E4D tutorial

Alain Plattner

February 20, 2017

1 Introduction

E4D is a powerful electrical resistivity tomography program written by Tim Johnson at PNNL. It can be freely downloaded and installed from `https://e4d.pnnl.gov/`. The program's strength are that it is parallelized and can therefore solve large problems (many inversion cells) very quickly. The disadvantage of E4D is that it is harder to learn than for example BERT.

E4D uses several types of configuration files which we will discuss in the following. The Matlab/Octave scripts that come with this tutorial will be helpful to generate the content for many of these input files.

In this tutorial we will go step by step, starting with generating a simple mesh and moving through calculating inversions. Preparing the input files for complex meshes is the hardest part of using E4D, so we will go step by step from simple (but usually useless) meshes to complicated meshes.

Generally the work flow for using E4D will be to first construct a mesh, and then run the inversion on a prepared mesh. E4D always requires an input file named `e4d.inp`. In this input file, the user specifies what needs to be done (e.g. mesh generation, inversion), and the names of the configuration and data files.

2 Mesh generation

On your desktop, or in any folder you would like to work in, create a file named `e4d.inp`. You can do that for example with emacs by running in your folder

```
emacs e4d.inp
```

2.1 My first mesh

To make e4d create a mesh, the first line of `e4d.inp` needs to be the number 1. The second line needs to be the name of our mesh configuration file. Let's call it `myfirstmesh.cfg`. So make your `e4d.inp` text file look like

```
1
myfirstmesh.cfg
```

Of course, we now need to create the mesh configuration file `myfirstmesh.cfg`. To do that, create it again using emacs

```
    emacs myfirstmesh.cfg
```

This file will now be a bit more complicated than the input file. A mesh file will always consist of the following components:

- The mesh quality (how deformed can the tetrahedra be), and maximum tetrahedra volume
- the bottom elevation of the mesh
- some information on how to build the mesh (this will always be the same so don't worry about this now)
- The points around which the mesh will be generated including electrode positions, topography, boundary points for the internal and external zones, electrode depth points for mesh refinement around the electrodes, etc.
- Internal planes to delineate between mesh zones
- holes (I have not yet worked with holes in the mesh so we will skip this in this tutorial)
- zone descriptions
- information about writing a paraview/exodus file

Let's start with the simplest possible configuration. We first need to learn how we write points. A point always has the form

pointnumber xcoord ycoord zcoord type

type can be subsurface point (0), or surface point (1) or external boundary point (2).

The external boundary points define the outside boundary of our region so we will always need them.

Let's write our `myfirstmesh.cfg` file. Let's make the mesh quality 1.3 and the maximum volume 10^{12}. The surface will be defined by the external boundary points (which by definition are on the surface), and the bottom will be at -20 m. We want the external boundary to be at at four points (-10/-30/0), (50/-5/1), (40/70/-1.2), and (-20/60/0). The units of these coordinates are meters.

So the lines in our file `myfirstmesh.cfg` will be:

```
1.3 1e12   (mesh quality, max volume)

-20   (bottom elevation of mesh)
1
"tetgen"
"triangle"

4   (number of points)
1 -10 -30 0 2   (first boundary point)
2 50 -5 1 2  (second boundary point)
3 40 70 -1.2 2  (etc)
4 -20 60 0 2

0   (no internal planes)

0   (no holes)

1   (one zone)
1 0 0 -2 50 0.01   (point in zone, max mesh size this zone, conductivity)

1   (save mesh as exodus file)
"bx"  (use the program "bx")

1   (mesh translation option)
```

If you want you can omit the comments in parenthesis. These will not be used by the program but it may make it easier for you to edit the mesh configuration file.

In this file we needed to define parameters such as maximum tetrahedra volume for each zone (at this point we only have one zone), and the conductivity. To do this we needed to pick a random point within our zone, we picked (0/0/-2), and set the maximum tetrahedra volume, we selected 50 m^3, and for the conductivity, we selected 0.01 S/m. These values are for zone 1, therefore the line read

1 0 0 -2 50 0.01

To create the mesh, run in your command line

```
    e4d
```

The program will automatically look in the input file `e4d.inp` and use the mesh configuration file you specified there and will create a lot of auxiliary files. Open the resulting file `myfirstmesh.exo` using paraview.

Exercise: Play around with the maximum mesh size for the zone (which we had originally set to 5) and see if you can create a finer mesh.

2.2 Adding electrodes

The mesh we created in Section 2.1 looks great but will be of limited use when we want to start running inversions. The minimal thing that we need to do is to add nodes where we have electrodes.

Let's assume the simple case where we have only 3 electrodes, all on the surface, at the points (0/0/0), (0/1/0.5), and (0/2/0.2). We will add these points to the mesh simply by including them in our list of points. Since our electrodes are all on the surface, their *type* has the value 1. Our new list of points therefore looks like

```
7   (number of points)
1 -10 -30 0 2   (first boundary point)
2 50 -5 1 2  (second boundary point)
3 40 70 -1.2 2  (etc)
4 -20 60 0 2

5 0 0 0 1   (first electrode)
6 0 1 0.5 1  (second electrode)
7 0 2 0.2 1  (third electrode)
```

The rest of the file `myfirstmesh.cfg` stays the same as before. You do not have to insert an empty line between the external boundary points and the electrodes, but it may make it easier when you have many electrodes.

Run in your command prompt

```
    e4d
```

and open the resulting mesh again with paraview. You will notice that the mesh looks a bit different. There are smaller tetrahedra where the electrodes are.

When running electrical resistivity inversions, you will likely run into problems when the mesh is too coarse around your electrodes. To create a fine mesh around the electrodes, we will add depth points underneath the electrodes. These depth points will have the same x and y coordinates as the electrodes but are a few cm (or m) below the electrodes (the depth will depend on how fine you will want the mesh around your electrodes). Let's pick 1 cm in this example (our electrodes are only 1 m apart). Because these points are below the surface, their *type* is 0. So your new list of points will be

```
10   (number of points)
1 -10 -30 0 2   (first boundary point)
2 50 -5 1 2  (second boundary point)
3 40 70 -1.2 2  (etc)
4 -20 60 0 2

5 0 0 0 1   (first electrode)
6 0 1 0.5 1  (second electrode)
7 0 2 0.2 1  (third electrode)

8 0 0 -0.01 0   (first electrode depth point)
9 0 1 0.49 0  (second electrode depth point)
10 0 2 0.19 0  (third electrode depth point)
```

When you run

```
    e4d
```

with this new `myfirstmesh.cfg` file, and you look at the result using paraview, you will see that the mesh around the electrodes is very fine.

Congratulations, you created your first mesh that could be used for electrical resistivity tomography. However, to get good results with reasonable computing power, there is more to learn.

2.3 Internal boundaries

One of the problems of electrical resistivity tomography is that our mesh ends somewhere. This of course is not the case in nature. The world does not just end. This boundary is therefore artificial.

To avoid problems with this artificial boundary, we will usually have the external boundary far away, by placing the external boundary points (the *type* 2 points) far away.

But this leads to the problem that we either can only have a coarse grid away from the electrodes, or we need to fill the entire volume with a fine mesh.

Luckily, E4D allows us to create an internal zone with fine mesh (where we have the electrodes and good coverage), and an external zone with coarse mesh (which allows us to push the external boundary far away). In fact, we can have as many zones as we want. We could make a very fine mesh zone very close to the electrodes, then a coarser mesh zone a bit further away, and finally a very course mesh zone surrounding everything.

In this example we will only create two zones, an external zone and an internal zone, but you can use what you learned here to create more complex cases.

To create zones we need to create zone boundaries and to create zone boundaries we need to first pick zone boundary points.

Let's say we want the inner-zone boundary points (-2/-2/-0.3), (-2/4/0.2), (2/4/0), and (2/-2/0.1). **Note that these points are in clockwise order**. It is important that you enter your points in clockwise (or counter-clockwise) order, otherwise the boundary generation will not work.

Writing your own internal boundaries can be difficult. To make life easier, I wrote the Matlab/Octave program `makezone.m` to do this for us. It's in the folder `m-files` in the same repository where you found this tutorial. Run, in Matlab or Octave

```
    >> help makezone
```

to see what input is required. The simplest way of using it is to give it the surface points (or shallower subsurface points, if you want a completely buried zone), the depth of the zone under the surface (let's say we want the internal zone to end at 5 m depth), and the number of the first point (we already have 10 points in our list, so the first point will be 11). Let's say we want the text output to be stored in the file `myfirstzone.txt`. To do all of this, run in Matlab or Octave

```
    >> makezone([-2,-2,2,2],[-2,4,4,-2],[-0.3,0.2,0,0.1],-5,11,'myfirstzone.txt')
```

The resulting file `myfirstzone.txt` will contain the points with point numbers between 11 and 18, and information about internal planes. Add the points to your list of points in your `myfirstmesh.cfg` file (don't forget to update the total number of points) and replace the line saying that there are no internal planes with the internal planes you generated.

So the points part of your file `myfirstmesh.cfg` will now be

```
18   (number of points)
1 -10 -30 0 2   (first boundary point)
2 50 -5 1 2  (second boundary point)
3 40 70 -1.2 2  (etc)
4 -20 60 0 2

5 0 0 0 1   (first electrode)
6 0 1 0.5 1  (second electrode)
7 0 2 0.2 1  (third electrode)
```

```
8 0 0 -0.01 0   (first electrode depth point)
9 0 1 0.49 0  (second electrode depth point)
10 0 2 0.19 0  (third electrode depth point)

11 -2.000000 -2.000000 -0.300000 1
12 -2.000000 4.000000 0.200000 1
13 2.000000 4.000000 0.000000 1
14 2.000000 -2.000000 0.100000 1

15 -2.000000 -2.000000 -5.000000 0
16 -2.000000 4.000000 -5.000000 0
17 2.000000 4.000000 -5.000000 0
18 2.000000 -2.000000 -5.000000 0
```

And the internal planes part will now be

```
5 number of internal planes
4 10
11 12 16 15
4 10
12 13 17 16
4 10
13 14 18 17
4 10
11 14 18 15
4 10
15 16 17 18
```

Note that here we used the internal boundary number 10, which I set as default. If you have several internal zones, then you may want to use different internal boundary numbers (one for each zone).

We will also need to add that we have a second zone. We do that by simply adding a line in the zones part of the file `myfirstmesh.cfg` and set the maximum mesh volume and conductivity for this zone. For this we need to find a point in the external zone. In our case, the point (10/10/-10) should do. The point (0/0/-2) is still within our internal zone. Let's set, for the internal zone, the maximum mesh volume 1 m^3 and for the external zone the maximum mesh volume 10^5 m^3. We set 0.01 S/m for the conductivity of both zones.

Hence our new "zones" part of the file `myfirstmesh.cfg` will be

```
2  (two zones)
1 0 0 -2 1 0.01  (internal zone with fine mesh)
2 10 10 -10 1e5 0.01  (external zone with coarse mesh)
```

After updating your file `myfirstmesh.cfg`, run on the command line

```
e4d
```

and look at the new mesh using paraview.

You will see that your new mesh has two different zones and the mesh of the internal zone is much finer than the mesh of the external zone.

Congratulations, you now have the knowledge needed to create electrical resistivity tomography meshes for quite general settings.

A note on topography: As you noticed, all of our points had a z-axis value. This is how you define topography. As long as your point is either of *type* 1 or 2, the mesh generator will know that it is on the surface and will incorporate it into the topography. To incorporate complex topography, simply add all the topography points to your points list as *type* 1 surface points (don't forget to update the total number of points).

Exercise: Instead of a 4 point internal zone boundary, make a 5 point internal zone boundary. Make sure you enter the points in clockwise orientation.

Exercise: Download topography points from an online database (for example `http://opentopo.sdsc.edu/datasets`) and create a mesh for these topography points.

To make your life easier I wrote MATLAB/OCTAVE functions that can help with some of the tasks for mesh generation. You can find them in the folder `m-files`. For example, the function `elecs2points.m` will write electrode coordinates in the format that can directly be copy-pasted into your e4d mesh configuration file. The function `topodata2grid.m` can turn ASCII point-cloud data downloaded from `http://opentopo.sdsc.edu/datasets` into the format that can be copy-pasted into the e4d mesh configuration file (including downsampling).

3 Inversion

In order to invert ERT data we need:

- A mesh that has the electrodes incorporated (see Section 2)
- A data file .srv
- An inversion options file
- An output options file

We learn how to do these things using a very simple data set collected on Fresno State Campus. The data `LNWEN3D.ohm` is in the folder `tutdata`. The folder `m-files` contains MATLAB/OCTAVE functions to turn .ohm files into .srv files. Thomas Gunther's program DC2DInvRes, freely available from `resistivity.net` allows to transform most commonly used ERT data formats into .ohm (and allows for basic data quality control).

I suggest that you create a new folder for the mesh generation and inversion that we will do in this chapter.

3.1 Step 1: Create the basic mesh file

To do this we need to decide what area we want for the internal and external zone. When opening the data file LNWEN3D.ohm with a standard text editor we see that the 28 electrodes are all along the x-axis at a 0.5 m spacing with the first electrode at the origin (0/0/0) and the last electrode at (13.5/0/0). Let's make a mesh with external zone between the four points (-100/-100/0), (-100/115/0), (100/115/0), and (100/-100/0). For the internal zone we pick (-5/-5/0), (-5/5/0), (20/5/0), and (20/-5/0). Make the internal zone 17 m deep and the external zone 50 m deep. For the internal zone, let's make the maximum mesh size 5 m^3 and for the external zone 10^7 m^3.

Task: Create a mesh for the given external- and internal-zone boundary points. Let's call the mesh .cfg file `myfirstinvmesh.cfg`. You will also need to create a new `e4d.inp` file. In the `myfirstinvmesh.cfg` file, do not yet add the electrodes. We will use the functions in the folder `m-files` to make that part easier.

Hint: If the mesh generation fails, you can try making the internal zone a bit deeper or move the boundary points around. For example, try to create the mesh with the current settings but change the depth of the internal zone to 10 m. The mesh generation will (probably) fail.

3.2 Step 2: Create the data file and add electrode positions to mesh file

We will transform a given .ohm file (see `http://resistivity.net/`) into an .srv file and use the loaded electrode positions to add them to the mesh file.

The function `OHM2E4D.m` will do this for us. In addition, it will return the electrode positions. Look at the help page of `OHM2E4D.m` to understand how to use it. The provided data file `LNWEN3D.ohm` does not have any data errors (open it with a text editor, scroll down to the measured data: The column with the error is missing). This means we need to estimate the error (in percent). For this example, let's say it's 5 %.

So we need to run in Matlab/Octave

```
>> electrodes=OHM2E4D('LNWEN3D.ohm','LNWEN3D',5);
```

Let's also save the electrodes

```
>> save('electrodes.mat','electrodes');
```

When you open the file `LNWEN3D.srv` in a text editor, you will find that it looks similar to the original .ohm file, but the points are in E4D format, the data are numbered, and there is a column for data error.

Let's use `elecs2points.m` to create the electrode points for the mesh file. Remember we also need to create electrode depth points to refine the mesh around the electrodes. Let's put the depth points at 1 cm depth. For wider electrode spacings you can set this depth to a greater value.

Remember that our points need to be numbered and that we already have 12 points in our `myfirstinvmesh.cfg` file. So we need to set shiftpointnr to 12.

```
    >> elecs2points(electrodes(:,1),electrodes(:,2),electrodes(:,3),...
'elecs.txt',12,0.01)
```

creates a file `elecs.txt`, from which we can copy the electrode points and the electrode depth points and paste them into our mesh .cfg file `myfirstinvmesh.cfg`. **Remember to set the number of points to the new correct value**. We now have, alltogether, 68 points.

Run

```
e4d
```

to generate the mesh and look at it using Paraview. The resulting mesh should have two zones. The external zone should have a very coarse mesh, the internal zone should have a mesh that is coarse at the boundary to the external zone and gets finer toward the electrodes. At the electrodes, the mesh should be very fine.

We now need to write an inversion configuration file. The E4D user guide `https://e4d.pnnl.gov/Documents/E4D_User_Guide.pdf` has more detailed descriptions on how to write inversion options files. Here we create the simplest possible file for our two-zone situation.

I took the following inversion file pretty much directly from the E4D user guide. Open a text file named `myinvoptions.inv` and write in it

```
2                 # of constraint blocks

1                 constraint zone 1
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                 zone # this block constrains
2  1.0 1.0 1.0    s_met wx wy wz
1  10.0 .01       fw mn sd
0                 no links, these are given already
0.0               v_ref
1.0               w_rel

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
```

I recommend studying Section 3 of the E4D user guide to understand what these parameters mean and how changing them will affect the inversion outcome.

We now need to adapt the e4d input file `e4d.inp`. I recommend to save the mesh generation version as `e4d_mesh.inp` such that we can use it again later.

To run an inversion, e4d input file needs to look as follows:

```
3                      (run mode; 3 means inversion. 1 was mesh generation)
myfirstinvmesh.1.node  (the name of your mesh file with .1.node appended)
LNWEN3D.srv            (The data file)
'average'              (How to calculate starting conductivity)
myoutput.out           (Output options file; not important at this point)
myinvoptions.inv       (Inversion options file)
none                   (reference model)
```

To run the inversion calculation, simply running e4d will not work. These computations are usually work-intense and e4d is parallelized to make the calculations much faster. Therefore to calculate the inversion we need to run e4d in parallel mode. Let's start by using 5 processors:

```
mpirun -np 5 e4d
```

To run it with 10 parallel processors, simply change 5 to 10 in the above command. The number of processors you want to use is limited by the number of processors you machine has.

NEXT: HOW TO ADD TO EXO FILE, HOW TO TURN INTO RESISTIVITIES INSTEAD OF CONDUCTIVITIES, AND CAVEATS WHEN USING PARAVIEW TO VIEW THE RESULTS.