metallic boundaries: 0000000000
Number of elements: 0000106405
Number of zones: 0000000004
.
.
.
WRITING POTENTIAL FILES
WRITING SIMULATED SURVEY FILE: two_blocks.sig.srv
Forward run complete.
```

<end e4d.log> (this line is not included in the file)

For this example, we are interested in the simulated survey file *two_blocks.sig.srv*, which contains the analytically simulated measurements. A portion of *two_blocks.sig.srv* is shown below.

<begin two_blocks.sig.srv> (this line is not included in the file)

```
48                  note the electrode block is the same as the input survey file, two_blocks.srv
1 -7.5 -5 0 1
2 -6.5 -5 0 1
3 -5.5 -5 0 1
.
.
.
46 5.5 5 0 1
47 6.5 5 0 1
48 7.5 5 0 1

2070                                 the survey block contains the simulated measurement values
1 1 2 3 4   -26.5258   1.32629
2 1 2 4 5   -6.63145   0.331573
3 1 2 5 6   -2.65258   0.132629
.
.
.
```

```
2068 30 46 31 47   127.942 6.39710
2069 30 46 32 48   50.0231 2.50116
2070 31 47 32 48   127.942 6.39710
```

<end two_blocks.sig.srv> (this line is not included in the file)

6.6.7 Numeric Simulation

To execute the numeric simulation, we first rename the simulated survey file to *two_blocks_analytic.srv* so that it is not overwritten during the numeric forward run. We then use *two_blocks_analytic.srv* as the survey file in the numeric forward run, so that the simulated data output file will report both the analytic and numeric measurement simulations. The *e4d.inp* file is first modified to be:

<begin e4d.inp> (this line is not included in the file)

```
2                              mode is changed to from 'analytic' to 2 for numeric forward run
two_blocks.1.node              mesh node file name
two_blocks_analytic.srv         survey file name is changed to use the simulated survey file
two_blocks.sig                 conductivity file name
two_blocks.out                 output options file name
```

<end e4d.inp> (this line is not included in the file)

In mode 2, E4D requires at least 2 processors, and will utilize as many as *ne+1* processors, where *ne* is the number of electrodes specified in the electrode block. In this case we use 7 processors to execute the forward run, which leaves 6 slave processors to compute the forward solutions. Since there are 48 electrodes, this leaves 8 electrodes per processor (or 8 pole solutions per processor, see E4D design guide). E4D is executed as:

```
mpirun -np 7 e4d
```

Upon completion, E4D outputs the simulated data file *two_blocks.dpd***,** a portion of which is shown below. Column 1 shows the measurement index, columns 2-5 show the "a b m n" electrode indexes for each measurement, column 6 shows the analytically simulated measurement, and column 7 shows the numerically simulated measurements.

<begin two_blocks.dpd> (this line is not included in the file)

```
2070
1 1 2 3 4   -26.5258 -25.2777
2 1 2 4 5   -6.63145 -6.51073
3 1 2 5 6   -2.65258 -2.67544
4 1 2 6 7   -1.32629 -1.32612
5 1 2 7 8   -0.757881 -0.761162
.
.
.
```

<end two_blocks.dpd> (this line is not included in the file)

6.6.8 Simulating a Survey and Viewing a Potential Field

In this section we produce a simulated survey arising from a heterogeneous conductivity distribution, and plot the potential distribution simulated for one current source/sink pair. We begin by modifying the homogeneous conductivity distribution from the previous example to that shown in Figure 6.2. This can be done by either changing the conductivity values for each zone in the mesh configuration file (*two_blocks.cfg*) and rebuilding the mesh, or by manually extracting the the zone numbers from the element file (*two_blocks.1.ele*) and constructing a new conductivity file with conductivity values assigned according to zone. If the first option is used, then the name of the new heterogeneous conductivity file (after mesh reconstruction) is *two_blocks.sig***.** We then execute E4D as before (i.e, mpirun -np 7 e4d), which produces a survey file (*two_blocks.sig.srv***)** that contains the simulated survey for the heterogeneous conductivity distributions shown in Figure 6.2.

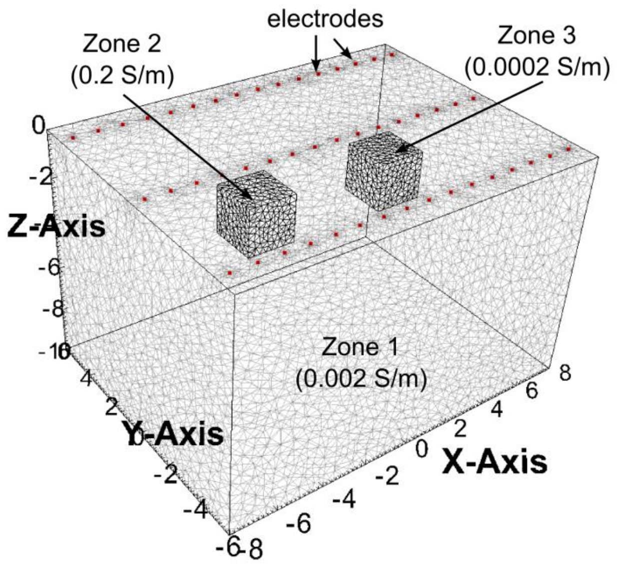


Figure 6.2. Zones 1-3 of computational mesh and electrode locations with blocks of anamolous conductivity. Zone 4 extends to the outer boundaries and is not shown.

As specified in the output options file, the potential distribution simulated for measurement 1449 is also produced upon execution of E4D and placed in the potential distribution file named *potential.1449*. The potential file can be added to the existing exodus file *two_blocks.exo***,** which was generated during mesh construction, using bx as follows (see bx documentation for details).

bx3d -af two_blocks.1 potential.1449 two_blocks.exo 1449

Figure 6.3 shows examples of potential distribution plots generated using VisIt. Figure 6.3.A shows potential isosurfaces for measurement 1449 given a homogeneous conductivity filed of 0.002 S/m, and Figure 6.3.B shows the corresponding potential isosurfaces for the conductivity distribution shown in

Figure 6.2. It is useful to render the potential distribution in terms of isosurfaces, because current flows parallel to the potential gradient. The isosurfaces therefore indicate the general nature of current flow. For example, the isosurface shown in Figure 6.3.B indicates the current is directed into the more conductive block (zone 2), and away from the less conductive block (zone 3), in comparison to the homogeneous distribution in Figure 6.3.B.

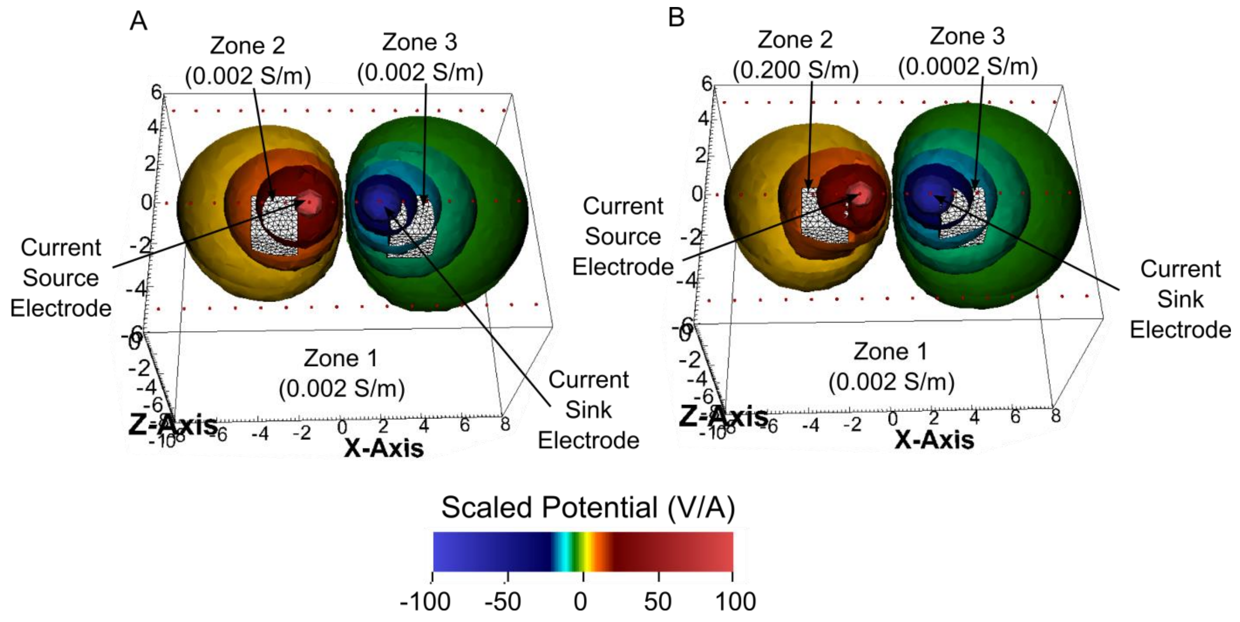


Figure 6.3. Example numerical potential distributions generated for A) a homogeneous halfspace and B) a heterogeneous halfspace with two blocks of anomalous conductivity.

6.7 ER Forward Run Tutorial 2.2: Buried Metallic Box, Sheet, and Line

6.7.1 Overview

This tutorial builds upon the ER mesh generation tutorials. Users are encouraged to review those examples before proceeding. The objective of this tutorial is to demonstrate how to use the infinite conductivity boundary modeling capabilities provide by E4D. The primary difference between modeling with and without infinite conductivity boundaries are mesh generation and construction of the survey file. The procedure for generating a mesh with infinite conductivity boundaries is discussed in the mesh generation documentation, and demonstrated in ER Mesh Generation Tutorial 1.2: Buried Metallic Box, Sheet, and Line. We begin with the mesh generated in that example, and focus in this example on constructing the survey file and visualizing potential fields. Forward modeling details pertinent to this example but demonstrated in the previous section are not repeated here.

A cut-out of the mesh generated in ER Mesh Generation Example 2: Buried Metallic Box, Sheet, and Line, is shown in Figure 6.4. The surface electrode configuration is the same as that shown in the previous tutorial, and we begin with the survey file for that example (*two_blocks.srv*) to construct the survey file for this example, which we will name *mbsl.srv*.

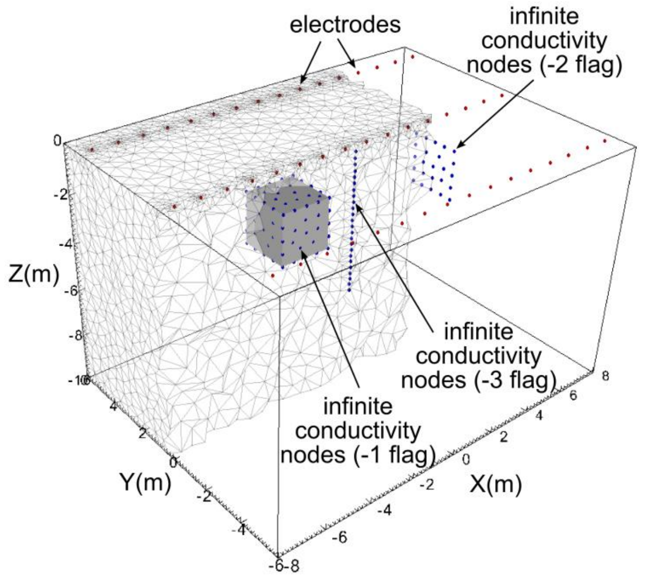

electrodes
infinite conductivity nodes (-2 flag)
infinite conductivity nodes (-3 flag)
infinite conductivity nodes (-1 flag)
Z(m)
Y(m)
X(m)

Figure 6.4. Cut-out view of the mbsl mesh showing nodes added by tetgen for the three infinite conductivity boundaries. The grey region representing the infinite conductivity box is shown for reference.

To enable each of the three infinite conductivity structures to be modeled by E4D, we need to append one electrode for each structure to the end of the electrode block of the survey file (see survey file format for details). This is required, regardless of whether or not each infinite conductivity boundary is used as a current source, sink, or potential measurement electrode. If no electrode is specified for an infinite conductivity boundary, that boundary will be ignored by E4D, meaning that 3D boundaries will be perfectly resistive (zero flux condition through the boundary), and 1D and 2D boundaries will have no special conditions imposed.

Sections of the survey file *mbsl.srv* are shown in Section 6.7.2 to illustrate. The full survey file may be found with the E4D distribution in <e4d_dir>*tuturial/mode_2/metal_box_sheet_line/mbsl.srv*. Electrodes number 49, 50, and 51 are points on the box, sheet, and line, respectively. These points were taken directly from the mesh configuration file (control points 61, 69, and 125, respectively). That is, since every control point specified in the mesh configuration file becomes a node on the mesh, it is not necessary to search the node file to locate a point on each infinite conductivity boundary. Note that the boundary flag for each infinite conductivity electrode is the corresponding boundary number specified in the mesh configuration file for each infinite conductivity boundary (i.e., -1 for the box, -2 for the sheet, -3 for the line).

Once the infinite conductivity electrodes have been added to the electrode block of the survey file, they may be used as current or potential electrodes just as any other electrode. Measurements 2071 and 2072 show two examples. In measurement 2071 the box (electrode 49) is used as the current source and the sheet (electrode 50) is used as the current sink. The line electrode (electrode 51) is used as the positive potential electrode, and electrode 1 is used as the negative potential electrode. In measurement 2072, electrode 1 is used as the current source and the line (electrode 51) is used as the current sink; electrodes 1 and 2 are used as the positive and negative potential electrodes, respectively.

6.7.2 Survey File: *mbsl.srv*

<begin survey file mbsl.srv> this line is not included in the file

```
51                number of electrodes (48 surface electrodes + 3 i.c. boundary electrodes)
1 -7.5 -5 0 1     eind x1 y1 z1 eflag (0 for subsurface, 1 for surface)
2 -6.5 -5 0 1
3 -5.5 -5 0 1
.
.
.
48 7.5 5 0 1
49 -4.0   -1.0   -1.0   -1        this is a node on the box (see control point 61 in mbsl.srv)
50  4.0   -1.0   -1.0   -2        this is a node on the sheet (see control pt. 69)
51    0      0  -0.25   -3        this is a node on the line (see control pt. 125)
2072                             number of measurements
1 1 2 3 4 10 1                   m_indx a b m n V/I stdev_V/I
2 1 2 4 5 10 1
3 1 2 5 6 10 1
.
.
.
2070   31   47   32 48   10 1
2071   49   50   51   1   10 1    injecting current from box to sheet
2072    1   51    2   3   10 1    injecting from electrode 1 to line
```

<end survey file mbsl.srv> this line is not included in the file

6.7.3 Potential Distributions for Measurement 2071 and 2072

The simulated potential distributions for measurements 2071 and 2072 were printed to potential files as specified in the output options file (see mbsl.out in the E4D distribution at <e4d_dir>/*tuturial/mode_2/metal_box_sheet_line/mbsl.out*). Using bx, each distribution was added to the exodus file *mbsl.exo*, which was created during mesh generation. Figure 6.5 shows the simulated potential distributions for each measurement as potential isosurfaces. Results above -1 m are clipped to better show the influence of the infinite conductivity boundaries in each case.

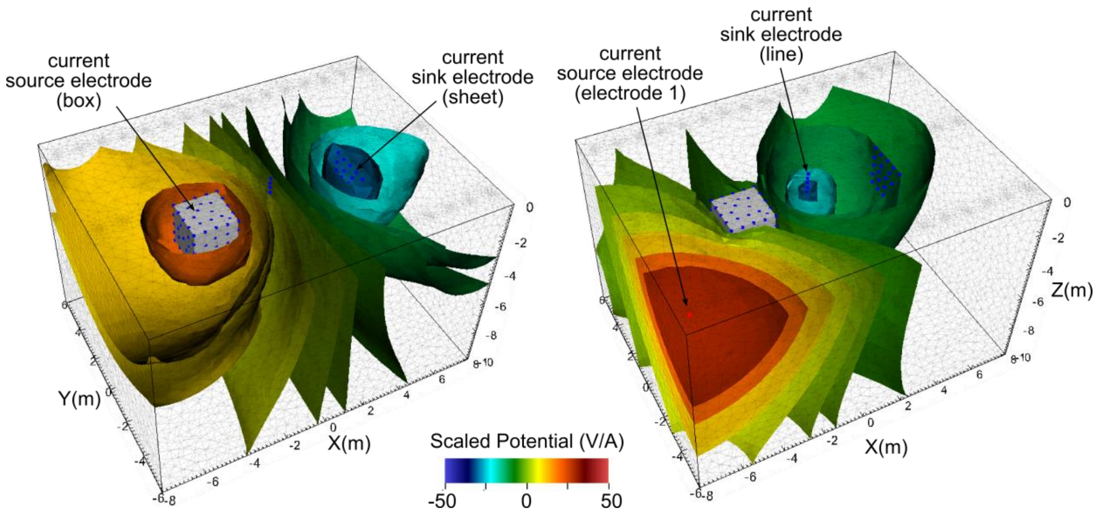


Figure 6.5. Simulated potential isosurfaces for measurement A) 2071 and B) 2072

7.0 Mode 3: ER static inversion mode

7.1 Overview

Inverse modeling refers herein to the process whereby actual or simulated electrical potential measurements are used within an optimization algorithm to estimate the electrical conductivity distribution of the subsurface. The results of an inversion are typically visualized as an image of subsurface conductivity in one form or another, such that the terms inversion, inverse solution, inverse model, and image, are synonymous. Of the run modes provided by E4D, inversion is by far the most computationally demanding in terms of both cpu cycles and memory requirements. To accommodate these requirements, the inversion algorithm is implemented in parallel; forward simulations, Jacobian matrix construction, and Jacobian matrix storage are evenly divided among the set of slave process. The theoretical and design aspects of the inversion algorithm implemented in E4D can be found in the E4D design and theory guides. In this section, focus is placed on describing how to execute static inversions within E4D, and how E4D informs users concerning the status of the inversion.

In mode 3, E4D executes a static ER inversion, meaning that ER data provided in a single survey file as well as a set of user supplied solution constraints are used to estimate the subsurface bulk conductivity distribution at some point in time. In addition to the five files describing the mesh, there are five or six files needed for E4D to execute an inversion, depending on whether or not the solution is constrained by a reference conductivity model. The name of each file is listed in *e4d.inp* as shown below.

7.2 File Format for *e4d.inp*

The *e4d.inp* format requirements for mode 3 are described below. Each row in the "Variable" column describes the required input for the corresponding row of *e4d.inp*.

Variable	Type	Description
run_mode	integer	For a static ER inversion the *run_mode* should be set to 3.
mesh_file	string	The *mesh_file* variable should set to <*mesh_prefix*>.1.node, or <*mesh_prefix*>.1.ele, where <*mesh_prefix*> is the prefix of the mesh files. These files are produced in mesh generation mode using a mesh configuration file name <*mesh_prefix*>.cfg.
survey_file	string	*survey_file* is the name of the user-created survey file that describes electrode locations and survey configuration.
conductivity_file	String or real number	*conductivity_file* specifies either the name of the conductivity file, or the word 'AVERAGE', 'Average', or 'average'. In mode 3 *conductivity_file* provides the starting conductivity distribution for the inversion. Users may use the conductivity file created during mesh generation or construct their own conductivity file to use as the starting model. If a real valued number is specified, E4D will use that value as the homogeneous starting conductivity. Alternatively, if *conductivity_file* is one of the forms of the word 'average' as described above, E4D computes the average apparent conductivity of each measurement in the survey file, and uses that conductivity as the homogeneous starting model value. Each apparent conductivity is weighted according the standard deviation of the corresponding measurement.

Variable	Type	Description
		In the absence of other information, it is generally recommended to use the average apparent bulk conductivity as the starting model. However, it may not be appropriate to use this approach in the presence of significant surface topography, because there is no analytic solution by which to compute the apparent conductivity when surface topography exists. In this case, E4D will use the average elevation of the surface nodes of the mesh as the surface elevation for the apparent conductivity computations, which may provide an unreasonable starting model value.
out_opts_file	string	*out_opts_file* specifies the name of the output options file. It is recommended to instruct E4D to produce the simulated data file, which provides a direct comparison between measured and simulated data. The simulated data file is overwritten by E4D each time the inverse solution is updated.
inv_opts_file	string	*inv_opts_file* specifies the name of the inversion options file, which provides E4D instructions concerning how to constrain the model, how to execute the inversion, and how to determine if the solution has converged. The inversion options file is discussion in detail in the next section.
ref_mod_file	string	*ref_mode_file* specifies the name of the conductivity file that is optionally used to constrain the inversion as specified in the inversion options file. If a reference model is not used to constrain the inversion, then *ref_mode_file* is not accessed by E4D, and therefore is not required to exist. However, a value for *ref_mode_file* must always be included in *e4d.inp* as a placeholder.

7.3 Mode 3 Inversion Options File

With the exception of the inversion options file, each of the files listed in *e4d.inp* for mode 3 are described in the documentation for modes 1 and 2. Those files have the same format and purpose in mode 3, and their descriptions and use are not repeated here. In this section we provide a detailed description of the format and function of the inverse options file.

The information provided by an ER survey is typically insufficient to uniquely determine the subsurface bulk conductivity distribution at the scale of the computational mesh. In order to produce a reasonable representation of the true subsurface conductivity, the inverse solution must be constrained by a priori information in addition to the information provided by the data. The information provided by the data is given implicitly in the survey file. The a priori solution constraints are provided in the inverse options file.

Often, the quality of the inverse solution is highly dependent upon the solution constraints supplied to the inversion algorithm. It is generally beneficial to provide as much information as possible to the inversion algorithm, which can provide marked improvements in imaging resolution. To this end, E4D is implemented with a flexible set of model constraints applied using the method of IRLS. The inversion option file provides E4D with solution constraints zone by zone, and tells E4D how those constraints should be implemented across zone boundaries. Each constraint is specified by two equations, a structural metric and a weighting function. The structural metric provides the constraint equation that, when minimized, imposes a particular structure in the conductivity distribution. The weighting function determines when, and how strongly, the structural metric should be imposed. As described below, E4D currently provides ten structural metrics and four weighting functions. Each is referenced by a unique integer value as shown in the forthcoming structural metric and weighting function tables. The general format of the inverse options is file is shown below.

Variable	Type	Description
n_reg_blocks	integer	Specifies the number of constraint blocks. A constraint block describes the solution constraints to be applied to a particular mesh zone. Each zone in the mesh should be supplied with at least one constraint block.
zone	integer	Specifies the zone number to which this constraint block applies. **This is the beginning of a constraint block.**
s_met wx wy wz	see description	*s_met* is an integer that indicates which structural metric to use for this constraint (see structural metric codes below). *wx wy* and *wz* are positive real values indicating the relative weighting of this structural metric in the x, y, and z directions respectively. Note: *wx*, *wy*, and *wz* are not used by every structural metric. In cases where they are not used, they are ignored by E4D in the inversion. However, they must be present in the inverse options file, regardless of which structural metric is used.
fw mn sd	see description	*fw* is an integer value indicating which re-weighting function to use with this structural metric (see weighting function codes below). *mn* is a real value indicating the mean of the re-weighting function *sd* is a real value indicating the standard deviation of the weighting function (see Figure A.1 - Figure A.4)
n_links l1 l2 ... ln_links	integer	*n_links* is the number of zones to which this zone is linked. *l1 l2 ... ln_links* are the zone numbers to which this zone is linked. Elements bounding the zone constrained by this block will be connected to this block (through the regularization constraints specified in this block) if the zones to which those elements belong are specified in the link list. A zone may not be linked to itself. In other words, each of the linked zones *l1 l2 ... ln_links* listed must be different from the variable *zone*.
v_ref	see description	*v_ref* specifies the reference value used for this constraint block. If *v_ref* is a real number, then *v_ref* is used as the reference value. If *v_ref* is specified as "*REF*" or "*Ref*" or "*ref*", then the reference model specified in *e4d.inp* is used to provide the reference values for each element in the zone constrained by this block. Not all structural metrics use a reference value. However, *v_ref* must always be specified in the inverse options file. If *v_ref* is not used by the structural metric specified by this constraint block, then it is ignored by E4D.

Variable	Type	Description
		Note: in mode 4, v_ref may also be specified as “PREF”, “Pref”, or “pref”, in which case the solution to the previous time-lapse inversion is used as the reference model for this block.
w_rel	real	*w_rel* is the relative weight applied to this constraint block. This parameter provides the capability to enforce some constraints more strongly than others. A value of 1.0 will suffice for most problems. **This is the end of a constraint block. There must be** ***n_reg_blocks*** **constraint blocks specified in the inversion options file.**
beta min_red beta_red	real	*beta* is the global constraint weighting value at the beginning of the inversion, and controls the importance the inversion places on enforcing the model constraints in comparison to fitting the data. This parameter is either held constant or automatically reduced (see *up_opt* below) by E4D during the inversion when required to reduce the misfit between measured and simulated data. (see the *E4D* theory guide) *min_red* is the minimum fractional decrease in the objective function between outer iterations that may occur before *beta* is reduced (see *E4D* theory guide). A conservative value for *min_red* is 0.05. Larger values of *min_red* will generally decrease time to convergence, but may also provide solutions that violate the model constraints more than what is necessary to appropriately fit the data. *beta_red* is the *beta* reduction factor. If the fractional decrease in the objective function between outer iterations is less than or equal to *min_beta*, then *beta* is reduced by a factor of *beta_red* for the next iteration. A conservative value for *beta_red* is 0.5.
chi_target	real	*chi_target* is the normalized chi-squared value at which the inversion is considered to have converged. In the absence of modeling errors and accurately specified data standard deviations, *chi_target* should reach a value of 1.0 at convergence. (see *E4D* theory guide)
miniter maxiter	integer	*miniter* is the minimum number of inner iterations (i.e. CGLS iterations) to execute before updating the solution (see E4D design guide, recommended value = 30) *maxiter* is the maximum number of inner iterations to execute before updating the solution (recommend value = 50) (see *E4D* design guide)
min_sig max_sig	real	*min_sig* and *max_sig* are respectively minimum and maximum conductivity values allowed by the inversion. Note that these values do not constrain the inversion, and are provided only as a safety mechanism to ensure the forward solutions remain stable. Maximum and minimum conductivity constraints can, and should, be specified as model constraints within the constraint block section of the inversion options file when necessary.

Variable	Type	Description
up_opt	integer	If *up_opt* = 1 a line search to estimate the optimum *beta* value is executed if *up_opt* = 2, no line search is executed and beta reduces as specified by *min_red* and beta_red if *up_opt = 3,* beta remains at its starting value, and the inversion converges when the reduction in the objective function is less than *min_red* or the target chi-squared value is reached, whichever comes first. **The beta line search has not been quality tested, therefore E4D defaults to up_opt = 2 when up_opt = 1 is specified.**
n_sfacs	integer	Specifies the number of line search scaling factors to test in order to estimate the optimal scaling factor. If *up_opt*=2, this value is not used by E4D, and should not be included in the inverse options file.
sfac1 ... sfac_n_sfacs	real	Line search scale factors to test during the line search. If up_opt = 2, these values are not used by E4D, and should not be included in the inverse file.
cflag cdev	see description	*cflag* is an integer value of 0 or 1. If *clag* = 1 then data outlier re-weighting is implemented. If *cflag* = 0 then all data will be used to constrain the inversion at every iteration. *cdev* is a positive real value specifying the outlier removal standard deviation. If the weighted residual error of any datum is greater than *cdev* standard deviations from the mean weighted residual error and *cflag* = 1, then that datum is not used to constrain the inversion in next iteration. (Recommended value = 3) Outlier conditions are checked at every iteration, so a particular datum may be removed for one iteration and included in the next, and vice-versa.

7.4 Structural Metric Codes

A structural metric is defined herein as an equation describing a relationship between the conductivity of a target element and 1) the conductivity of the target elements neighbor, 2) a specified reference conductivity value (see *v_ref* in the inverse options file format above), or 3) the conductivity of the corresponding element specified in the reference model file. When a particular structural metric is specified to a zone, E4D applies that metric to every element in that zone, and optionally to elements bounding that zone if so specified in the corresponding constraint block. E4D attempts to minimize each structural metric, which imposes some desired structure in the inverse solution (see *E4D* theory guide). E4D determines the conditions under which to apply the constraints according to the weighting function applied to each structural metric. Thus the behavior of each structural metric in terms of constraining the inversion is also dependent upon the corresponding weighting function for that metric. In the table below, we provide the equations defining each structural metric. In the next section, we provide the weighting functions, and follow-up with several examples of how structure metric/weighting function combinations can be used to provide the inversion with information concerning the subsurface conductivity structure. Details concerning the implementation of IRLS in E4D are provided in the E4D theory guide.

Structure metric code (*s_met*)	**Equation**
1	X = m_t - m_n
2	X = |m_t - m_n|
3	X = m_t - *v_ref*$_t$
4	X = |m_t - *v_ref*$_t$|
5	X = m_t - m_n (with anistropic weighting)
6	X = |m_t - m_n| (with anistropic weighting)

7	X = (m_t - *v_ref*$_t$) - (m_n - *v_ref*$_n$)
	
	X = Dm_t - Dm_n

8	X = |(m_t - *v_ref*$_t$) - (m_n - *v_ref*$_n$)|
	
	X = |Dm_t - Dm_n|

9	X = m_t - m_n (applied only at specified boundaries)
10	X = |m_t - m $_n$| (applied only at specified boundaries)
Variables: X = value of the structural metric, and the independent variable in each weighting function. m_t = log conductivity of the target element m_n = log conductivity of the target elements neighbor v_ref$_t$ = log conductivity of the target elements reference value, taken from either the constraint block or converted from the reference model file v_ref$_n$ = log conductivity of neighbors reference value, taken from either the constraint block or converted from the reference model file Dm_t = change in log conductivity of target element with respect to v_ref$_t$ Dm_n = change in log conductivity of neighbor element with respect to v_ref$_n$	

7.5 Weighting Function Codes

The primary purpose of the weighting function is to determine the conditions under which the corresponding structural metric should or should not be used to constrain the inversion. In essence, the weighting functions turn the structural metrics on and off, and determine how much weight should be placed on minimizing the structural metric in the transition between on and off. The independent variable for each weighting function is the value of the structural metric (i.e., the value X in the structural metric code table above). Each of the four weighting functions is based on the normal and cumulative normal distribution functions, and ranges between 0 and 1. If the weighting function evaluates to zero, then the structural metric is inactive and plays no role in constraining the inversion. If it evaluates to one, then the structural metric is fully active. The following table and figures show the equation and form of each weighting function.

Code	Equation
1	W_f=.5(1- erf((X-*mn*) / sqrt(2**sd*^{2}))) (see Figure 7.1)
2	W_f=.5(1- erf((X+*mn*) / sqrt(2**sd*^{2}))) (see Figure 7.2)
3	W_f=1-exp(-((X-*mn*)^2) / (2**sd*^{2})) (see Figure 7.3)
4	W_f=exp(-((X-*mn*)^2) / (2**sd*^{2})) (see Figure 7.4)
Variables: W_f = value of the weighting function where (0 <= W_f <= 1) X = value of the structural metric (see structural metric code table above) erf = error function *mn* = mean of the weighting function as specified in the constraint block (see inversion options file format and Figure 7.1-Figure 7.4 below). *sd* = standard deviation of the weighting function as specified in the constraint block (see inversion options file format and Figure 7.1-Figure 7.4 below).	

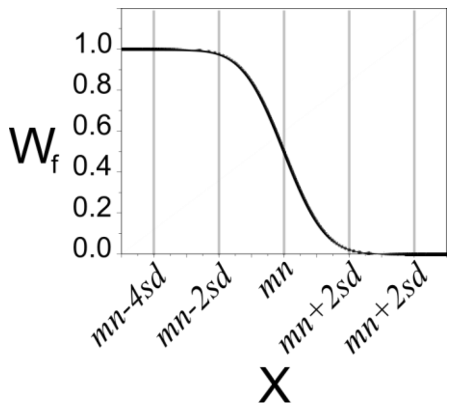


Figure 7.1. Weighting function 1 causes E4D to minimize X if X drops below *mn+2sd*, reaching full weight if X drops below *mn-2sd*.

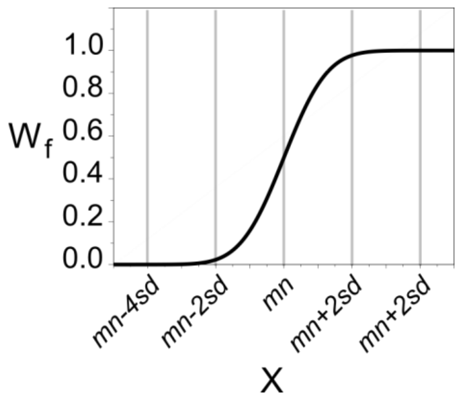


Figure 7.2. Weighting function 2 causes E4D to begin to minimize X if the value of X rises above *mn-2sd*, reaching full weight if X rises above *mn+2sd*.

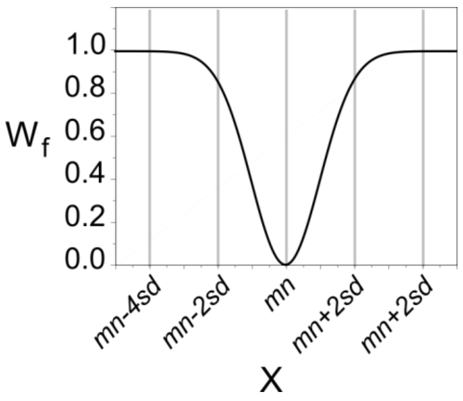


Figure 7.3. Weighting function 3 causes E4D to begin to minimize X if the value of X deviates from mn, reaching full weight if X deviates from mn by more than (approximately) 2sd.

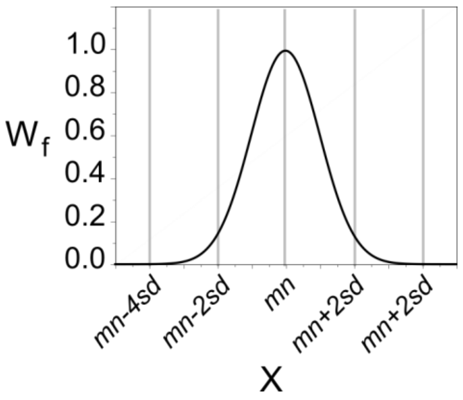


Figure 7.4. Weighting function 4 causes E4D to begin to minimize X as the values of X approaches mn, reaching full weight when X is equal to mn.

7.6 Example Constraints for Static Inversions

The combination of structural metric, weighting function, and custom mesh generation (i.e., incorporation of known conductivity boundaries) capabilities are intended to provide flexibility and creativity in applying solution constraints within E4D. The examples below show how structural metrics, weighting functions, and zone boundaries work together to inform E4D concerning known subsurface conductivity conditions. They do not present all of the possible constraint combinations, but are intended to provide users with enough familiarity to formulate their own problem specific constraints. It is important to note that similar constraints can often be implemented with different pairs of structural metrics and weighting functions.

7.6.1 Smoothness Constrained Inversion

Standard smoothness constraints are implemented by encouraging the inversion to enforce similarity between neighboring elements. For reasons that will become evident in the next example, we recommend using structural metric 2 and weighting function 1 (with a large mean value) to enforce smoothing constraints. Structural metric 1 increases as the absolute difference between the log conductivity of each element and its neighbor increases. Combined with structural metric 2, weighting function 1 will encourage the inversion to minimize the absolute log conductivity difference with full weight as long as that difference is greater than approximately *mn-2*sd*. If *mn* is larger than any reasonably expected absolute difference between any two elements, and *sd* is small, then weighting function 2 will apply full weight at each iteration. As described in the E4D theory guide, this is equivalent to standard L2-norm regularization weighting (e.g., Tikhonov regularization).

For example, suppose we have a mesh with three zones. Zone 2 has boundaries adjacent to both zones 1 and 3. We wish to constrain zone 2 with smoothness constraints, including smoothness constraints across the boundaries with zones 1 and 3. A constraint block that fulfills these requirements is given by:

<begin constraint block> this line is not included in the inverse options file

```
2                          this is the zone number constrained by this block
2  1.0  1.0  1.0           use structural metric 2, (x,y,z weighting values are not used with 2)
1   10   0.001             use weighting function 1 with  mean = 10 and  st. dev. of 0.001.
2   1  3                   apply the constraints across boundaries with zone 1 and 3
0.0                        this is the reference value, which is not used by structure metric 2
1.0                        this is the relative weight for this structure metric.
```

<end constraint block> this line is not included in the inverse options file

Smoothness constraints are demonstrated in tutorial 3.1 at the end of this chapter.

7.6.2 Smoothing with Sharp Boundaries (Blocky Inversion)

Several methods have been proposed that enable inversions to provide solutions with sharp internal boundaries. In general, most of these approaches are implemented by reducing the similarity constraints between neighboring elements as the difference in conductivity between those elements increases. This enables a 'fracture' to develop between elements that exhibit a relatively large difference in conductivity, and enables elements on either side of the 'fracture' to become similar, thereby providing a compact or blocky conductivity image. Such constraints can be implemented in E4D using the same constraint block described above for smoothness constrained inversions, but with a smaller mean in the weighting function. By doing so, the similarity constraint between neighboring elements is removed as the conductivity difference between those elements increases, depending on the values of *mn* and *sd* characterizing the weighting function. For example, consider the constraint block below.

<begin constraint block> this line is not included in the inverse options file

```
2                      this is the zone number constrained by this block
2  1.0  1.0  1.0       use structural metric 2, x,y,z weighting values are not used with 2
1   .5   0.01          use weighting function 1 with a mean = 0.5 and st. dev. = 0.01.
2   1  3               apply the constraints across 2 boundaries with zones 1 and 3
0.0                    this is the reference value, which is not used by structure metric 2
1.0                    this is the relative weight for this structure metric.
```

<end constraint block>

This constraint block will enforce similarity between neighboring elements as long as the absolute difference in log conductivity is less than 0.5-0.01. When the log conductivity difference between two elements exceeds 0.5-0.01, E4D will begin to reduce the similarity constraints between those elements, enabling the difference between them to further increase. If the log conductivity difference exceeds 0.5+.001, then the similarity constraints between those elements are removed in the next iteration, enabling the inversion to place a sharp contrast at the boundary without penalty. As described in the E4D theory guide, commonly used weighting functions published in the literature (e.g., compactness constraints and l^n-norm) can be closely approximated by modifying *mn*, *sd*, and *w_rel* (the relative weight) for each constraint block.

7.6.3 Minimum Conductivity

Reasonable limits on the minimum and maximum subsurface conductivity are often available for a given application. Inequality constraints, which enforce conductivity limits, can provide a valuable source of information to the inversion algorithm and significantly improve imaging resolution. Minimum conductivity constraints can be implemented in E4D using structural metric 3 and weighting function 1.

Structural metric 3 provides the relative difference between the log conductivity of each element and the log conductivity of a corresponding reference value, which is provided in the constraint block or computed from the values in the reference conductivity file. E4D minimizes structural metric 3 by moving the log conductivity of each element toward its reference value.

The minimum conductivity constraint is applied by instructing E4D to enforce the condition that the log conductivity of each element is always greater than or equal to the reference value. Using structural metric 3, this is accomplished by applying weighting function 1 with a mean of 0 and a relatively small standard deviation. For example, suppose zone 3 of a given mesh represents a body of surface water, and it is known the conductivity of the water is (depending on temperature) always greater than 0.01 S/m. A constraint block enforcing this condition is shown below.

<begin constraint block> this line is not included in the inverse options file

```
3                      this is the zone number constrained by this block
3  1.0  1.0  1.0       use structural metric 3, x,y,z weighting values are not used
1   0   0.05           use weighting function 1 with a mean = 0.0 and st. dev. = 0.05
```

0	do not apply this constraint across any boundaries
-4.61	this is the reference value, (i.e. log(0.01))
1.0	this is the relative weight for this structure metric.

<end constraint block>

The weighting function in this case provides full weight toward minimizing the structural metric if the structural metric drops below approximately *mn*-2**sd* = 0.0 - 2*(0.05) = -0.10. In other words, E4D will apply full weight toward moving the log conductivity to the reference value (thereby minimizing the structural metric), of any element whose log conductivity drops 0.10 S/m below the reference value. As the log conductivity increases, the weighting function will apply less weight toward minimizing the structural metric, until the constraint is removed (i.e. the weighting function is zero) at approximatey X=*mn*+2*sd* = 0.10 S/m, which means the log conductivity of the element is -4.61 + 0.10 s/m = -4.51 S/m.

7.6.4 Maximum Conductivity

Maximum conductivity constraints may be applied similarly to minimum conductivity constraints, except that weighting function 2 is used instead of weighting function 1. For example, suppose zone 3 of a given mesh represents a body of surface water, and it is known the conductivity of the water is (depending on temperature) always less than 0.01 S/m. A constraint block enforcing this condition is shown below.

<begin constraint block> this line is not included in the inverse options file

3	this is the zone number constrained by this block
3 1.0 1.0 1.0	use structural metric 3, the x,y,z weighting values are not used with 3
2 0 0.05	use weighting function 1 with mean = 0.5 and st. dev. of 0.001.
0	do not apply this constraint across any boundaries
-4.61	this is the reference value, (i.e. log(0.01))
1.0	this is the relative weight for this structure metric.

<end constraint block>

7.6.5 Known Value with Uncertainty

In some cases the bulk conductivity of the subsurface represented by a mesh zone may be known with some degree of certainty. Such information can be provided to E4D using structural metric 3 or 4 with weighting function 3, using a mean of zero and a standard deviation that represents the uncertainty in the reference conductivity value. In a manner analogous to the minimum and maximum conductivity constraints discussed above, weighting function 3 (with structural metric 3 or 4) will cause the inversion algorithm to move the log conductivity of an element toward the reference value with increasing weight as the deviation from the reference value increases, reaching maximum weight when the absolute value of the deviation reaches *mn*+2**sd* = 2**sd.*

7.7 Reporting

E4D reports difference aspects of inversion progress to the screen and to several different files. Text printed to the screen primarily provides information indicating the current task the inversion is executing. In batch run modes, screen text is printed to whatever is identified as standard output, which is typically a user specified file in most parallel computing queuing systems. The file *run_time.txt* records details concerning the run times of each iteration, and is updated as the inversion progresses. The file *e4d.log* provides all information necessary to re-run the inversion, summarizes the convergence progress of each iteration, and provides other information relevant to the inversion. E4D also writes a conductivity file after each inverse update, which enables users to visualize how the conductivity distribution is changing as the inversion progresses, and provides valuable insight concerning the performance of the model constraints. The starting model is written to the conductivity file *sigma.0*, and each subsequent iteration is written to the file *sigma.x*, where *x* specifies the iteration number. The simulated data file is also updated at each iteration if so specified in the output options file.

7.7.1 *e4d.log* File

As in mode 2, E4D prints summary information describing the input files, survey, mesh, and computational loading to the *e4d.log* in mode 3. E4D then prints information contained in the inverse options file, including constraining specifications from each constraint block, and general weighting and convergence specifications. E4D produces a summary of the current state of the inversion after each iteration. Shown below is example text from the *e4d.log* (produced from tutorial 3.1 at the end of the chapter).

<begin e4d.log> this line is not included in the file
.
.
.

```
********** CONVERGENCE STATISTICS AT STARTING MODEL *******************

    Phi_dat        Phi_Mod     Phi_Mod/Beta      Phi_Tot
     31530.         0.0000         0.0000         31530.

 Total Number of Constraint Eqs: 0000242717
 Block     # Constraints     % of Total Error     Error per Constraint
     1           157537          0.0000                    0.0000
     2            85180          0.0000                    0.0000

 Chi2 is currently 15.532       Target value is 1.0000
 Mean error is .61704
 RMS error is 3.9411
 Number of data culled is 0000040
 Prior to data culling ...
    Chi2 is   29.3
    Mean error is  0.439
    RMS error is   5.41
 ***********************************************************************

 ********** CONVERGENCE STATISTICS AFTER INVERSE UPDATE # 001 **********
```

```
    Phi_dat         Phi_Mod     Phi_Mod/Beta      Phi_Tot
     5311.6          2928.4         29.284         8240.0

 Total Number of Constraint Eqs: 0000242717
 Block     # Constraints     % of Total Error     Error per Constraint
     1            157537          81.344                  0.70122E-02
     2             85180          18.656                  0.29744E-02

 Chi2 is currently 2.5999       Target value is 1.0000
 Mean error is **********
 RMS error is 1.6124
 Number of data culled is 0000027
 Prior to data culling ...
    Chi2 is   4.59
    Mean error is -0.486
    RMS error is   2.14
 **********************************************************************
 Decrease in objective function is:   0.738664746
 Decrease in objective function is sufficient to continue at beta =
100.000000

.
.
.
```

<begin e4d.log> this line is not included in the file

The iteration summary sections shown above provide the convergence statistics of the starting model and the updated model after iteration 1. Similar reports are provided for each subsequent iteration. Each variable is described below.

Variable	Description
Phi_dat	This is the data term of the objective function, also known as the sum of squared weighted data residual errors, where a data residual error is the difference between an observed and simulated measurement divided by the standard deviation for that measurement. (see E4D theory guide for details).
Phi_Mod	This is the model term of the objective function, or the sum of the squared model residual errors, multiplied by the current beta value. A model residual error is the value of the structure metric (i.e the value X in the structure metric tables). Phi_Mod is a measure of the total difference between the actual model structure and the structure specified via minimization of the constraint equations (see E4D theory guide).
Phi_Mod/Beta	This is a relative measure of the structural complexity in the current inverse solution, where complexity is defined according to Phi_Mod (see E4D theory guide).
Phi_Tot	Current value of the total objective function, which is equal to *Phi_dat* + *Phi_Mod*. E4D attempts to minimize this value during the inversion.
Total # of constraint equations	Total number of constraint equations that E4D attempts to minimize during the inversion, subject to appropriately fitting the data.
Block	Specifies a constraint block listed in the inversion options file.
# Constraints	Total number of constraints added to the inversion by this constraint block.
% of Total Error	% of total model residual error attributable to this constraint block. E4D will generally focus on minimizing constraint blocks with larger portion of the total error. This metric is

Variable	Description
	useful for diagnosing which constraints are influencing the inversion.
Error per Constraint	Average model residual error for this constraint block.
Chi2	This is the chi-squared value of the current iteration after data culling. The chi-squared value is equal to *Phi_dat/(nd-nc)* where *nd* is the total number of measurements specified in the survey file and *nc* is the number of measurements that were culled during the current iteration.
Target Value	Target chi-squared value as specified in the inverse options file. If data standard deviations are appropriately specified, then the chi-squared value should be equal to one at convergence. However, since true data (and modeling) errors are often unknown, the appropriate target value may not be equal to one, even if data are weighted appropriately relative to one another. Assuming data standard deviations are appropriate relative to other data in the survey file, an appropriate target chi-squared value may be determined by viewing the solution at each iteration. Overly heterogeneous solutions are indicative of over-fit data (i.e., the target chi-squared value is too small), and overly smooth solution are indicative of under-fit solutions (i.e. the target chi-squared value is too large).
Mean Error	This is the mean of the weighted data residual distribution, and should generally be near zero at convergence for an unbiased solution.
RMS Error	The RMS error is the square root of the chi-squared value.
# Data Culled	Total number of data whose residual errors fall outside of the culling limits specified in the inversion options file. These data are temporarily given very large standard deviations, effectively eliminating their influence in inverse solution update. Each culled data point may be included in subsequent iterations, and it is not uncommon for the number of culled data to decrease as the inversion progresses.
Decrease in Objective Function	Fractional decrease in the objective function from the previous iteration, given by (Phi_Tot_n - Phi_Tot_{n-1})/Phi_Tot_n where *n* represents the iteration number. E4D will decrease *beta* for the next iteration if this value is less than the threshold specified in the inverse options file (i.e. *min_red*).

7.8 ER Inversion Example 3.1: Imaging Two Blocks

7.8.1 Overview

This tutorial builds upon the two_blocks examples presented in the mesh generation and forward modeling tutorials. In tutorial 1.1, we constructed a computational mesh composed of two blocks embedded within an otherwise homogeneous halfspace. The conductivity of the halfspace was 0.002 S/m, and the conductivity of the left and right blocks were 0.02 S/m and 0.2 S/m, respectively, as shown in Figure 7.5. This represents the true model and the target for the synthetic ERT imaging in this tutorial.

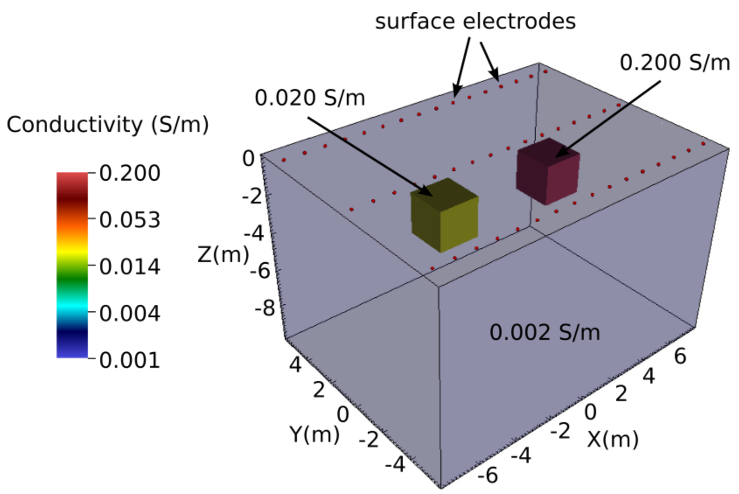


Figure 7.5. True subsurface conductivity distribution

In tutorial 2.1, we used E4D forward modeling capabilities to generate a synthetic survey file named *two_blocks.sig.srv*. To simulate field conditions, we added random noise to the transfer resistance measurements in *two_blocks.sig.srv* , adjusted the standard deviations accordingly, and renamed the survey file to *two_blocks.srv*, which is the survey file used in this tutorial (note: noise addition was done outside of E4D and was not shown in tutorial 2.1).

To further simulate realistic field conditions, we assume in this tutorial that the locations and dimensions of subsurface anomalies are unknown, and build a mesh consistent with that assumption. That is, for the inversion problem, we build a mesh with no implicit internal boundaries. To do so, we simply modify the mesh configuration file *two_blocks.cfg* by removing the subsurface boundary definitions for the two blocks. This results in two zones in the inversion mesh; a foreground zone containing the electrodes and target imaging volume (zone 1), and a zone of elements extending to the mesh boundaries (zone 2). The modified mesh configuration file used to build the mesh is named *two_blocks_inv.cfg***,** and is included in the directory *<e4d_dir>/tutorial/mode_3/two_blocks* of the E4D distribution. Users are encouraged to review tutorials 1.1 and 2.1 and build the mesh for this example using *two_block_inv.cfg*, thereby generating the mesh files *two_blocks_inv.1.node***,** *two_blocks_inv.1.ele***,** *two_blocks_inv.1.face***,** *two_blocks_inv.1.neigh*, and *two_blocks_inv.trn*. A cut-out view of the resulting mesh is shown in Figure 7.6.

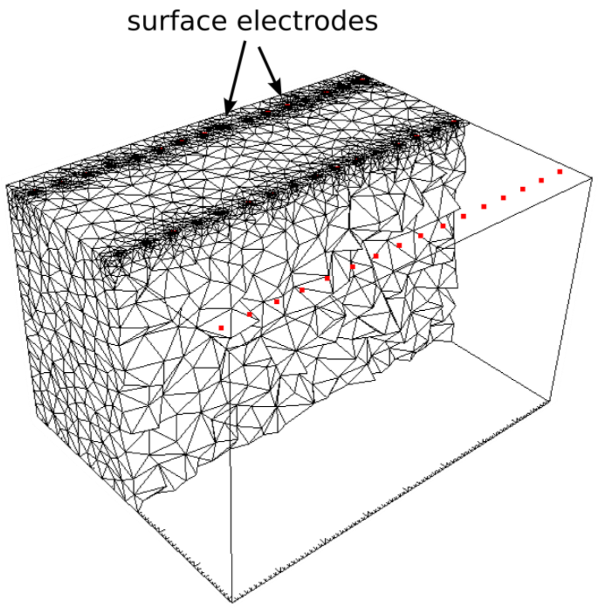


Figure 7.6. Cut-out view of the inversion mesh, which is refined around electrodes anddoes not include the block boundaries, and is coarser than the forward modeling mesh (Figure 6.2).

With the exception of the mesh files, the directory for this tutorial *<e4d_dir>/tutorial/mode_3/two_blocks* contains all of the files necessary to execute the inversions demonstrated in this example. This includes the four inversion options files *two_blocks_1.inv*, *two_blocks_2.inv***,** *two_blocks_3.inv*, and *two_blocks_4.inv***.** Each inversion option file builds upon the previous by adding constraint blocks that represent some additional a priori information about the subsurface conductivity, thereby providing improved resolution with each subsequent inversion.

7.8.2 Smoothness Constrained Inversion

Smoothness constraints or similarity constraints are the most common constraints applied in ERT inversion problems, and are based on the hypothesis that neighboring elements are likely to have similar conductivity values. According to Occam's Razor, in the absence of constraining information other than transfer resistance measurements, the inverse solution that honors the measurements and has the least amount of heterogeneity is the most likely solution. Hence, smoothness constrained inversions are also called Occam's inversions. In E4D, smoothness constraints are specified by choosing a structural metric and weighting function that encourages the inversion to minimize the difference between neighboring elements, regardless of the conductivity values of those elements. Structure metric 2 provides the absolute difference between the log conductivity values of neighboring elements, and is the recommended metric to use for implementing smoothness constraints. Weighting function 1 will apply full weight to structure metric 2 if a large value is specified for the weighting function mean, so that the value of the weighting function never reduces from unity for reasonably expected conductivity differences between neighboring elements.

In this example, we implement smoothness constraints as described above. Below we provide the *e4d.inp* file and the inverse options file for discussion. These files are also included in the directory *<e4d_dir>/tutorial/mode_3/two_blocks*.

7.8.2.1 E4D Run File: *e4d.inp*

<begin e4d.inp> this line is not included in the file

```
3                          run mode
two_blocks_inv.1.node      mesh node file name
two_blocks.srv             survey file name
'average'                  use the average apparent conductivity for the starting model
two_blocks.out             output options file
two_blocks_1.inv           inverse options file
none                       reference model file (no reference model specified two_blocks.1.inv)
```

Notes: The mesh files and survey file are created as described in the Overview section. The inversion options file is shown below.

<end e4d.inp> this line is not included in the file

7.8.2.2 Inversion Options File: two_blocks_1.inv

<begin two_blocks_1.inv> this line is not included in the file

```
2                      # of constraint blocks, this file implements
1                      constrain zone 1 with this block
2 1.0 1.0 1.0          structural metric 2 (wx wy wz ignored)
1 10.0 .01             weight func. 1 with mean of 10 and small s.d.
1 2                    one link to zone 2
0.0                    reference value (not used)
1.0                    relative weight
2                      constrain zone 2 with this block
2 1.0 1.0 1.0          structural metric 2 (wx wy wz ignored)
1 10 0.01              weight func. 1 with mean of 10 and small s.d.
0                      no links (already linked to zone 1 above)
0.0                    reference value (not used)
1.0                    relative weight

100 0.25 0.5           beta min_red beta_red
1.0                    chi_targ
30 50                  miniter maxiter
0.00001 1.0            minsig maxsig
2                      up_opt
1 3.0                  cflag cdev
```

Notes: This inversion options file implements an Occam's type inversion by specifying similarity constraints between neighboring elements (structural metric 2) that are not removed unless the difference between neighbors is extreme. That is, the weight on the similarity constraints will not be reduced unless the absolute difference in log conductivity between neighbors exceeds 10, as specified by the weighting function. Zone 1 is linked to zone 2, which specifies that similarity constraints should be applied at the zone 1/zone 2 boundary. Since zone 1 is linked to zone 2, it is not necessary to link zone 2 to zone 1. Hence, zone 2 is not linked to another zone.

<end two_blocks_1.inv> this line is not included in the file

7.8.2.3 Running the Inversion

The inversion was executed for this example on 7 cores (1 master and 6 slave) using the command:

<e4d_dir>/bin/mpirun -np 7 <e4d_dir/bin/e4d>.

Or, if *mpirun* and *e4d* are in the exectuable path:

mpirun -np 7 e4d.

Upon execution, E4D completes the inversion and reports progress to the following files:

e4d.log:	This is primary log file discussed in the next section.
run_time.txt:	This file contains run-time information and is updated as the inversion progresses
two_blocks.dpd:	This is the predicted data file specified in the output option file two_blocks.out
sigma.*	Estimated conductivity at the *th inverse iteration.

Status updates are also printed to screen. With the specified mesh and settings, the inversion converged in seven iterations, requiring approximately 10 minutes to complete on a laptop computer with 8 processing cores and 4 Gb of RAM.

7.8.2.4 E4D Log File: *e4d.log*

The E4D log file records information describing the inversion parameters and performance, and is updated as the inversion progresses. Users are encouraged to run the inversion as described above, and review the log file. The first block of the log summarizes mesh, survey, and inversion information, and is relatively self-explanatory. For the sake of discussion, we provide excerpts from the log file below, including the iteration records for several iterations.

Iteration 0: Starting model

<begin e4d.log> this line is not included in the file

```
.
.
.
********** CONVERGENCE STATISTICS AT STARTING MODEL *******************
Phi_dat      Phi_Mod      Phi_Mod/Beta      Phi_Tot
31530.       0.0000       0.0000            31530.
Total Number of Constraint Eqs:      0000242717
Block      # Constraints      % of Total Error      Error per Constraint
1               157537             0.0000                  0.0000
2                85180             0.0000                  0.0000
Chi2 is currently 15.53                  Target value is 1.0000
Mean error is 0.6170
RMS error is 3.941
Number of data culled is 0000040
Prior to data culling ...
   Chi2 is 29.31
```

```
    Mean error is 0.4395
    RMS error is 5.414
*******************************************************************************
.
.
.
```
<end e4d.log>   this line is not included in the file

The starting model is homogeneous with a conductivity of 0.00219 S/m, which is the average apparent conductivity computed by E4D, and is reported in *e4d.log (not shown)***.** As shown above, the data norm of the objective function at the starting model is 31350, which reflects the weighted misfit between observed and simulated as described in the mode 3 documentation. The model norm of the objective function is zero because the starting model is homogeneous, causing the constraint equations specified for each of the constraint blocks to be perfectly minimized in this case. Block 1 contains more constraint equations than block 2 because there are more elements in block 1 than block 2. There are 40 data points culled, meaning that 40 of the weighted transfer resistance residuals are outside of the specified data culling bounds, which is three standard deviations from the mean weighted residual in this case. Without the 40 culled data points, the chi-squared value at the starting model is 15.53. The target value is 1.00 as specified in the inverse options file. Convergence statistics prior to data culling are also printed as a reference.

Iteration 1

<begin e4d.log>   this line is not included in the file

```
.
.
.

********** CONVERGENCE STATISTICS AFTER INVERSE UPDATE # 001 **********
Phi_dat        Phi_Mod        Phi_Mod/Beta        Phi_Tot
5311.6         2928.4         29.284              8240.0
Total Number of Constraint Eqs:        0000242717
Block        #Constraints        % of Total Error        Error per Constraint
1                  157537                81.344                0.70122E-02
2                   85180                18.656                0.29744E-02
Chi2 is currently 2.600                   Target value is 1.0000
Mean error is -.3537
RMS error is 1.612
Number of data culled is 0000027
Prior to data culling ...
    Chi2 is 4.594
    Mean error is -.4862
    RMS error is 2.143
***************************************************************************************
************
Decrease in objective function is: 0.7387
Decrease in objective function is sufficient to continue at beta = 100.0
.
.
.
```
<end e4d.log>   this line is not included in the file

After the first iteration, the chi-squared value dropped by approximately 74% from the starting model, indicating a significant improvement in the data fit. The constraint block errors indicate the degree to which the model constraints are honored. Zone 1 has a greater degree of misfit, indicating a greater degree of heterogeneity. This is expected, since the data are primarily sensitive the conductivity in zone 1. The number of culled data has dropped from 40 at the baseline model to 27 after iteration 1. The objective function decreased by more than 0.25 (or 25%), so the beta value was not reduced.

Iteration 6

<begin e4d.log> this line is not included in the file

```
.
.
.

********** CONVERGENCE STATISTICS AFTER INVERSE UPDATE # 006**********
Phi_dat      Phi_Mod      Phi_Mod/Beta      Phi_Tot
2364.6       1853.9       74.155            4218.4
Total Number of Constraint Eqs:      0000242717
Block      #Constraints      % of Total Error      Error per Constraint
1              157537            82.813              0.10484E-01
2               85180            17.187              0.40242E-02
Chi2 is currently 1.159              Target value is 1.0000
Mean error is -.5535
RMS error is 1.077
Number of data culled is 0000030
Prior to data culling ...
   Chi2 is 2.23
   Mean error is -.6552
   RMS error is 1.493
*************************************************************************************
************
Decrease in objective function is: 0.3157E-02
Solution did not converge at beta = 25.00
Decreasing beta by 0.5000   from  25.00  to 12.50
.
.
.
```

<end e4d.log> this line is not included in the file

The inversion required 7 iterations to converge. Iteration 6 is shown above. At iteration 6, the chi-squared value is near the target value. The Phi_Mod/Beta value increased from approximately 29 to 74 between iteration 1 and iteration 6, indicating increased heterogeneity in the model. The objective function decreased by less than 0.25 during this iteration, so the beta value decreased for the next iteration.

7.8.2.5 Inversion Image

Figure 7.7 shows the final solution for the smoothness constrained inversion. The visualization was produced using VisIt. The location of both blocks are clearly evident, although both are smoothed, or blurred, as expected. The relative conductive magnitudes of each block are also evident.

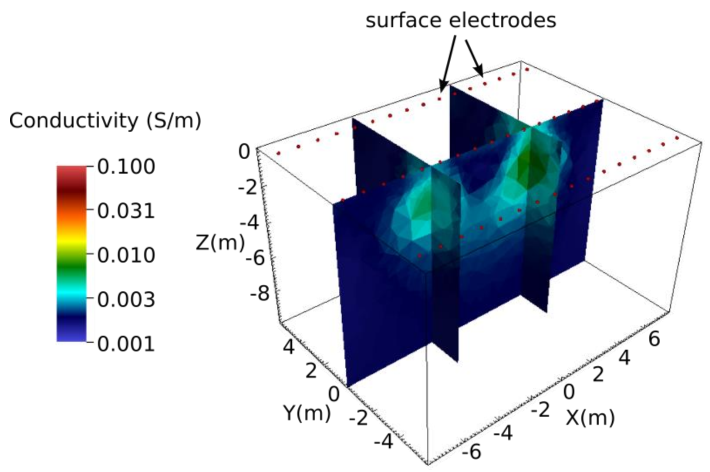


Figure 7.7. VisIt visualization of the smoothness constrained inversion results

7.8.3 Smoothness Constrained Inversion + Minimum Conductivity Constraint

In this example, we assume it is known that the subsurface conductivity is greater than or equal to 0.002 S/m within the imaging zone, and we apply an additional constraint block instruction in E4D to enforce that condition in the inversion. The inversion options file with the new constraint is included with the E4D distribution as *<e4d_dir>/tutorial/mode_3/two_blocks/two_blocks_2.inv*, and is shown below for discussion. To use the constraints specified in *two_blocks_2.inv*, simply modify the *e4d.inp* file listed above so that *two_blocks_2.inv* is specified on the inversion options file line instead of *two_blocks_1.inv*.

7.8.3.1 Inversion Options File: *two_blocks_2.inv*

<begin two_blocks_2.inv> this line is not included in the file

```
3                    # of constraint blocks, this file implements
1                    constrain zone 1 with this block
2 1.0 1.0 1.0        structural metric 2 (wx wy wz ignored)
1 10.0 .01           weight func. 1 with mean of 10 and small s.d.
1 2                  one link to zone 2
0.0                  reference value (not used)
1.0                  relative weight
2                    constrain zone 2
2 1.0 1.0 1.0        structural metric 2 (wx wy wz ignored)
1 10 0.01            weight func. 1 with mean of 10 and small s.d.
0                    no links (already linked to zone 1 above)
0.0                  reference value (not used)
1.0                  relative weight
1                    constrain zone 1 with this block
3 1.0 1.0 1.0        structural metric 3 (wx wy wz ignored)
1 0 .1               weight func. 1 with mean of 0.0 and s.d. of 0.1
0                    no links
-6.22                reference value (log(0.002) S/m)
2                    relative weight
```

```
100 0.25 0.5                    beta min_red beta_red
1.0                             chi_targ
30 50                           miniter maxiter
0.00001 1.0                     minsig maxsig
2                               up_opt
1 3.0                           cflag cdev
```

Notes: This inversion options file implements an Occam's type inversion by specifying similarity constraints between neighboring elements (structural metric 2) that are not removed unless the difference between neighbors is extreme. That is, the weight on the similarity constraints will not be reduced unless the absolute difference in log conductivity between neighbors exceeds 10, as specified by the weighting function. Zone 1 is linked to zone 2, which specifies that similarity constraints should be applied at the zone 1/zone 2 boundary. Since zone 1 is linked to zone 2, it is not necessary to link zone 2 to zone 1. Hence, zone 2 is not linked to another zone.

The third constraint block implements a minimum conductivity constraint. Weighting function 1, having a mean of zero, turns the constraint on if the value of the structural metric drops below zero, which occurs if the log conductivity drops below the reference value of -6.22 (i.e., log(.002)). When the constraint is turned on, the inversion minimizes the structural metric by moving the conductivity toward the reference value. A relative weight of 2 is specified to ensure the constraint is enforced.

<end two_blocks_2.inv> this line is not included in the file

7.8.3.2 Running the Inversion

The inversion is executed for this example on 7 cores (1 master and 6 slave) using the command:

<e4d_dir>/bin/mpirun -np 7 <e4d_dir/bin/e4d>.

Or, if *mpirun* and *e4d* are in the exectuable path:

mpirun -np 7 e4d.

Upon execution, E4D completes the inversion, and reports progress to the following files:

e4d.log:	This is primary log file discussed in the next section.
run_time.txt:	This file contains run-time information and is updated as the inversion progresses
two_blocks.dpd:	This is the predicted data file specified in the output option file *two_blocks.out*.
sigma.*	Estimated conductivity at the *th inverse iteration.

Status updates are also printed to screen. With the specified mesh and settings, the inversion converged in seven iterations, requiring approximately 10 minutes to complete on a laptop computer with 8 processing cores.

7.8.3.3 E4D Log File: *e4d.log*

The convergence for this inversion (and the log file) is similar to the previous inversion, except that the model constraint errors associated with the minimum conductivity constraint are also reported. Users are encouraged to run the inversion and review the log file for details.

7.8.3.4 Inversion Image

Figure 7.8 shows the final solution for the smoothness constrained inversion. The visualization was produced using VisIt. There is a slight, but perceptible improvement in resolution over the inversion without a minimum conductivity constraint as shown in the previous example. In particular, the conductivity of each block is slightly increased toward the true value, and smoothing between the blocks is slightly decreased.

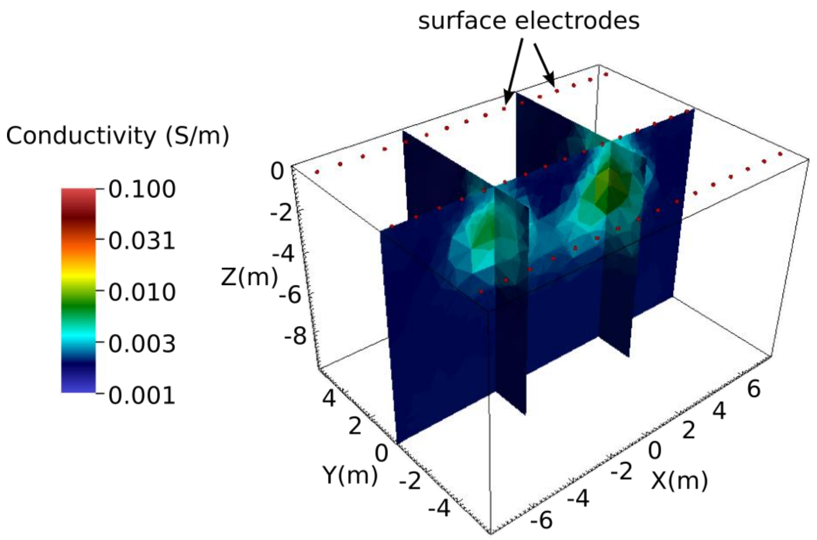


Figure 7.8. Results of the smoothness constrained inversion with a minimum conductivity constraint provides a slight improvement in resolution over the inversion without the minimum conductivity constraint (Figure 7.7).

7.8.4 Blocky Inversion

In this example, we assume it is known that, in addition to the minimum conductivity specified in the previous example, the boundaries between subsurface conductivity anomalies are discontinuous, or equivalently, that the conductivity structure is blocky. Because this assumption is consistent with the true model, we expect that providing this information to the inversion will improve imaging resolution. Discontinuous (or blocky) constraints are implemented by instructing the inversion to reduce or remove similarity constraints between neighboring elements that develop a relatively large contrast in conductivity. If the starting model is homogeneous, such contrasts will occur only if demanded by the data. In contrast to similarity constraints that are applied to heterogeneous subsurface, it is possible for discontinuity constraints to obtain a state (through the IRLS method) that is consistent with the subsurface, and thus, with the data. Under such conditions, it is possible for the inversion to converge to a solution that honors both the data and the model constraints, such that it may not be necessary to reduce the beta value in order to fit the data. Such is the case in this example.

In this example, discontinuity constraints are implemented by simply reducing the mean of the weighting function given in the first two constraint blocks (one for zone 1 and one for zone 2) of the previous inversions. By doing so, the constraints are turned off (i.e., given zero weight) when the absolute log conductivity difference between neighboring elements exceeds the mean value of the weighting function. The inversion options file shown below is provided in <*e4d_dir*>*/tutorial/mode_3/two_blocks/two_blocks_3.inv.*

7.8.4.1 Inversion Options File: *two_blocks_3.inv*

<begin two_blocks_3.inv> this line is not included in the file

```
3                       # of constraint blocks, this file implements
1                       constrain zone 1 with this block
2 1.0 1.0 1.0           structural metric 2 (wx wy wz ignored)
1  0.1  .05             weight func. 1 with mean of 1.0 and s.d. of 0.05
1 2                     one link to zone 2
0.0                     reference value (not used)
1.0                     relative weight
2                       constrain zone 2
2 1.0 1.0 1.0           structural metric 2 (wx wy wz ignored)
1  0.1  0.05            weight func. 1 with mean of 0.1 and s.d. of 0.05
0                       no links (already linked to zone 1 above)
0.0                     reference value (not used)
1.0                     relative weight
1                       constrain zone 1 with this block
3 1.0 1.0 1.0           structural metric 3 (wx wy wz ignored)
1 0 .1                  weight func. 1 with mean of 0.0 and s.d. of 0.1
0                       no links
-6.22                   reference value (log(0.002) S/m)
2                       relative weight

200 0.25 0.5            beta min_red beta_red
1.0                     chi_targ
30 50                   miniter maxiter
0.00001 1.0             minsig maxsig
2                       up_opt
1 3.0                   cflag cdev
```

Notes: This inversion options file implements a blocky inversion by enforcing similarity constraints between neighboring elements (structural metric 2) only if the absolute difference in log conductivity between elements is (approximately) less than the 0.1, which is the mean of the weighting function. With a standard deviation of 0.05 (i.e., one half of the mean), the weighting function transition between one and zero is relatively smooth. Zone 1 is linked to zone 2, which specifies that similarity constraints should be applied at the zone 1/zone 2 boundary. Since zone 1 is linked to zone 2, it is not necessary to link zone 2 to zone 1. Hence, zone 2 is not linked to another zone.

The third constraint block implements a minimum conductivity constraint. Weighting function 1, having a mean of zero, turns the constraint on if the value of the structural metric drops below zero, which occurs if the conductivity drops below the reference value of -6.22 (i.e., log(.002)). When the constraint is turned on, the inversion minimizes the structural metric by moving the conductivity toward the reference value (in this case). A relative weight of 2 is specified to ensure the constraint is enforced.

<end two_blocks_3.inv> this line is not included in the file

7.8.4.2 Running the Inversion

The inversion is executed for this example on 7 cores (1 master and 6 slave) using the command:

<e4d_dir>/bin/mpirun -np 7 <e4d_dir/bin/e4d>.

Or, if *mpirun* and *e4d* are in the exectuable path:

mpirun -np 7 e4d.

Upon execution, E4D completes the inversion and reports progress to the following files:

e4d.log:	This is primary log file discussed in the next section.
run_time.txt:	This file contains run-time information and is updated as the inversion progresses
two_blocks.dpd:	This is the predicted data file specified in the output option file *two_blocks.out***.** I
sigma.*	Estimated conductivity at the *th inverse iteration.

Status updates are also printed to screen. With the specified mesh and settings, the inversion converged in four iterations, requiring approximately 5 minutes to complete on a laptop computer with 8 processing cores.

7.8.4.3 E4D Log File: *e4d.log*

Users are encouraged to run the inversion as specified above (using from 2 to 49 cores), and review the *e4d.log* file. The inversion converges in four iterations and never reduces the starting beta value. In addition, both the data and model constraint misfits decrease with each successive iteration after iteration 1, demonstrating the agreement between the data and model constraints in this case.

7.8.4.4 Inversion Image

The blocky inversion image is shown n Figure 7.9, and displays a marked improvement in imaging accuracy over the previous inversions.

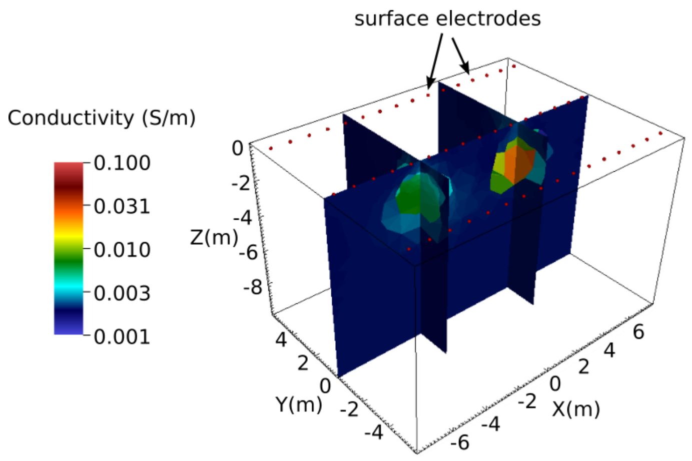


Figure 7.9. Blocky inversion results display a significant improvement in resolution and converge in approximately 50% of the time required for the smoothness constrained inversions. This demonstrates the value of providing the inversion with prior information through the model constraints.

7.8.5 Discrete Value Inversion

In this example, the inversion is constrained to construct a blocky conductivity structure with only three conductivity values: 0.002 S/m, 0.02 S/m, and 0.2 S/m. There are five constraint blocks. The first two blocks impose blocky constraints in zones 1 and 2, respectively, and the last three impose the discrete value constraints in zone 1. The discrete value constraints use structural metric 3 with weighting function 4. Structural metric 3 provides the log conductivity of an element minus a specified reference value. Weighting function 4 increases as the value of the structural metric approaches the mean of the weighting function. In this case, the mean is zero, so the weighting function increases as the log conductivity of an element approaches the reference value. This effectively causes the inversion to 'snap' the conductivity of elements to one of the reference values specified in the structural metric blocks, depending which value best fits the data. The inversion options file used to implement the constraints is shown below, and is provided with the E4D distribution in *<e4d_dir>/tutorial/mode_3/two_blocks/two_blocks_4.inv*.

7.8.5.1 Inversion Options File: *two_blocks_4.inv*

<begin two_blocks_4.inv> this line is not included in the file

```
5                    # of constraint blocks, this file implements

1                    constrain zone 1 with this block
2 1.0 1.0 1.0        structural metric 2 (wx wy wz ignored)
1  0.2  .01          weight func. 1 with mean of 0.2 and s.d. of 0.01
1 2                  one link to zone 2
0.0                  reference value (not used)
1.0                  relative weight

2                    constrain zone 2
2 1.0 1.0 1.0        structural metric 2 (wx wy wz ignored)
1  0.1  0.01         weight func. 1 with mean of 0.1 and s.d. of 0.01
```

```
0                         no links (already linked to zone 1 above)
0.0                       reference value (not used)
1.0                       relative weight

1                         constrain zone 1 with this block
3 1.0 1.0 1.0             structural metric 3 (wx wy wz ignored)
4 0.0 0.5                 weight func. 1 with mean of 0.0 and s.d. of 0.5
0                         no links
-6.22                     reference value (log(0.002) S/m)
1                         relative weight

1                         constrain zone 1 with this block
3 1.0 1.0 1.0             structural metric 3 (wx wy wz ignored)
4 0.0 0.5                 weight func. 1 with mean of 0.0 and s.d. of 0.5
0                         no links
-3.90                     reference value (log(0.02) S/m)
1                         relative weight

1                         constrain zone 1 with this block
3 1.0 1.0 1.0             structural metric 3 (wx wy wz ignored)
4 0.0 0.5                 weight func. 1 with mean of 0.0 and s.d. of 0.5
0                         no links
-1.61                     reference value (log(0.2) S/m)
1                         relative weight

50 0.1 0.75               beta min_red beta_red
1.0                       chi_targ
30 50                     miniter maxiter
0.00001 1.0               minsig maxsig
2                         up_opt
1 3.0                     cflag cdev
```

Notes: This inversion options file implements a blocky inversion by enforcing similarity constraints between neighboring elements (structural metric 2) only if the absolute log conductivity difference between elements is approximately less than 0.2, which is the mean of the weighting function. With a standard deviation of 0.01, the weighting function transition between one and zero is relatively abrupt. Zone 1 is linked to zone 2, which specifies that similarity constraints should be applied at the zone 1/zone 2 boundary. Since zone 1 is linked to zone 2, it is not necessary to link zone 2 to zone 1. Hence, zone 2 is not linked to another zone.

The last 3 constraint blocks provide discrete value constraints, which encourage the inversion to 'snap' the log conductivity of an element to a reference value if the log conductivity of that elements is near the reference value, depending on the value of the weighting function standard deviation. The reference values in this example are log(.002), log(.020), and log(.200).

<end two_blocks_4.inv> this line is not included in the file

7.8.5.2 Running the Inversion

The inversion is executed for this example on 7 cores (1 master and 6 slave) using the command:

<e4d_dir>/bin/mpirun -np 7 <e4d_dir/bin/e4d>.

Or, if *mpirun* and *e4d* are in the exectuable path:

mpirun -np 7 e4d.

Upon execution, E4D completes the inversion and reports progress to the following files:

e4d.log:	This is primary log file discussed in the next section.
run_time.txt:	This file contains run-time information and is updated as the inversion progresses
two_blocks.dpd:	This is the predicted data file specified in the output option file *two_blocks.out*.
sigma.*	Estimated conductivity at the *th inverse iteration.

Status updates are also printed to screen. With the specified mesh and settings, the inversion converged in four iterations, requiring approximately 5 minutes to complete on a laptop computer with 8 processing cores.

7.8.5.3 Inversion Image

The inversion results are shown in Figure 7.10. As illustrated, the inversion was able to fit the data to the target chi-squared value using only two conductivity values (0.002 and 0.02 S/m), with the exception of a few elements in the 0.2 S/m block. The block boundaries are well recovered in comparison to the smooth inversion, which demonstrates the value of including prior information in the form of model constraints in terms of improving resolution.

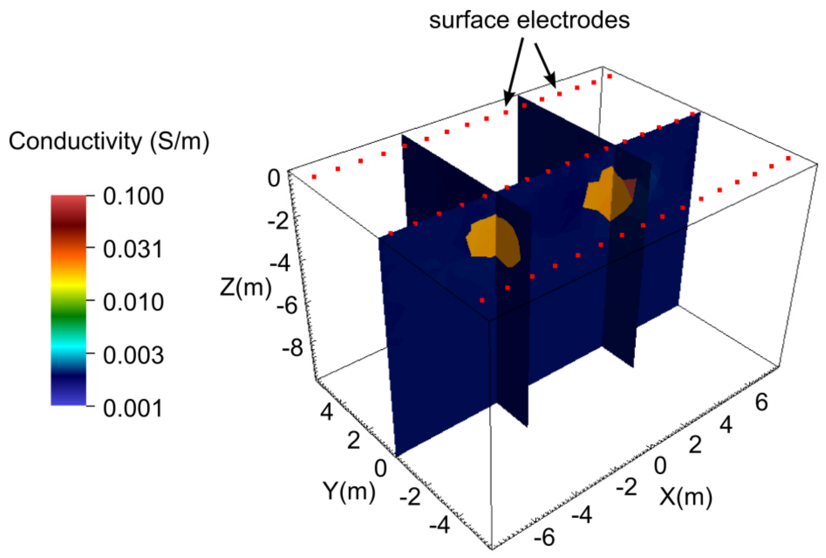


Figure 7.10. Constrained value inversion, whereby the inversion is constrained to produce a solution with only 3 conductivity values (0.002, 0.02, and 0.2 S/m).

7.9 ER Inversion Example 3.2: Imaging in the Presence of Buried Metal

7.9.1 Overview

The objective of this example is to demonstrate: 1) imaging in the presence of buried metallic objects with known position and dimension, 2) the generation of a custom conductivity distribution outside of E4D, and 3) anisotropic similarity constraints, which encourage conductivity continuity in a user-defined direction. We build upon mesh generation tutorial 1.2, whereby a mesh was generated that contains a buried metallic box, sheet, and line. Users are encouraged to review example 2.2 before proceeding with this tutorial. We begin by generating the synthetic survey file, which involves 1) defining the conceptual model (i.e., the actual or target conductivity distribution), 2) generating a refined computational mesh to accurately represent the conceptual model, 3) generating a customized conductivity file, and 4) running the forward simulation to generate the synthetic survey file. We then demonstrate two inversions with different inversion options; the first uses standard smoothness constraints, and the second uses blocky constraints with anisotropic weighting and inequality constraints to bound the upper and lower conductivity limits. The visualization in this example demonstrates several different ways in which VisIt can be used to view the mesh and conductivity distributions, although not all visualization options available in VisIt are represented. All files necessary to reproduce the examples are included with the E4D distribution in the directory *<e4d_dir>/tutorial/mode_3/metal_box_sheet_line***.**

7.9.2 Conceptual Model and Forward Simulation

Figure 7.11 shows the conceptual subsurface conductivity and metal inclusion used in this example. The model consists of a buried metal box and a sheet; these are the same as those incorporated into the mesh in tutorial 1.2. There is also a dipping plane with a conductivity of 0.01 S/m embedded into a homogeneous background material of 0.001 S/m. Three lines of 48 point electrodes are distributed on the surface as shown.

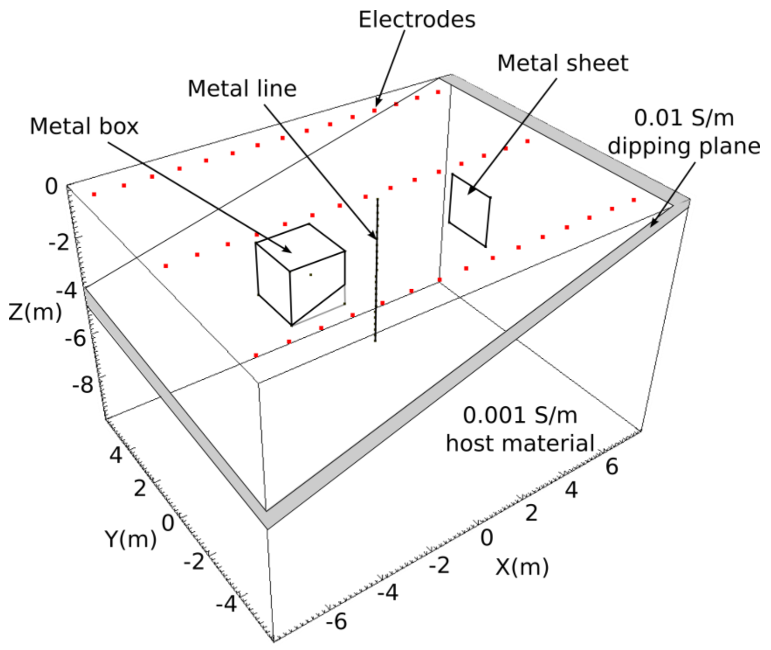


Figure 7.11. Conceptual model conductivity distribution. The metal sheet is beneath the dipping plane. The metal line and box each penetrate the dipping plane.

7.9.3 Forward Mesh Generation

The conceptual conductivity model can be incorporated into the mesh in one of two ways. First, three zones can be explicitly defined in the mesh configuration file, one for the 0.001 S/m foreground region, one for the 0.01 S/m dipping plane, and one for the 0.001 S/m region that extends to the mesh boundaries (not shown). Because the dipping plane intersects the metallic box, such a mesh configuration file would be complicated in comparison to the original (see tutorial 1.2 file *mbsl.cfg*). The second option is to generate a conductivity file using the mesh element centroids to assign a conductivity to each element. For example, elements whose centroids fall within the dipping plane will be assigned a conductivity of 0.01 S/m. Elements whose centroids are not within the dipping plane are assigned a conductivity of 0.001 S/m. This approach requires elements that are small enough to adequately represent conceptual model boundaries, but require no modifications to the *mbsl.cfg* other than changing the maximum element volume limit for zone 1. However, it does require a method of generating the conductivity file outside of E4D. We demonstrate how this may be done in the next section.

As mentioned, to generate a mesh with finely divided elements in zone 1, we change the maximum volume limit (*mz_vol)* for zone 1 and execute E4D in mode 1 to produce the mesh. In this case, we change the maximum volume from 0.1 m^3 to 0.01 m^3 in the mesh configuration file *mbsl.cfg*, as shown in red below. The full mesh configuration file is included with the E4D distribution under *<e4d_dir>/tutorial/mode_3/metal_box_sheet_line/mbsl.cfg.*

<begin mesh configuration file mbsl.cfg>   this line is not included in the file

```
.
.
.
1                                    number of holes ... 1 for the metal box(n_holes)
2 -2.5 0.0 -2.5                      point inside the metal box
2                                    number of zones ... 2, 1 for forground and 1 for extension to outer
boundaries (n_zones)
1 0.0 0.0 -5.0 0.01 0.002            1 xz_1 yz_1 zz_1 mz_vol_1 cond_1
2 0.0 0.0 -20.0 1e12 0.002
1                                    flag to build exodus file (bex_flag)
'../../../bin/bx'                    command to build exodus file (bex_loc)
```

<end mesh configuration file mbsl.cfg>   this line is not included in the file

To generate the mesh, E4D is executed in mode 1 (see mesh generation documentation). A visualization of the resulting mesh is shown in Figure 7.12.

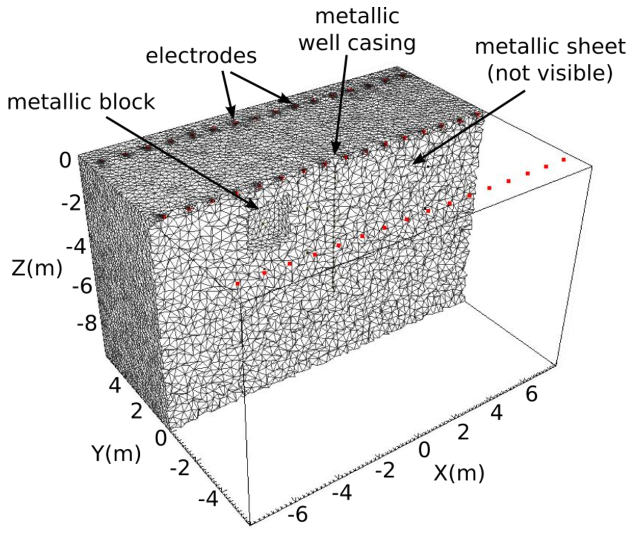


Figure 7.12. Cut-out view of zone 1 of the forward modeling mesh

7.9.4 Generating a Custom Conductivity Distribution Outside of E4D

Users can generate any desired conductivity distribution using the following the steps:

1. Read the mesh node, element, and translation files (i.e**.**, *****.*node, *.ele, *.trn*).
2. Use the node, mesh, and translation files to compute the centroid of each element.
3. Assign a conductivity value to each element using the element centroids.
4. Write the conductivity values to a conductivity file.

An annotated Scilab script that executes each of these steps is included in the E4D distribution, under *<e4d_dir>/tutorial/mode_3/metal_box_sheet_line/build_truesig.sci***,** which writes the conductivity distribution to the conductivity file *mbsl_truesig.sig.* Following this example, users may write similar scripts in a language of choice to complete the same function.

Once the conductivity file has been generated, it can be appended to the exodus file ***(****mbsl.exo****)*** constructed during mesh generation using bx. For example, if the conductivity file is named *mbsl_truesig.sig*, it is appended to *mbsl.exo* using the command:

bx3d -af mbsl.1 mbsl_truesig.sig 1

A visualization of the conductivity distribution produced using the *mbsl_truesig.sig* on the *mbsl.1** mesh is shown in Figure 7.13.

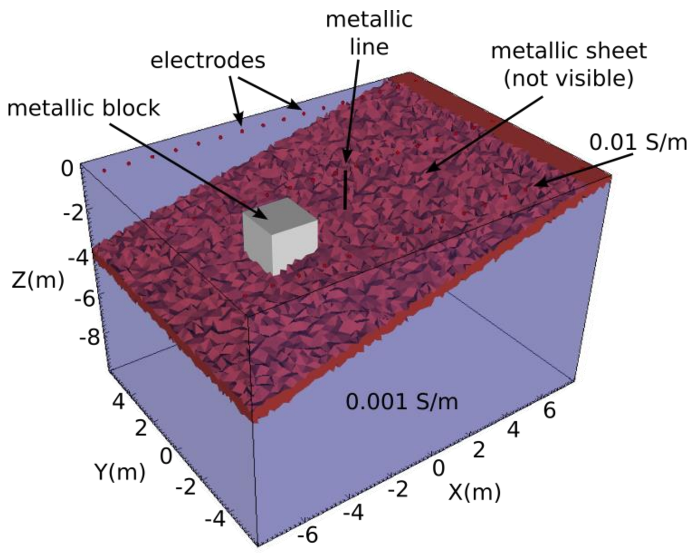


Figure 7.13. VisIt visualization of the true conductivity distribution representing the conceptual model in Figure 7.11.

7.9.5 Synthetic Survey File Generation

Now that the conductivity file has been constructed, we are ready to build the synthetic survey file by running E4D in ER forward mode (mode 2). In mode 2, E4D requires an input survey file, which describes electrode positions and measurement configurations. The measurement values listed in the input survey file are not used in mode 2, but must be included in the file. E4D generates an output survey file that includes the simulated measurements for the given conductivity distribution. Hence, the output survey file is named after the conductivity file used for the simulation, as described in the mode 2 documentation.

In this example, we use the same survey that was used in tutorial 3.1, as specified in the survey file *two_block.srv*. However, we must add the metallic structures to the electrode block of the survey file. To do so, we copy *two_blocks.srv* to *mbsl.srv***,** which will be the input survey file for the forward simulation. We then append the non-point '*electrodes*' after the 48 point electrodes previously specified, one for each metallic structure, making 51 electrodes in total as shown below. The entire file is provided with the E4D distribution in *<e4d_dir>/tutorial/mode_3/metal_box_sheet_line/mbsl.srv*.

<**begin input survey file mbsl.srv**> this line is not included in the file

```
51                                      number of electrodes (48 point and 3 non-point)
1    -7.5    -5.0    0.0    1           electrode_index x y z surface_flag(1 for surface electrode)
.
.
.
49    -4.0    -1.0    -1.0    -1        this is a control point on the metal block (see control pt.61)
50     4.0    -1.0    -1.0    -2        this is a control point on the metal sheet (control pt. 69)
51     0.0     0.0    -0.25   -3        this is a control point on the well casing (see control pt. 125)
2070                                    number of measurements
1  1  2  3  4   -23.41   1.17           measurement# a b m n V/I  st_dev
2  1  2  4  5    -6.03   0.30
.
.
.
```

<**end input survey file mbsl.srv**> this line is not included in the file

Although the metal structures are not used as electrodes in this survey, they can be used as either current or potential electrodes through specification in the measurement block in the same manner as a point electrode.

Given the input files as specified above, the corresponding E4D input file for the forward simulation is shown below.

<**begin input file e4d.inp**> this line is not included in the file

```
2                     run mode
mbsl.1.node           mesh node file name
mbsl.srv              survey file name
```

mbsl_truesig.sig	conductivity file name
mbsl.out	output options file name

<**end input file e4d.inp**>   this line is not included in the file

The forward simulation is executed by running the command (assuming *e4d* and *mpirun* are in the executable path):

mpirun -np N e4d

where *N* is the number of processors to use in the simulation. Note that a single processor cannot compute the pole solutions for both a point and a non-point electrode (see E4D design guide for details). Since there are both point and non-point electrodes in the survey, the minimum number of processors needed to execute the forward run is three; one for the master process, one for the slave process computing the point electrode pole solutions, and one for the slave process computing the non-point electrode pole solutions. The maximum number of processors that can be used is 52, one for the master process and one for each of the 51 electrodes. Thus, N must be at least 3 and at most 52. To give an indication of the compute time required for a forward run, the forward mesh in this simulation has 518,631 elements. The forward simulation executed in approximately 45 seconds on a laptop computer with 8 cores using N=7 (see *run_time.txt* after executing the simulation).

Upon execution, E4D generates as output the synthetic survey file, which is named *mbsl_truesig.sig.srv*. In practice, noise should be added to the simulated measurements to represent the desired field conditions, and the standard deviation should be adjusted accordingly (this can be done outside of E4D using a spreadsheet or other applicable software). We renamed *mbsl_truesig.sig.srv* to *mbsl_dipping_plane.srv* for use in the inversions below.

7.9.6 Smoothness Constrained Inversion

The mesh used to generate the synthetic data was refined in order to adequately represent the boundaries of the dipping plane. We use a coarsened version of the foward mesh for the inversions, which is produced by changing the maximum volume constraint for zone 1 from 0.01 m^3 to 0.1 m^3 in *mbsl.cfg* and running E4D in mode 1. Prior to doing so, we renamed the configuration file to *mbsl_inv.cfg* so that the forward mesh files are not overwritten, thereby producing the inversion mesh files *mbsl_inv.1.node, mbls_inv.1.ele, mbls_inv.1.face, mbsl_inv.1.neigh, and mbsl_inv.trn***.** For reference, the forward mesh contains 518,621 elements, and the inversion mesh contains 95,683 elements. (Note that mesh sizes will vary slightly when generated on different computing systems.) A visualization of the inversion mesh is shown in Figure 7.14, and is visibly coarser than forward mesh shown in Figure 7.12.

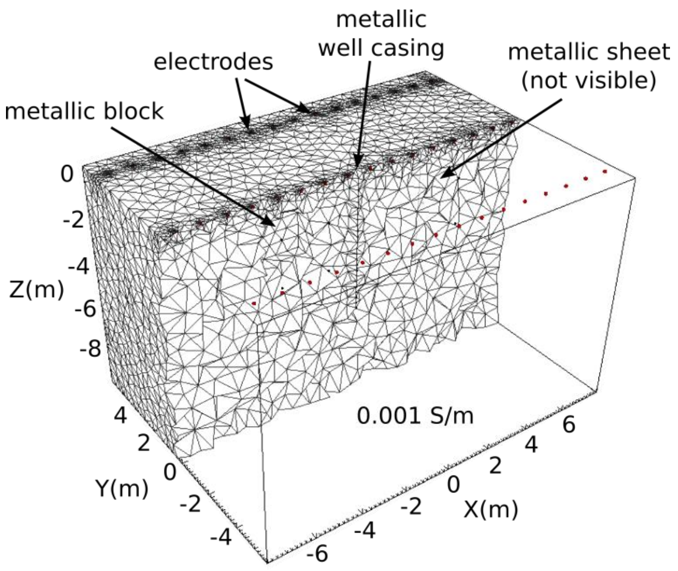


Figure 7.14. Cut-out view of the inversion mesh mbsl_inv.1* .

To execute a smoothness constrained inversion, we construct an inversion options file with only two constraint blocks, one for zone 1, which is bounded by the black lines in Figure 7.14, and one for zone 2, which extends to the mesh boundaries (not shown). To implement the smoothness constraints, we use the same structure metric and weighting function as described in tutorial 3.1. The constraints are shown below in the inversion options file mbsl_1.inv, which is also included in the E4D distribution under <*e4d_dir*>*/tutorial/mode_3/metal_box_sheet_line/mbsl_1.inv.*

7.9.6.1 Inversion Options File: *mbsl_1.inv*

<**begin input survey file mbsl_1.inv**> this line is not included in the file

```
2                        # of constraint blocks
1                        constrain zone 1 with this block
2 1.0 1.0 1.0            use structural metric 2 (absolute log difference between neighbors)
1 5.0 .05                use weighting function 1 with a large mean
1 2                      1 link to zone 2 (i.e. link to zone to using this constraint at the boundary)
0                        structure metric 2 does not use a reference value
1                        relative weight
2                        constrain zone 2 with this block
2 1.0 1.0 1.0            use structural metric 2 (absolute log difference between neighbors)
1 5.0 .01                use weighting function 1 with a large mean
0                        no links necessary, already linked to zone 1 in the first constraint block
0                        structure metric 2 does not use a reference value
1                        relative weight
100 0.25 0.5             start with beta = 100, reduce if % reduction is < 25, reduce by 50%
1.0                      target chi-squared value is 1
30 50                    minimum of 30 inner iterations, maximum of 50
0.00001 1.0              lower and upper conductivity 'safety' limits
```

```
2                              do not use line search
1 3.0                          use data culling with a residual standard deviation of 3
```

<**end input survey file mbsl_1.inv**> this line is not included in the file

The corresponding *e4d.inp* file is shown below.

7.9.6.2 Run Configuration File: *e4d.inp*

<**begin input file e4d.inp**> this line is not included in the file

```
3                          run mode
mbsl_inv.1.node            mesh node file name
mbsl_dipping_plane.srv     survey file name
'average'                  conductivity file name
mbsl.out                   output options file name
mbsl_1.inv                 inverse options file name
none                       no reference file is used
```

<**end input file e4d.inp**> this line is not included in the file

The inversion is executed by calling E4D in the usual manner, using between 3 and 52 processing cores as required. Users are encouraged to execute the inversion and review the log file ***e4d.log.*** Using the settings given in *mbsl_1.inv*, the inversion reduces beta three times to 12.5, and converges in 7 iterations. Other notable features of the inversion that have not been discussed previously include:

- Since 'average' was specified for the starting model conductivity, E4D computed the average apparent conducting and used that value for the homogeneous starting model. That value is approximately 0.016 S/m, which is larger than the conductivity of both the background material and the dipping plane, and is caused by the presence of metallic structures. In similar field cases (i.e., in the presence of metallic structures), it may be possible to estimate a more appropriate starting model value— one that is not influenced by the metal.
- Within E4D, a single slave processor can be assigned to more than one electrode, but cannot be assigned both a point electrode and a non-point 'electrode' (i.e., a metal structure). Given this constraint, E4D balances the computations as well as possible. For example, using 7 processors, E4D assigns all three metal structures to one slave process, and divides the 48 point electrodes evenly among the remaining five slave processes (note there are six slaves and one master). Each of these five slave processes is assigned to compute the pole solutions for either 9 or 10 point electrodes.
- The inversion accounts for the metal structures through forward modeling, and does not attempt to image them because their positions and dimensions are assumed known (see E4D theory guide). The inversion estimates the conductivity of the soils surrounding the metal structures, while accounting for the effects of those structures on the observed data through forward modeling.

7.9.6.3 Visualizations

Figure 7.15 shows three different visualizations of the inverse solution produced by 1) constructing an exodus file using ***bx*** and 2) visualizing the exodus file using VisIt. The left-most visualization shows the estimated conductivity structure of zone 1 in the region y > 0. The center visualization adds transparent conductivity isosurfaces to the left-most visualization, and the right-most visualization shows only the transparent isosurfaces.

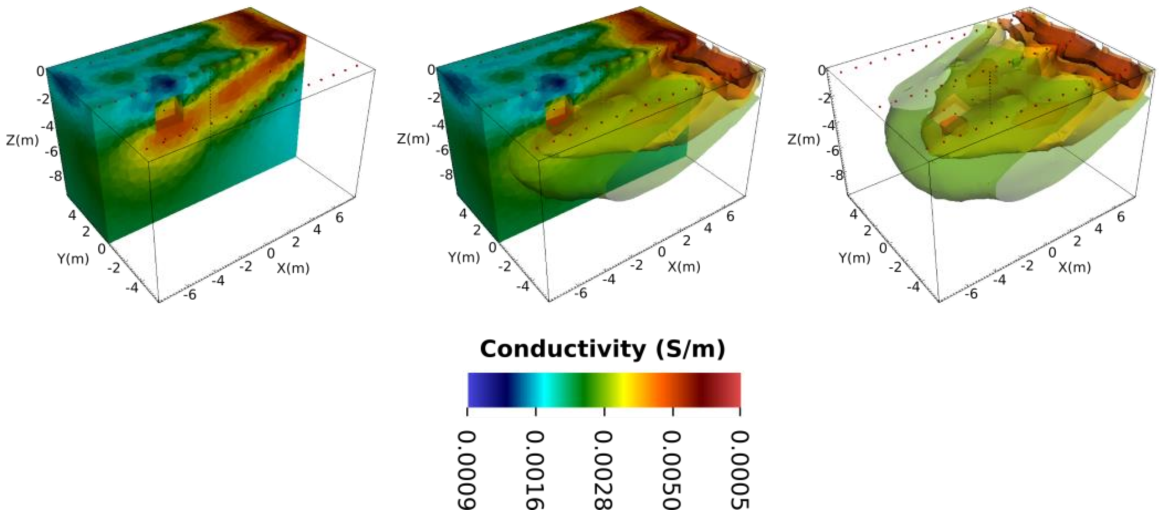


Figure 7.15. Three different VisIt visualizations of the smoothness constrained inverse solution. Resolution of the planar structure decreases with as the plane move deeper into the subsurface, causing it to 'disappear' in the deeper regions. The the conductivities of the metallic structures are not apparent in the inversion results because they are accounted for in the forward modeling step of the inversion (see E4D theory guide).

7.9.7 Anisotropic Weighting and Inequality Constraints

For this inversion we build upon the last by providing, through the constraint blocks in the inversion options file, two additional pieces of information: 1) the orientation of the plane and 2) minimum and maximum conductivity limits. The orientation of the plane is provided with structural metric 6, which provides the absolute difference in log conductivity between adjacent elements, and takes as input a vector normal to the plane of preferred smoothing. Neighboring elements with centroids aligned parallel to the plane are weighted more heavily than elements aligned perpendicular to the plane, thereby fostering continuity along the plane. The minimum and maximum conductivity constraints are implemented as inequality constraints in the same manner as was demonstrated in tutorial 3.1. Herein, we set the minimum and maximum conductivity limits to 0.001 S/m and 0.02 S/m. An annotated version of the inversion options file that implements the constraints is shown below. This file is included in the E4D distribution under *<e4d_dir>/tutorial/mode_3/metal_box_sheet_line/mbsl_2.inv*.

7.9.7.1 Inversion Options File: *mbsl_2.inv*

<**begin input survey file mbsl_2.inv**> this line is not included in the file

```
4                      4 constraint blocks listed below
1                      constrain zone 1 with this block
6   0.25   0.0   1.00  use metric 6 to encourage continuity normal to the dipping plane
1 5.0 .02              use weight function 1 with large mean to encourage smooth structure
1 2                    apply these constraints at the boundary with zone 2
0                      reference value (not used for structure metric 6)
1                      relative weight
2                      constraint zone 2 with this block (same constraints as zone 1 above)
6  0.25   0.0  1.00
1  5.0   .02
0
0
1
1                      constraint zone 1 with this block
3  1.0  1.0  1.0       use structure metric 3 (log(conductivity) - log(reference))
1  0.0    .1           use weighting function 1 to implement a lower bound inequality constraint
0                      no links
-6.91                  reference value: lower bound = log(0.001) = -6.91
2                      double weight for stronger enforcement
1                      constrain zone 1 with this block
3  1.0  1.0  1.0       use structure metric 3 (log(conductivity) - log(reference))
2  0.0  0.1            use weighting function 2 to implement an upper bound inequality constraint
0                      no links
-3.91                  reference value: upper bound = log(.02) = -3.91
2                      double weight for stronger enforcement
200 0.25 0.5           start with beta = 200, reduce if % reduction is < 25, reduce by 50%
1.0                    target chi-squared value is 1.0
30 50                  at least 30 and at most 50 inner iterations
0.00001 1.0            lower and upper conductivity 'safety' limits
2                      do not use line search
1 3.0                  use data culling with a residual standard deviation of 3
```

<**end input survey file mbsl_2.inv**> this line is not included in the file

7.9.7.2 Run Configuration File: *e4d.inp*

<**begin input file e4d.inp**> this line is not included in the file

```
3                        run mode
mbsl_inv.1.node          mesh node file name
mbsl_dipping_plane.srv   survey file name
'average'                conductivity file name
mbsl.out                 output options file name
mbsl_1.inv               inverse options file name
none                     no reference file is used
```

<**end input file e4d.inp**> this line is not included in the file

The inversion is executed by calling E4D in the usual manner, using between 2 and 52 processing cores as required. Users are encouraged to execute the inversion and review the log file *e4d.log*. In this case, the inversion does not reduce beta from the initial value of 200 and converges in only 2 iterations, compared to 7 iterations for the smoothness constrained inversion.

7.9.7.3 Visualizations

Three visualizations similar to those shown in Figure 7.15 are shown below in Figure 7.16. Resolution of the dipping plane is improved in comparison to the smoothness constrained inversion. Note that blocky boundaries develop on the top and the bottom of the imaged plane, even though weighting function settings for this inversion are the same as those used for the smoothness constrained inversion. The sharp conductivity contrasts at the boundaries of the plane are enabled by the anisotropic weighting, which in this case reduces the similarity constraints normal to the plane, thereby allowing the inversion to place a sharp contrast normal to the plane without penalty. This demonstrates the value of providing the inversion with a priori information in terms of constraining the inversion to improve imaging resolution.

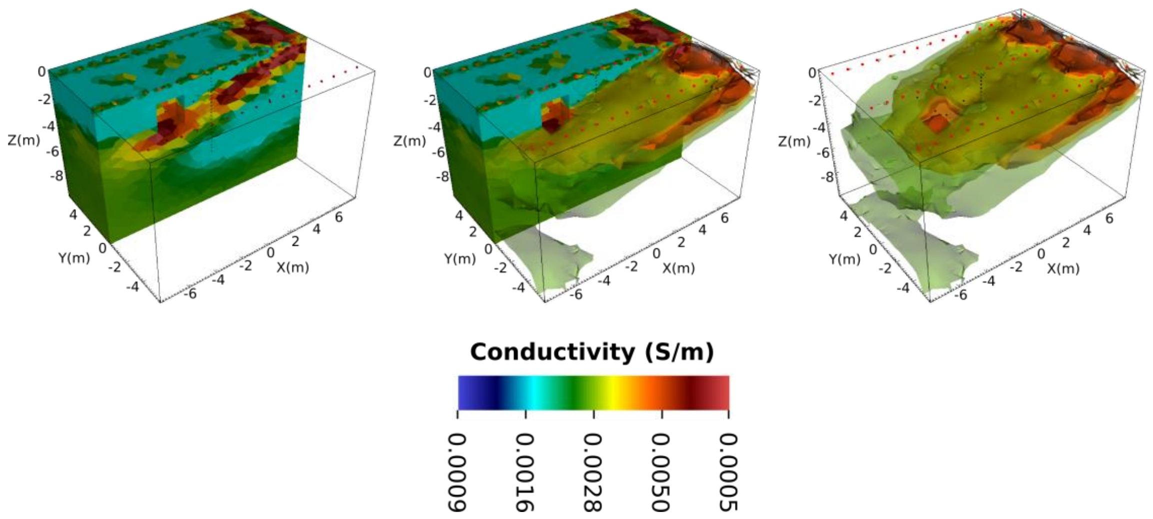


Figure 7.16. Three visualization of the inversion with anisotropic smoothing and upper and lower conductivity boundary constraints.

8.0 Mode 4: ER Time-Lapse Inversion Mode

8.1 Overview

In mode 4, E4D sequentially inverts a list of user supplied survey files to produce a series of time-lapse ERT images. Although users may specify a number of model constraints in the space and/or time dimension, the standard approach is to use the space-time smoothness constraints given by structural metrics 7 and 8. Structural metrics 7 and 8 compute the difference between the temporal change in log conductivity among spatially adjacent elements, where the temporal change for a given element is the log conductivity of that element minus the log conductivity of the same element at some reference time. The reference model can be either the baseline inverse solution (i.e., the first solution in the time-lapse sequence), or the previous solution in the sequence. Using structural metrics 7 and 8, a number of non-linear temporal inequality constraints can be placed on the time-lapse solutions, depending on the weighting function, and the weighting function parameters chosen for the metric.

All of the constraint options available to mode 3 are also available in mode 4. In addition, the functionality enabled by mode 4 can be duplicated using repeated executions of mode 3. Mode 4 automates the process for list of surveys and reduces computation time. The input requirements for mode 4 are equivalent to mode 3, except that an additional input file, the time lapse survey list file, must be provided. In this chapter we describe the mode 4 input requirements and provide several model constraint examples. We then discuss how E4D reports time-lapse inversion results, and end with a synthetic time-lapse inversion tutorial.

8.2 File Format for *e4d.inp*

The mode 4 format for the run configuration file *e4d.inp* is identical to that of mode 3, except the addition of one line at the end of the file that specifies the name of the time-lapse survey list file, and a flag indicating which conductivity distribution should be used as the reference (the baseline conductivity or the previous inverse solution), as shown below.

Variable	Type	Description
run_mode	integer	For a time-lapse ER inversion the *run_mode* should be set to 4
mesh_file	string	The *mesh_file* variable should set to <*mesh_prefix*>.1.node, or <*mesh_prefix*>.1.ele, where <*mesh_prefix*> is the prefix of the mesh files. These files are produced in mesh generation mode using a mesh configuration file name <*mesh_prefix*>.cfg.
survey_file	string	*survey_file* is the name of the user-created survey file that describes electrode locations and the survey configuration. In mode 4 *survey_file* is the name of the baseline survey file.
conductivity_file	String or real	*conductivity_file* specifies either the name of the conductivity file, or the word 'AVERAGE', 'Average', or 'average'. In mode 4 *conductivity_file* provides the starting conductivity distribution for the inversion. Users may use the conductivity file created during mesh generation or construct their own conductivity file to use as the starting model. Alternatively, if

Variable	Type	Description
		conductivity_file is one of the forms of the word 'average' as described above, E4D computes the average apparent conductivity of each measurement in the survey file, and uses that conductivity as the homogeneous starting model value. Each apparent conductivity is weighted according the standard deviation of the corresponding measurement. In the absence of other information, it is generally recommended to use the average apparent bulk conductivity as the starting model. However, it may not be appropriate to use this approach in the presence of significant surface topography, because there is no analytic solution by which to compute the apparent conductivity when surface topography exists. In this case, E4D will use the average elevation of the surface nodes of the mesh as the surface elevation for the apparent conductivity computations, which may provide an unreasonable starting model value. If conductivity file is a real number, then that value is used as the homogeneous starting conductivity.
out_opts_file	string	*out_opts_file* specifies the name of the output options file.
inv_opts_file	string	*inv_opts_file* specifies the name of the inversion options file, which provides E4D instructions concerning how to constrain the model, how to execute each time-lapse inversion, and how to determine if the solution has converged. This file is discussed in detail in the forthcoming section titled 'Inversion options file'.
ref_mod_file	string	*ref_mode_file* specifies the name of a baseline conductivity file the that is optionally used to constrain the inversion as described in the inversion options file. If a baseline reference model is not used to constrain the inversion, then *ref_mode_file* is not accessed by E4D, and therefore does not have to exist. However, a value for *ref_mode_file* must always be included on this line in *e4d.inp* as a placeholder.
tl_list sm_opt	see description	*tl_list* is the name of the time lapse survey list file, which specifies the names and time stamps of the time-lapse survey files. *sm_opt* is the time-lapse starting model option, and has integer valuee of 1 or 2. Enter 1 if *ref_mode_file* (see above) should be used as the starting model for all time-lapse inversions. Enter 2 if the solution to the previous inversion listed in *tl_list* should be used as the starting model for each time-lapse inversion.

If a reference model (either baseline or previous solution) is to be used to constrain each time-lapse inversion, it is required to first invert the baseline solution in mode 3. In addition to providing the baseline reference model (see *ref_mode_file* above), the baseline inversion will provide important information concerning appropriate model constraint options and convergence settings for the time-lapse inversion. It is therefore advisable to invert the baseline data set prior to executing a time-lapse inversion. Examples are provided in the tutorial at the end of the chapter.

The baseline survey file (see table above) and each of the time-lapse survey files listed in *tl_list* must have the same electrode list and survey configuration (i.e., the same number of measurements listed in the same order). If any of the time-lapse survey files have survey configurations that are different from the baseline survey, E4D will print an error message and exit.

8.3 Time-Lapse Survey List File Format

The format of the time-lapse survey list file is shown below and in Appendix A

Variable	Type	Description
n_files	integer	Specifies the number of survey files listed
s_file t_stamp	See description	One row is included for each of the *n_files* survey files, where *s_file* is the name of the survey file (string) *t_stamp* is a time stamp for the file (real)

8.4 Mode 4 Inversion Options File

The inversion options file for mode 4 is identical to the one for mode 3. Users are encouraged to review the mode 3 documentation to become familiar with the inversion options file, particularly the structural metric and weighting function options. All of the inversion options available in mode 3 are available in mode 4. However, users will typically use different inversion options in mode 4 than in mode 3, assuming transient constraints are used in the time-lapse inversion. In the next section, several time-lapse constraint options are given.

8.5 Example Time-Lapse Inversion Constraints

8.5.1 Spatially Smoothing the Transient Change in Conductivity

One useful aspect of time-lapse ER imaging is that static effects can be removed by subtracting the baseline image from the time-lapse image, revealing only what has changed with time. This facilitates the capability to reveal small changes in conductivity that may not be evident in the original images. The time-lapse analog to the static smoothness constraint, through which spatial changes in conductivity are constrained to vary smoothly in space, is the transient smoothness constraint, where transient changes in conductivity are constrained to vary smoothly in space. The transient smoothness constraint is provided by structural metrics 7 and 8, which are given below for reference.

Structure metric code (***s_met***)	Equation
7	$X = (m_t - v\_ref_t) - (m_n - v\_ref_n)$ $X = Dm_t - Dm_n$
8	$X = \|(m_t - v\_ref_t) - (m_n - v\_ref_n)\|$ $X = \|Dm_t - Dm_n\|$
Variables: X = value of the structural metric, and the independent variable in each weighting function. m_t = log conductivity of the target element m_n = log conductivity of the target elements neighbor v_ref_t = log conductivity of the target elements reference value, taken from either the constraint block, the reference model file, or the previous time-lapse solution as specified in the inversion options file v_ref_n = log conductivity of neighbors reference value, taken from either the constraint block, the reference model file, or the previous time-lapse solution as specified in the inversion options file Dm_t = change in log conductivity of target element with respect to v_ref_t Dm_n = change in log conductivity of neighbor element with respect to v_ref_n	

We focus here on structural metric 8, which provides the absolute difference of transient changes in conductivity between neighboring elements. By minimizing structural metric 8, the inversion smooths the transient change in conductivity from the reference model, subject to fitting the data. Consider the following constraint block:

<begin constraint block> this line is not included in the inverse options file

1	this is the zone number constrained by this block
8 1.0 1.0 1.0	use structural metric 8, the x,y,z weighting values are not used
1 10 0.001	use weighting function 1 with a mean = 10 and a st. dev. = 0.001.
1 2	apply the constraints across 1 boundaries; the boundary with zone 2
'ref'	use the reference model as the reference value in the structural metric
1.0	this is the relative weight for this structure metric.

<end constraint block>

Here we have constrained zone 1 with structural metric 8 and weighting function 1. Weighting function 1 applies full weight (i.e., 1) to the constraint implemented by the structural metric until the value of the structural metric reaches the specified mean value of the weighting function. Since the mean of the weighting function is specified as 10, the constraint implemented by structural metric 8 will be imposed unless X (see table above for metric 8) reaches a value of 10, which requires a very large change in transient conductivity. As long as the transient changes in log conductivity between neighbor elements remains below 10, the inversion will impose the transient smoothing constraint.

We have also specified that the constraint should be enforced at the zone 1/zone 2 boundary, and that the reference model should be used to provide the reference value, as will typically be the case in mode 4. The reference model used is specified by *tl_opt* in the run configuration file *e4d.inp* (shown above). If *tl_opt* = 1, then the baseline reference model is used as the reference for all time-lapse inversions. If *tl_opt* = 2, then the previous time-lapse solution is used as the reference model.

8.5.2 Transient Smoothing with Sharp Boundaries

As discussed in the mode 3 documentation, sharp boundaries are blocky inversions that are enabled by reducing the mean of the weighting function given in the smoothness constrained inversion shown above. For example:

<begin constraint block> this line is not included in the inverse options file

1	this is the zone number constrained by this block
8 1.0 1.0 1.0	use structural metric 8, the x,y,z weighting values are not
1 .1 0.01	use weighting function 1 with a mean = 10 and st. dev. = 0.001.
1 2	apply the constraints across 1 boundary; the boundary with zone 2
'ref'	use the reference model as the reference value in the structural metric
1.0	this is the relative weight for this structure metric.

<end constraint block>

With these settings, the transient smoothness constraints imposed by metric 8 will be removed if X rises above the weighting function mean of 0.1 for any pair of neighboring elements in zone 1 (or at the zone 1/zone 2 boundary), enabling a sharp contrast to develop between those elements, similar to that described in the mode 3 documentation.

8.6 Notes on Time-Lapse Inversion

As mentioned previously, it is generally advisable to invert the baseline data set in mode 3 prior to inverting the time-lapse data. This is required if the inversion options for the baseline inversion are different from the inversion options for the time-lapse inversion, which will always be the case if reference models are used to constrain the time-lapse inversion. The general strategy is outlined by the bulleted items below.

- Invert the baseline data set and use the baseline inversion to determine the appropriate target chi-squared value, if necessary.
- For the time-lapse inversion, specify the baseline survey file in *e4d.inp*, and the baseline inversion result as both the starting model and the reference model conductivity file.
- Set the target chi-squared value for the time-lapse inversion to the value at which the baseline inversion converged. This is the appropriate value if data errors are comparable in each time-lapse data set.
- Execute the time-lapse inversion with other options as chosen.

There are advantages and disadvantages to using the baseline solution or previous solutions as reference models. In both cases, the reference model is used as the starting model, and the inversion iterates until one of two conditions are satisfied, depending on whether *up_opt* is specified as 2 or 3 in the inversion options file. If *up_opt* is specified as 2, then the inversion will reduce the *beta* value as required in order to reach the target chi-squared value *chi_target***.** Therefore, there is no maximum limit on the model constraint error that can be imposed by the inversion in order to fit the target chi-squared value. If *up_opt* is specified as 3, then the *beta* value is not reduced during the inversion. The inversion converges if 1) the chi-squared value reaches the target chi-squared value or 2) the objective function decreases by less than *min_red* from the previous iteration. Because the *beta* value is not reduced, there is an implicit limit on the model constraint error that can be imposed by the inversion before converging. However, because beta is not allowed to decrease, the target chi-squared value may not be achieved.

When analyzing time-lapse inversion results, the baseline solution is typically subtracted from each time-lapse solution to reveal the change in conductivity with time. When the baseline solution is used as the reference model, constraints are imposed with respect to a common conductivity distribution for every time-lapse inversion result. This means that, for example, if smoothness constraints are applied, then each time-lapse solution will provide a smooth difference from the baseline solution as desired. The undesirable aspects of using the baseline as reference follow. First, the starting model for every time-lapse solution is the baseline solution, which may over time become 'far' from the final solution, requiring more computation time than if the previous solution were used as the starting model. Second, there is no smoothness constraint between subsequent time-lapse data sets, and therefore no constraint requiring that conductivity of a given element vary smoothly in time. These two disadvantages are addressed by using the previous model as the reference model. However, when using the previous model as reference, there is no constraint to a common baseline. Because of this, model heterogeneity can accumulate as the time-lapse inversion progresses if the beta value is set too low, resulting in smoothly varying, but overly heterogeneous solutions.

8.7 Reporting

E4D reporting in mode 4 is similar to that of mode 3. The *e4d.log* file provides convergence information for each data set. Simulated and observed data are printed to the simulated data file after each inverse iteration. The current conductivity is printed to the conductivity *si*.* where * is the current iteration number for the current inversion time, and the iteration numbers restart at zero with each new time-lapse survey. The solutions for each time-lapse inversion are printed to the conductivity files *tl_sig** where * is the time stamp provided in the survey list file. The status of the inversion is printed to the standard output, and timing information is printed to the file *run_time.txt*.

9.0 ER Time-Lapse Inversion Tutorial 4.1: Sinking Plume

This tutorial reviews mesh generation and forward modeling steps presented in previous tutorials, with the objective of generating a set of synthetic time-lapse survey files that contain the data by which to image a spherical anomaly that moves downward through the subsurface. Once the synthetic survey files are generated, we use them in mode 4 to image the movement of the anomaly using different time-lapse inversion settings.

The forward and inversion meshes are based on those used in previous examples, and the conductivity distributions are generated using a scilab script. The conceptual model for the sinking plume is shown in Figure 9.1. The center of the plume is located at (0,0,z) m, where z is the elevation, which starts at zero and moves downward 0.5 m at each time step. The conductivity of the plume is 0.1 S/m for $r<=1$, and $0.1/(r^4)$ S/m for $1<r<2$, where r is the distance from the plume centroid in meters. The background conductivity is homogeneous at 0.001 S/m.

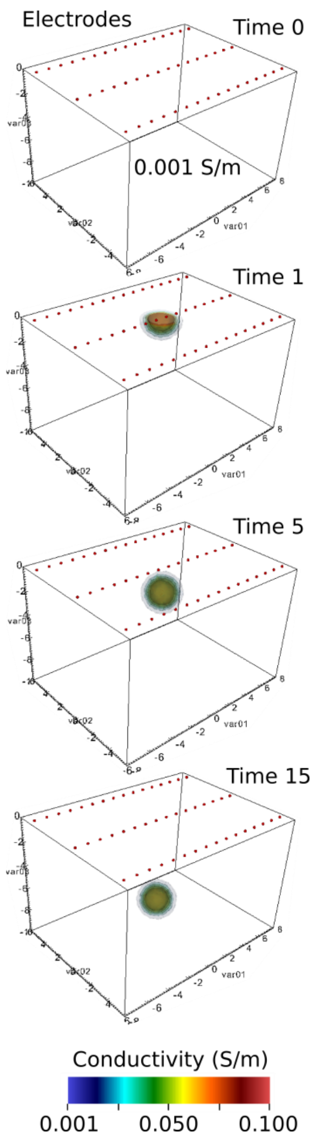


Figure 9.1. Conceptual model of a spherical conductivity anomaly moving downward at 0.5 m per time step in a background medium of 0.001 S/m.

9.1 Forward Modeling

9.1.1 Generating the Forward Modeling Mesh

To model the plume, we begin by generating a forward modeling mesh consisting of elements that are small enough to accurately represent plume boundaries. The mesh configuration file used to generated the forward modeling mesh is provided with the E4D distribution under *<e4d_dir>/tutorials/mode_4/sinking_plume/sp.cfg*. Users are encouraged to review *sp.cfg* and generate the forward modeling mesh (see mesh generation documentation for details). Zone 1 of the forward modeling mesh is shown in Figure 9.2. The maximum volume of elements in zone 1 is 0.01 m^3 as specified in *sp.cfg*, thereby providing a highly refined mesh (518,384 elements in this case).

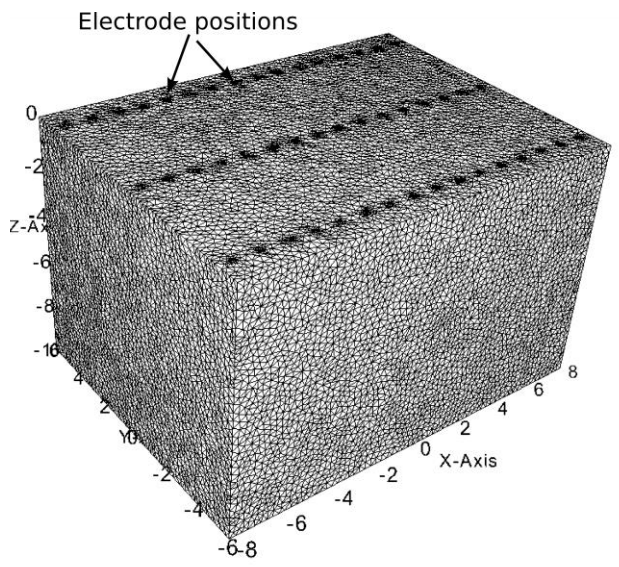


Figure 9.2. Zone 1 of the forward modeling mesh.

9.1.2 Generating the Time-Lapse Conductivity Distributions

The time-lapse conductivity distributions are generated outside of E4D, and include a homogeneous baseline conductivity distribution and one anomalous conductivity distribution for each of the depth increments of the plume (i.e., 0 to -8 meters elevation in 0.5 m increments). The scilab script used to generate the conductivity files corresponding to each distribution are provided with the E4D distribution under *<e4d_dir>/tutorials/mode_4/sinking_plume/build_conds.sci*. Users are encouraged to review that script to determine how custom conductivity distributions may be generated for E4D. The basic steps include: 1) reading node and element files, 2) translating the node locations (using the translation coordinates in *.trn), 3) using the node and elements definitions to determine the centroid of each element, 4) using the element centroids to define the conductivity fields, and 4) generating the conductivity files for each conductivity field. If users choose not to execute this step, the forward mesh files are provided with the E4D distribution in a gzipped tar archive under *<e4d_dir>/tutorials/mode_4/sinking_plume/spf.tgz.* The corresponding conductivity files are located in *<e4d_dir>/tutorials/mode_4/sinking_plume/true_sigs.tgz.* Figure 9.3 shows visualizations of several time steps generated using bx and VisIt.

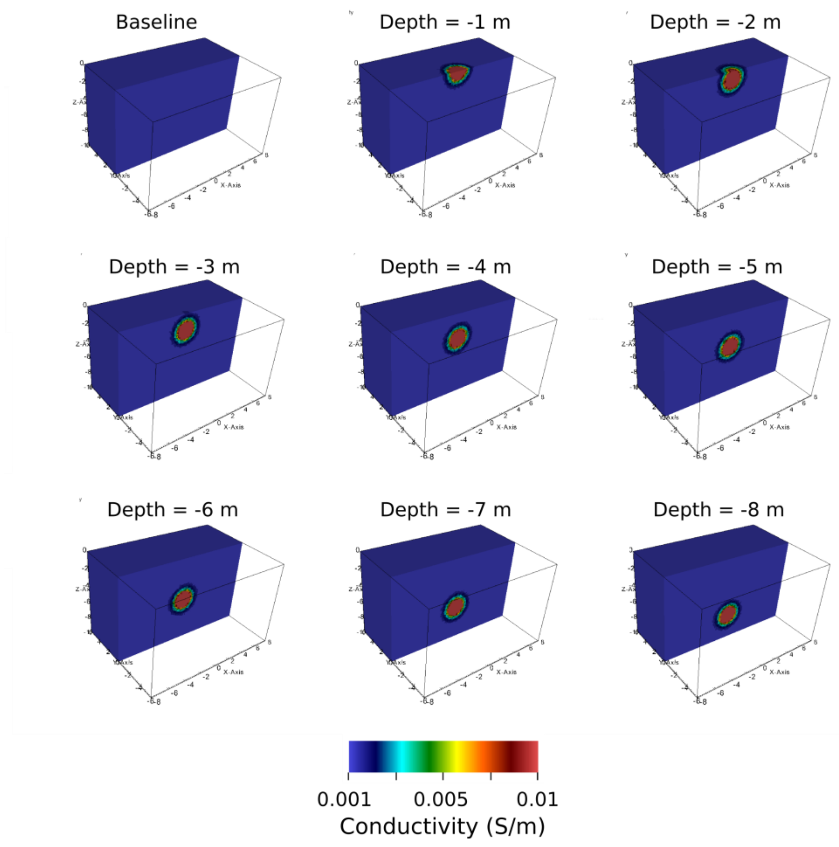


Figure 9.3. Synthetic conductivity distribution at several depths. The conductivity files were generated for the forward mesh using scilab to execute *build_conds.sci*, which is a scilab script included with the E4D distribution.

9.1.3 Synthetic Survey Generation

Synthetic survey files are generated by executing *e4d* in mode 2 once for each of the conductivity files. The scilab script discussed in the previous section generates conductivity files named *sig*.****, where *.*** is the depth of the center of the plume. When *sig*.**** is specified as the conductivity file in *e4d.inp*, and *e4d* is executed in mode 2, *e4d* produces an output survey file named *sig*.***.srv*. This survey file contains the simulated transfer resistance measurements specified in the survey file, given the conductivity field specified in *sig*.****. The input survey file is provided with the E4D distribution under *<e4d_dir>/tutorials/mode_4/sinking_plume/sp.srv*. The corresponding output survey files generated using mode 2 are provided in *<e4d_dir>/tutorials/mode_4/sinking_plume/surveys.tgz*. Note that noise was not added to the measurements, but the standard deviations are specified as 5% of the transfer resistance magnitude plus 0.001 ohms. Although the inversion will be able to fit the data to a high tolerance in this case (i.e., a very low chi-squared value), it is set to converge at chi-squared = 1 in the inversion.

9.1.4 Generating the Inversion Mesh

The inversion mesh is constructed using mode 1 with a survey file identical to *spf.cfg*, except that the maximum allowable volume for zone 1 is specified as 0.1 m^3 instead of 0.01 m^3. The mesh configuration file for the inversion mesh is included in with the E4D distribution under *<e4d_dir>/tutorial/mode_4/sinking_plume/sp.cfg*. A visualization of zone 1 of the corresponding mesh is show in Figure 9.4. The mesh contains 95,387 elements in total.

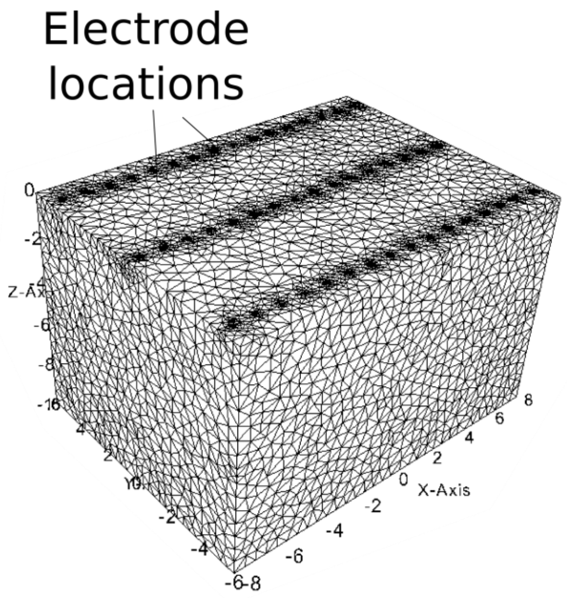


Figure 9.4. Zone 1 of the inversion mesh

9.1.5 Baseline Inversion

The baseline conductivity distribution is homogeneous (0.001 S/m) in this tutorial. We invert the baseline survey (baseline.srv) using a starting conductivity of 0.002 S/m. The *e4d.inp* and inversion options files are shown below.

<begin file e4d.inp> this line is not included in the file

```
3                 run mode
sp.1.node         mesh node file
baseline.sig.srv  baseline survey file
0.002             starting conductivity value
sp.out            output options file
sp.inv            inversion options file
none              reference model file (not used … see sp.inv below)
```

<end file e4d.inp> this line is not included in the file

<begin file sp.inv> this line is not included in the file

```
2                 number of constraint blocks
```

```
1                  constrain zone 1 with this block
2 1.0 1.0 1.0      use structural metric 2 (absolute difference in log conductivity between neighbors)
1 10.0 0.15        use weighting function 1 with a large mean value to impose smoothness constraints
1 2                apply the constraints across the boundary with zone 2
0                  no reference model for this metric (this value is ignored)
1.0                relative weight for this constraint

2                  constrain zone 2 with this block
2 1.0 1.0 1.0      use structural metric 2 (absolute difference in log conductivity between neighbors)
1 10.0 0.15        use weighting function 1 with a large mean value to impose smoothness constraints
0                  no reference model for this metric (this value is ignored)
0                  do not link to any zones (already linked to zone 1 in the block above)
1.0                relative weight for this constraint

500  0.20 0.5      start beta at 500, decrease if obj. fnc. changes by less than 20%, decrease by 50%
1.0                target chi-squared value
30  50             minimum maximum number of inner iterations
0.00001 1.0        min and max allowable conductivity
2                  no line search on beta
1 3.0              use data culling with if the weighted residual >= 3.0 deviations from the mean
```

<end file sp.inv> this line is not included in the file

The inversion options file implements an L2 norm smoothness constrained inversion (see E4D theory guide), which is consistent with the homogeneous baseline conductivity in this case. The inversion converges in two iterations, providing the baseline conductivity distribution shown in Figure 9.5. Note that had we specified 'average' for the conductivity file in *e4d.inp*, *e4d* would have computed the average apparent conductivity and used that value as the starting value. Since the apparent conductivity of every measurement is the true conductivity in this case, the data would have been fit exactly at the starting model and no inversion would have been executed.

Although the baseline inversion results are nearly homogeneous at the true value of 0.001 S/m, there is obviously some heterogeneity. Because the homogeneous model constraints are perfectly consistent with the data in this case, a perfectly recovered homogeneous model might have been expected . Such results may be possible by increasing the starting beta value, thereby forcing the inversion to better honor the smoothness constraints. The data lose sensitivity to the conductivity distribution below about -6 m elevation, as evidenced by the upward trend in conductivity toward the starting model of 0.002 S/m. This loss in resolution at greater depths will also be evident in the time-lapse inversion results.

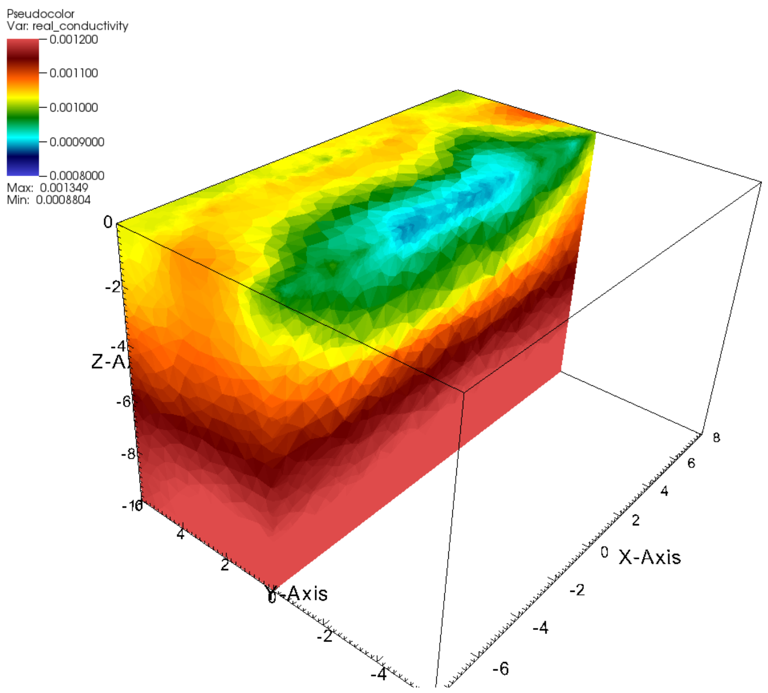


Figure 9.5. Baseline inversion conductivities range from approximately 0.0008 to 0.0012 S/m, bounding the true conductivity of 0.001 S/m.

9.2 Time-Lapse Inversions Using the Baseline Solution as Reference

In this tutorial, we demonstrate two time-lapse inversions using time-lapse smoothness constraints. In this section, we use the baseline solutions as a reference model for each time solution. In the next section, we use the previous time-lapse solution as the reference model. To investigate other options, we encourage users to modify these examples to include other constraints, such as maximum and minimum conductivity bounds, blocky time-lapse constraints, known value constraints, etc. For example, similar to the static inversions demonstrated in tutorial 3.1, blocky time-lapse constraints can be implemented by simply decreasing the mean of the weighting function in each of the examples shown below. The inversions execute relatively quickly and are informative concerning the effects of different constraining conditions. It is also informative to plot the intermediate solutions and interpret in conjunction with the log file.

When constraining a time-lapse inversion to use the baseline solution as reference, it is generally recommended to use *up_opt = 2* in the inversion options file, which enables *e4d* to reduce the beta value when necessary. This is particularly important if the time-lapse solutions are expected to change significantly from the baseline solution. The input options file and inversion options file shown below illustrate the basic settings that can be used for a time-lapse inversion that is constrained using a baseline solution reference model. Note the baseline solution conductivity file in Figure 9.5 is named *baseline_sp.sig* in *e4d.inp*, as shown below. The inversion options file is name *sp_tl.inv*, and the survey list file is named *surveys.txt*. Each of these files is included with the E4D distribution under *<e4d_dir>/tutorials/mode_4/sinking_plume* .

<begin file e4d.inp>   this line is not included in the file

```
4                    run mode
sp.1.node            mesh node file
baseline.sig.srv     baseline survey file
baseline_sp.sig      starting conductivity value
sp.out               output options file
sp_tl.inv            inversion options file
baseline_sp.sig      reference model file
surveys.txt  1       survey list file, using baseline model as starting model
```

<end file e4d.inp>   this line is not included in the file

<begin file sp_tl.inv>   this line is not included in the file

```
2               number of constraint blocks

1               constrain zone 1 with this block
8 1.0 1.0 1.0   use structural metric 8 (spatial difference of temporal differences)
1 10.0 0.15     use weighting function 1 with a large mean value to impose smoothness constraints
1 2             apply the constraints across the boundary with zone 2
‘ref’           use reference model specified in e4d.inp as reference
1.0             relative weight for this constraint

2               constrain zone 2 with this block
8 1.0 1.0 1.0   use structural metric 8
1 10.0 0.15     use weighting function 1 with a large mean value to impose smoothness constraints
0               do not link to any zones (already linked to zone 1 in the block above)
‘ref’           use reference model specified in e4d.inp as reference
1.0             relative weight for this constraint

200  0.05 0.5   start beta at 500, decrease if obj. fnc. changes by less than 20%, decrease by 50%
1.0             target chi-squared value
30  50          minimum maximum number of inner iterations
0.00001 1.0     min and max allowable conductivity
2               no line search on beta, reduce beta as needed
1 3.0           use data culling with if the weighted residual >= 3.0 deviations from the mean
```

<end file sp_tl.inv>   this line is not included in the file

<begin file surveys.txt> this line is not included in the file

```
21                              number of time-lapse survey files
sig_0.sig.srv      0.0          first survey file and time stamp
sig_0.5.sig.srv    0.5          second survey file and time stamp
sig_1.sig.srv      1.0          third survey file and time stamp
.
.
.
sig_10.sig.srv     10           21st and final survey file and time stamp
```

<end file surveys.txt> this line is not included in the file

The inversion is executed as normal, and requires from 9 to 12 iterations for convergence for each time-lapse solution with these settings when the plume is within the resolved zone. As will be shown, the inversion requires from 1 to 3 iterations per solution when using the previous solution as the reference. This is because the previous solution is generally closer to the true solution than the baseline solution. However, using the baseline solution enables some useful constraints that are not available when the previous solution is used. For example, induced tracer tests often involve conductive tracers, and it can be assumed that the subsurface conductivity is always greater than or equal to the baseline conductivity. Therefore, a constraint can be implemented that encourages the time-lapse conductivity distribution to be everywhere greater than the baseline conductivity distribution for each inversion. Note that, although it is not demonstrated in this tutorial, it is possible to use both the reference model and previous model to constrain the inversion using separate constraint blocks.

A selected set of the time-lapse solutions generated using the settings above is shown in Figure 9.6 in comparison to the true solutions and the previous model reference solutions discussed in the next section.

9.3 Time-Lapse Inversions Using the Previous Solution as Reference

The file modification necessary to run the inversion using the previous solution as the reference model for each time-lapse inversion are shown below. In this case each time-lapse solution converges in one to three iterations. Results are shown in comparison to the true model and the inversion executed in section 9.2. Note that the results using the previous model as reference show the 'shadow' of the previous inversion. This happens when the plume moves below the depth where it is sensitive to the data (about 4-5 m depth), whereby changes in the data are insufficient to resolve changes in the conductivity distribution from the previous solution (which is also the starting model).

Users are encouraged to experiment with these inversions by, for example, adding a positivity constraint to the change in conductivity with respect to baseline (e.g. structural metric 3, weighting function 1 with zero mean and the baseline solution as reference).

<begin file e4d.inp> this line is not included in the file

```
4                               run mode
sp.1.node                       mesh node file
```

baseline.sig.srv	baseline survey file
baseline_sp.sig	starting conductivity value
sp.out	output options file
sp_tl.inv	inversion options file
baseline_sp.sig	reference model file
surveys.txt 2	survey list file, using previous model as the starting model

<end file e4d.inp> this line is not included in the file

<begin file sp_tl.inv> this line is not included in the file

2	number of constraint blocks
1	constrain zone 1 with this block
8 1.0 1.0 1.0	use structural metric 8 (spatial difference of temporal differences)
1 10.0 0.15	use weighting function 1 with a large mean value to impose smoothness constraints
1 2	apply the constraints across the boundary with zone 2
‘pref’	use the previous solution as reference
1.0	relative weight for this constraint
2	constrain zone 2 with this block
8 1.0 1.0 1.0	use structural metric 8
1 10.0 0.15	use weighting function 1 with a large mean value to impose smoothness constraints
0	do not link to any zones (already linked to zone 1 in the block above)
‘pref’	use the previous solution as reference
1.0	relative weight for this constraint
100 0.05 0.5	start beta at 100, converge if change in obj. function <= 5% (since up_opt =3 below)
1.0	target chi-squared value
30 50	minimum maximum number of inner iterations
0.00001 1.0	min and max allowable conductivity
3	do not reduce beta
1 3.0	use data culling with if the weighted residual >= 3.0 deviations from the mean

<end file sp_tl.inv> this line is not included in the file

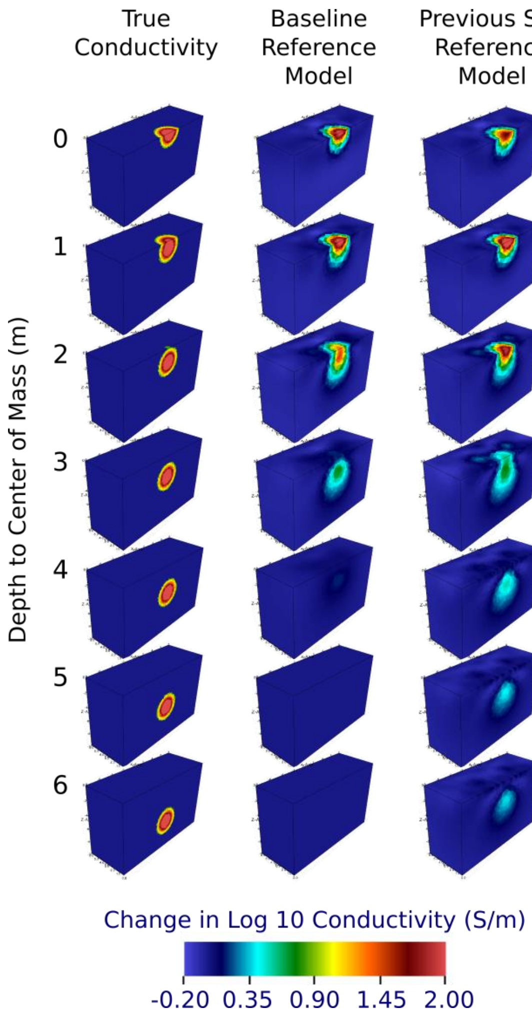


Figure 9.6. Comparison between true conductivity and imaged conductivity using the baseline solution and previous solutions as the reference model with transient smoothing constraints (structural metric 8).

10.0 bx Utility Program

The E4D distribution is supplied and compiled with a utility program that enters E4D mesh generation, simulation, and inversion results into an EXODUS II formatted file to facilite viewing using VisIt. The program is named ‘bx’ and is placed in <e4d_dir>/bin upon successful compilation, where <e4d_dir> is the base installation directory. The bx program reads a list of command line specifications, and based on those specifications, reads E4D generated files as input and outputs a corresponding EXODUS II file. The command line sequence for bx is as follows:

```
bx  <in_opt>   <mesh_pre>  <file_opt>  <out_fil>  <t_stamp>
```

where each of the command line options is described in the table below.

Command line option	Type	Description
<in_opt>	string	*<in_opt>* is a flag describing how the input files are specified, and whether the are to be included in a new exodus file or appended to an existing exodus file. The options are are as follows: *-f* add a single potential or conductivity file to a new exodus file *-af* add a single potential or conductivity file to an existing exodus file *-l* add a list of potential and/or conductivity files to a new exodus file *-al* add a list of potential and/or conductivity files to an existing exodus file If the *–af* or *–al* options are chosen, a new exodus file named *<out_fil>* will be created and overwritten if it exists.
<mesh_pre>	string	*<mesh_pre>* is the prefix to the mesh files. For example, if the mesh file names are *mesh.1.node*, *mesh.1.ele*, *mesh.1.neigh*, *mesh.1.face*, and *mesh.trn*, then *<mesh_pre>* is *mesh.1* . If the file append options –af or –al are specified, then *<mesh_pre>* must specify the same mesh that was used to build the original exodus file. Also, all input potential must have the same number of nodes as specified in the node file *<mesh_pre>.node*. All conductivity files must have the same number of elements as specified in the element file *<mesh_pre>.ele*.
<file_opt>	string	*<file_opt>* specifies a single potential or conductivity file if the *–f* or *–-af* values are specified for *<in_opt>*, or a file containing a list of potential and conductivity files if *–l* or *–al* is specified for *<in_opt>*. The first row of the list file contains the number of files in the list, and the remaining specify the input file names, followed by a time stamp. For example, a list file with three conductivity files would be: *3* number of input files specified below *File1.sig 10.0* first file and time stamp *File2.sig 20.0* second file and time stamp *File3.sig 30.0* third file and time stamp

Command line option	Type	Description
<out_fil>	string	*<out_fil>* is the name of the exodus file to be created (or appended to) by *bx*. If *–af* or *–al* is specified for *<in_opt>*, then a new file is created, or overwritten if it exists.
<t_stamp>	real	*<t_stamp>* is the time stamp specification for a file, and is only required if the *–f* or *–af* options are specified for *<in_opt>*. If –l or –al are specified for *<in_opt>*, then the time stamps for each file are specified in the listing file as described above.

Upon execution, bx enters the finite element mesh and specified files into specified exodus file. In addition to the mesh, four variables are written to the file. The variable names are visible when the exodus file is opened in VisIt, and may be specified for visualization. The variables are real_conductivity, complex_conductivity, real_potential, and complex_potential. The complex variables are associated with the SIP modeling and inversion modes, which are not discussed in this document.

VisIt is a highly flexible visualization software capable of performing and visualizing a number of transforms on conductivity and potential fields, or any specific function of those fields. Users are encouraged to review the VisIt documentation for details.

Appendix A

File Formats

Appendix A

File Formats

A.1 Input File Formatting Requirements

Input files are ascii text files that provide information necessary for E4D to execute in a particular mode. Each input file contains one or more records on a single line, where a record is a required number or text value. Input files may also contain comments after required records on each line or at the end of the file, but they are not required for execution. Records required by each input file are described in the file format descriptions following this section. All input files have the same four formatting requirements, which are:

1. Records must be listed in the order stated in the format descriptions.
2. Blank lines are permitted between lines containing records.
3. Comments are not permitted on lines that do not contain records.
4. Comments are permitted 1) on any line that contains records, after all required records for that line are listed or 2) at the end of the input file (i.e., after all record lines have been entered).

A.2 Run Configuration Input File: *e4d.inp*

The run configuration must by named *e4d.inp*. It is the first file read by E4D, and contains the names of input files and other options necessary for E4D to execute a particular run mode. For reference, each row of the 'Variable' column in the table below shows all records required for the corresponding rows in *e4d.inp*. Note that the requirements change slightly depending on the run mode. Therefore, requirements specific to each run mode are given in the documentation for that mode.

Variable	Type	Description
run_mode	integer (string for analytic mode, see description)	Indicates the mode E4D should execute as follows: 1: ER mesh generation mode 2: ER forward run mode 3: ER static inversion mode 4: ER time-lapse inversion mode 21: SIP mesh generation mode 22: SIP forward run mode 23: SIP static inversion mode 31: ER tank mesh generation mode 32: ER tank boundary forward run mode 33: ER tank boundary static inversion mode 34: ER tank boundary time-lapse inversion mode analytic or ANALYTIC or Analytic : forward analytic mode for homogeneous halfspace
mesh_file	string	If mode = 1, *mesh_file* specifies the name of the mesh configuration file. In all other modes *mesh_file* specifies the name of the node (or optionally the element)

Variable	Type	Description
		file generated by tetgen. Unless renamed by the user, the node and element files will end with *.1.node* and *.1.ele* respectively (see mesh generation section)
survey_file	string	Specifies than name of the survey file. Required only if mode > 1.The survey file provides electrode positions and describes how those electrodes are used to collect measurements. It also provides the observed data for inverse modes.
conductivity_file	string	Specifies the name of the conductivity file (real or complex). Required only if mode > 1. The conductivity file specifies the conductivity for each element in the computational mesh.
out_opts_file	string	Specifies the name of the output options file. Required only if mode > 1. The output option file provides E4D with instructions concerning how to report simulation and inversion results.
inv_opts_file	string	Specifies the name of the inversion options file. Required only in inversion modes. The inversion options file provides E4D with instructions concerning how to execute the inversion, including model constraints and convergence criteria.
ref_mod_file	string	Specifies the name of the reference model file, and must be present in all inverse mode. The reference model file must exist only if it is used to constrain the inversion, as specified in the inversion options file.
tl_list, tl_opt	string	Specifies the name of the time-lapse survey list file, and the time-lapse reference model option. The time-lapse survey list file provides a list of sequential survey files constituting a time-lapse survey. The time-lapse reference option specifies whether the baseline inversion or the previous inversion should be used as a reference to constrain the current time step.

A.3 Mesh Configuration File

The mesh configuration file is specified on the second line of *e4d.inp* in all mesh generation modes. It is used to provide E4D with instructions concerning how to build the mesh. A complete description of the mesh configuration file is provided in the specific documentation for mesh generation mode (mode 1).

A.4 Survey File

The survey file provides the locations of electrodes, describes how electrodes are used to produce measurements, and lists measurement values and standard deviations. There are no naming requirements for the survey file. However, example survey files provided with the E4D distribution are appended with ***.srv***. In forward modeling modes, the survey file is used to produce a simulated survey based on the conductivity distribution specified in *e4d.inp* (see documentation for mode 2, mode 22, and mode 32 for details). In inversion modes, the survey file provides the observed data used to estimate the subsurface conductivity distribution. Each row of the 'Variable' column in the table below shows all records required for the corresponding rows in the survey file.

Variable	Type	Description
n_elec	integer	specifies the number of electrodes
e_ind ex ey ez eflag	see description	One row is included for each of the *n_elec* electrodes, where: *e_ind* is the electrode index, labelled consecutively from 1 to *n_elec* (integer) *ex* is the x-position of the electrode (real value) *ey* is the y-position of the electrode (real value) *ez* is the z-position of the electrode (real value) *eflag* is 0 if the electrode is below the surface, 1 if the electrode is on the surface (integer) NOTE: If infinite conductivity boundaries (e.g. metallic) are included in the mesh, then one electrode must be specified for each boundary (e.g. each negative node boundary number in the mesh). In this case, *e_flag* is the boundary number on which the electrode lies. For example, if electrode 11 is the electrode corresponding to the infinite conductivity boundary whose nodes have a flag of -1, then the row for this electrode would read 11 *ex ey ez* -1, where *ex ey* and *ez* indicate the position of any node with a boundary flag of -1. Infinite conductivity boundaries included in the mesh must have a corresponding electrode included in the survey file, regardless of whether or not that boundary is used as a current source or sink. If an electrode is not specified for a particular boundary, that boundary will not be modeled as an infinite conductivity boundary. Furthermore, these electrodes must be specified in the electrode list after the point electrodes are specified.
n_meas	integer	specifies the number of measurements
For ER modes only *m_ind a b m n v_obs v_std*	see description	One row is included for each of the *n_meas* measurements, where: *m_ind* is the measurement index, labelled consecutively from 1 to *n_meas* (integer) *a* is the electrode index of the positive current electrode (integer) *b* is the electrode index of the negative current electrode (integer) *m* is the electrode index of the positive potential electrode (integer) *n* is the electrode index of the negative potential electrode (integer) *v_obs* is the observed transfer resistance (real) *v_std* is the standard deviation of the transfer resistance (real)
For SIP modes only *m_ind a b m n v_obs v_std i_obs* i_std**	see description	One row is included for each of the *n_meas* measurements, where: *m_ind* is the measurement index, labelled consecutively from 1 to *n_meas* (integer) *a* is the electrode index of the positive current electrode (integer) *b* is the electrode index of the negative current electrode (integer) *m* is the electrode index of the positive potential electrode (integer) *n* is the electrode index of the negative potential electrode (integer) *v_obs* is the real component of the measured transfer resistance (real) *v_std* is the standard deviation of *v_std* (real) *i_obs* is the complex component of the measured transfer resistance (real) *i_std* is the standard deviation of *i_obs* (real)

A.5 Conductivity File

Conductivity files are used to specify the conductivity (either real or complex) of each element in the computational mesh. Conductivity files are produced during mesh generation and inversion, but can also be user generated. The order of conductivity values listed in the conductivity file is the same order as the listing of the elements in the elements file produced by tetgen. There are no naming requirements for the conductivity file, but they generally end with *.sig* in the E4D documentation and examples. The format of the conductivity file is given below.

Variable	Type	Description
n_sigma n_col	integer	*n_sigma* specifies the number of conductivity values in listed, and must be equal to the number of elements in the mesh. *n_col* specifies the number of columns in the file, which is 1 in ER modes, and 2 in SIP modes (see below)
For ER modes only *sigma*	real	Specifies the conductivity values for each element. One row is recorded for each of the *n_sigma* values.
For SIP modes only *sigma isigma*	real	*sigma* specifies the real component of the conductivity. *isigma* specifies the complex component of the conductivity. One row is recorded for each of the *n_sigma* values.

A.6 Output Options File

The output options file provides E4D with instructions concerning how to report observed vs. simulated data values, and whether to construct potential files for specified current injection configurations. The format of the output options file is as follows (each row of the table provides variable requirements for the corresponding row in the output options file):

Variable	Type	Description
dp_flag	Integer (0 or 1)	If 0: do not print the simulated data file after each forward run If 1: print the simulated data file after each forward run
dp_file	string	Name of the simulated data file. The simulated data file reports the measurement configurations, observed, and simulated data values.
n_pot	integer	Number of potential distribution files to construct after each forward simulation.
pot_i	integer	Index of the measurement for which to construct the simulated potential distribution file. This index references a corresponding measurement index *m_ind* specified in the survey configuration file. One row is included for each of the *n_pot* potential files to be output.

A.7 Inversion Options File

The inversion options file specifies inversion options including regularization parameters, regularization weighting, convergence criteria, iteration options, line search options, and data culling options. The format of the inversion options file is given below. For a more detailed description of the options, see the individual documentation and examples for each inversion mode (mode 3, mode 4, mode 23, mode 33, and mode 34).

Variable	Type	Description
n_reg_blocks	integer	Specifies the number of regularization option blocks listed below.
zone	integer	Specifies the zone number for which this block operates. This is the beginning of a regularization block.
s_met wx wy wz	see description	*s_met* is an integer that indicates which structural metric to use for the regularization *wx wy* and *wz* are positive real values indicating the relative weighting in the x, y, and z directions respectively. Note: wx, wy, and wz are not used by every structural metric. In cases where they are not used, they are ignored by E4D during regularization. However, they must be present in the inverse options file, regardless of which structural metric is used.
fw fw_mn fw_dv	see description	*fw* is an integer value indicating which re-weighting function to use during the inversion *fw_mn* is a real value indicating the mean of the re-weighting function *fw_dv* is a real value indicating the standard deviation of the weighting function
n_links l1 l2 ... ln_links	integer	*n_links* is the number of zones to which this zone is linked. *l1 l2 ... ln_links* are the zone numbers to which this zone is linked. Elements bounding the zone regularized by this block will be connected to this block (through the regularization constraints specified in this block) if the zones to which those elements belong are specified in the link list. A zone may not be linked to itself. In other words, the each of the linked zones *l1 l2 ... ln_links* listed must be different from the variable *zone*.
v_ref	see description	*v_ref* specifies the reference value used for this regularization block. If *v_ref* is a real number, then *v_ref* is used as the reference value. If *v_ref* is specified as "*REF*" or "*Ref*" or "*ref*", then the reference model specified in *e4d.inp* is used to provide the reference values for this regularization block. Note that not all structural metrics use a reference value. However, *v_ref* must always be specified in the inverse options file. If *v_ref* is not used by the specified structural metric, then it is ignored by E4D.
w_rel	real	*w_rel* is the relative weight applied to the regularization constraints specified in this regularization block. This is the end of a regularization block. There must be *n_reg_blocks* regularization blocks specified in the inversion options file.
beta min_red beta_red	real	*beta* is the global regularization weighting value at the beginning of the inversion *min_red* is the minimum fractional decrease in the objective function between outer iterations that may occur before *beta* is reduced. *beta_red* is the *beta* reduction factor. If the fractional decrease in the objective function between outer iterations is less than or equal to *min_beta*, then *beta* is reduced by a factor of *beta_red* for the next iteration.
chi_target	real	*chi_target* is the chi-squared value at which the inversion is considered to have converged
miniter maxiter	integer	miniter is the minimum number of inner iterations (i.e. CGLS iterations) to execute before updating the solution maxiter is the maximum number of inner iterations to execute before updating the solution
min_sig max_sig	real	minimum and maximum conductivity values allowed by the inversion. Note these values do not constrain the inversion, and are provided only as a safety mechanism to ensure the forward solutions remain stable. Maximum and minimum conductivity constraints can, and when possible should, be specified as regularization constraints

Variable	Type	Description
		within the regularization block section of the inversion options file.
up_opt	integer	If *up_opt* = 1 a line search to estimate the optimum *beta* value is executed if *up_opt* = 2, no line search is executed and beta reduces as specified by *min_red* and beta_red if *up_opt = 3*, beta remains at its starting value, and the inversion converges when the reduction in the objective function is less than *min_red* or the target chi-squared value is reached, whichever comes first. The beta line search has not been quality tested, therefore E4D defaults to up_opt = 2 when up_opt = 1 is specified.
n_sfacs	integer	Specifies the number of line search scaling factors to test in order to estimate the optimal scaling factor. If *up_opt*=2, this value is not used by E4D, and should not be included in the inverse options file.
sfac1 ... *sfac_n_sfacs*	real	Line search scale factors to test during the line search. If up_opt = 2, these values are not used by E4D, and should not be included in the inverse file.
cflag cdev	see description	*cflag* is an integer value of 0 or 1. If *clag* = 1 then data outlier removal is implemented. If *cflag* = 0 then all data will be used to constrain the inversion at every iteration. *cdev* is a positive real value specifying the outlier removal standard deviation. If the residual error of any datum is greater than *cdev* standard deviations from the mean residual error and *cflag* = 1, then that datum is not used to constrain the inversion in next iteration. Outlier conditions are checked at every iteration, so a particular datum may be removed for one iteration and included in the next, and vice versa

A.8 Structural Metric Codes

A structural metric is defined herein as an equation describing a relationship between the conductivity of a target element and 1) the conductivity of the target elements neighbor, 2) a specified reference conductivity value (see *v_ref* in the inverse options file format above), or 3) the conductivity of the corresponding element specified in the reference model file. When a particular structural metric is specified for a zone, E4D applies that metric to every element within that zone, and optionally to elements bounding that zone if specified in the constraint block. E4D attempts to minimize each structural metric, which imposes some desired structure in the inverse solution (see *E4D* theory guide). E4D determines the conditions under which to apply the constraints according to the weighting function specified for each structural metric. Thus, the behavior of each structural metric in terms of constraining the inversion is also dependent upon the corresponding weighting function for that metric. In the table below, we provide the equations describing each structural metric. In the next section we provide the weighting functions, and follow-up with several examples of how structural metric/weighting function combinations can be used to provide the inversion with information concerning the subsurface conductivity structure.

Structure metric code (*s_met*)	**Equation**
1	X = m_t - m_n
2	X = |m_t - m_n|
3	X = m_t - v_ref_t
4	X = |m_t - v_ref_t|
5	X = m_t - m_n (with anistropic weighting)
6	X = |m_t - m_n| (with anistropic weighting)

7	X = (m_t - v_ref_t) - (m_n - v_ref_n)
	X = Dm_t - Dm_n

8	X = |(m_t - v_ref_t) - (m_n - v_ref_n)|
	X = |Dm_t - Dm_n|

9	X = m_t - m_n (applied only at boundary)
10	X = |m_t - m_n| (applied only at boundary)
Variables: X = value of the structural metric, and the independent variable in each weighting function. m_t = log conductivity of the target element m_n = log conductivity of the target elements neighbor v_ref_t = log conductivity of the target elements reference value, taken from either the constraint block or the reference model file v_ref_n = log conductivity of neighbors reference value, taken from either the constraint block or the reference model file Dm_t = change in log conductivity of target element with respect to v_ref_t Dm_n = change in log conductivity of neighbor element with respect to v_ref_n	

A.9 Weighting Function Codes

The primary purpose of each weighting function is to determine the conditions under which the corresponding structural metric should or should not be used to constrain the inversion. In essence, the weighting functions turn the structural metrics on and off and determine how much weight should be placed on minimizing the structural metric in the transition between on and off. The independent variable for each weighting function is the value of the structural metric (i.e., the value X in the structural metric code table above). Each of the four weighting functions is based on the normal and cumulative normal distribution functions and ranges between 0 and 1. If the weighting function evaluates to zero, then the structural metric is inactive and plays no role in constraining the inversion. If it evaluates to one, then the structural metric is fully active. The following table and figures show the equation and form of each weighting function.

Code	Equation
1	W_f=.5(1- erf((X-***mn***) / sqrt(2****sd2***))) (see Figure A.1)
2	W_f=.5(1- erf((X+***mn***) / sqrt(2****sd2***))) (see Figure A.2)
3	W_f=1-exp(-((X-***mn***)^2) / (2****sd***^{2})) (see Figure A.3)
4	W_f=exp(-((X-***mn***)^2) / (2****sd***^{2})) (see Figure A.4)
Variables: W_f = value of the weighting function where (0 <= W_f <= 1) X = value of the structural metric (see structural metric code table above) erf = error function ***mn*** = mean of the weighting function (see inversion options file format and Figure A.1-Figure A.4 below). ***sd*** = standard deviation of the weighting function (see inversion options file format and Figure A.1-Figure A.4 below).	

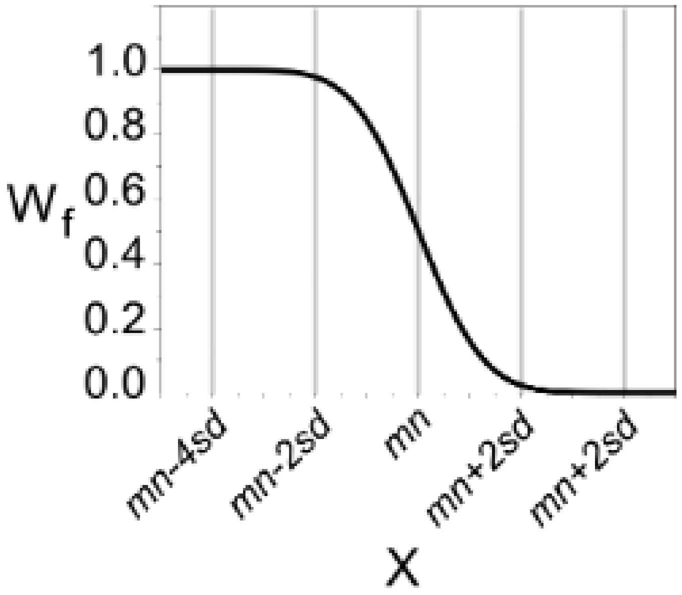


Figure A.1. Weighting function 1 causes E4D to begin to minimize X if the value of X drops below *mn+2sd*, reaching the full wieight if X drops below *mn-2sd*

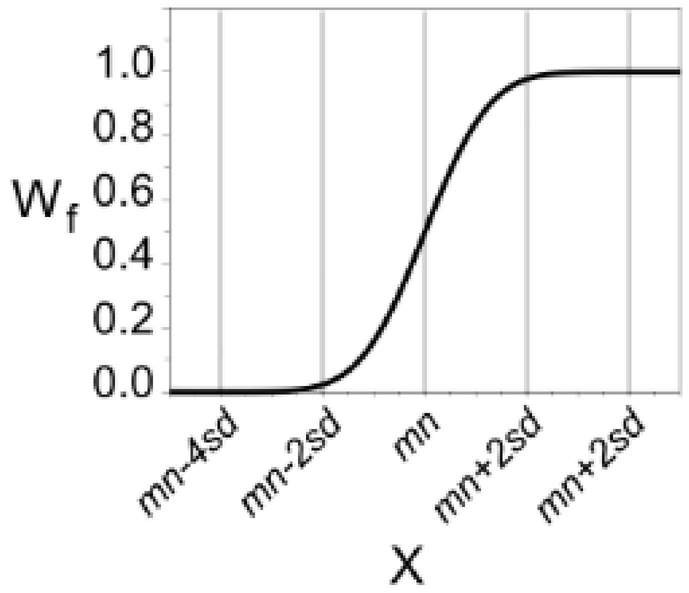


Figure A.2. Weighting function 2 causes E4D to begin to minimize X if the value of X rises above *mn-2sd*, reaching the full wieight if X rises above *mn+2sd*

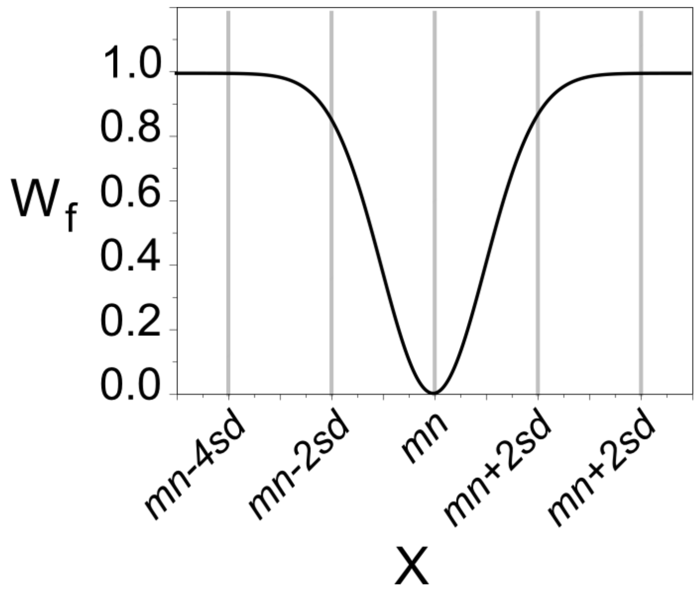


Figure A.3. Weighting function 3 causes E4D to begin to minimize X if the value of X deviates from mn, reaching the full wieight if X deviates from mn more than (approximately) 2*sd*

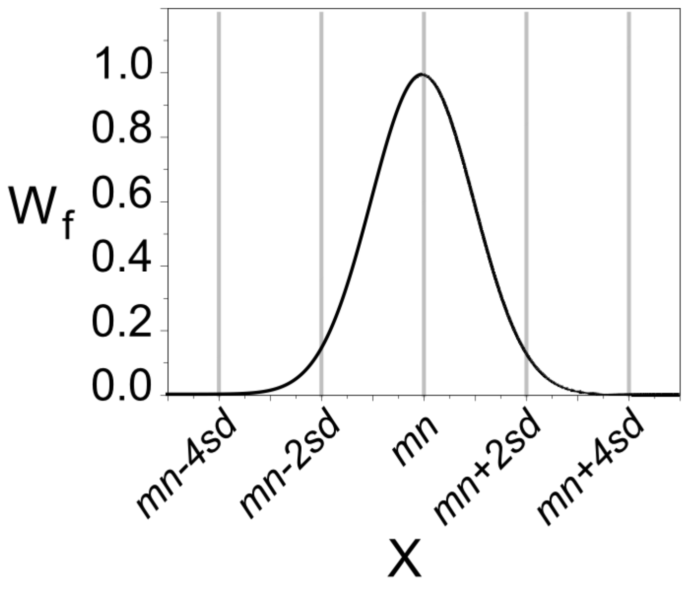


Figure A.4. Weighting function 4 causes E4D to begin to minimize X as the value of X approaches *mn*, reaching the full weight when X is equal to *mn*

A.10 Time-Lapse Survey List File

The time-lapse survey list file is used to specify the survey files used in a time-lapse inversion. The format is shown below. Details concerning the use of the time-lapse survey list file are given in the individual documentation for each time-lapse inversion mode (mode 4).

Variable	Type	Description
n_files	integer	Specifies the number of survey files listed
s_file t_stamp	see description	One row is included for each of the n_files, where *s_file* is the name of the survey file (string) *t_stamp* is a time stamp for the file (real)

A.11 Potential File

Potential files specify the electrical potential (real or complex) at each mesh node per ampere of current injected. They are optionally produced in forward run modes as specified in the output options file, and can be written to an exodusII file for visualization using bx.

Variable	Type	Description
nnodes n_col	integer	*nnodes* specifies the number of potential values in listed, and is equal to the number of nodes in the mesh. *n_col* specifies the number of columns in the file, which is 1 for ER modes and 2 for SIP modes (see below)
For ER modes only *pot_i*	real	Specifies the potential value for each node per ampere of current injected. One row is recorded for each of the *nnodes* values.
For SIP modes only *pot_i ipot_i*	real	*pot_i* specifies the real component of potential value for each node per ampere of current injected. *ipot_i* specifies the complex component of potential value for each node per ampere of current injected. One row is recorded for each of the *nnodes* values.

A.12 Simulated Data File

The simulated data file is similar to the measurement section of the survey file. It provides a comparison of the observed and simulated data, and is optionally output after each forward run.

Each row of the 'Variable' column in the table below shows all records printed in the simulated data file.

Variable	Type	Description
n_meas	integer	specifies the number of measurements
For ER modes only *m_ind a b m n av avs*	see description	One row is included for each of the *n_meas* measurements, where: *m_ind* is the measurement index, labelled consecutively from 1 to *n_meas* (integer) *a* is the electrode index of the positive current electrode (integer) *b* is the electrode index of the negative current electrode (integer) *m* is the electrode index of the positive potential electrode (integer) *n* is the electrode index of the negative potential electrode (integer) *av* is the observed transfer resistance for this measurement as specified in the survey file *avs* is the simulated transfer resistance for this measurement *Only included in SIP modes
For SIP modes only *m_ind a b m n av avs ai ais*	see description	One row is included for each of the *n_meas* measurements, where: *m_ind* is the measurement index, labelled consecutively from 1 to *n_meas* (integer) *a* is the electrode index of the positive current electrode (integer) *b* is the electrode index of the negative current electrode (integer) *m* is the electrode index of the positive potential electrode (integer) *n* is the electrode index of the negative potential electrode (integer) *av* is real component of the observed transfer resistance as specified in the survey file *avs* is the real component of the simulated transfer resistance *ai* is the imaginary component of the observed transfer resistance as specified in the survey file *ais* is the imaginary component of simulated transfer resistance