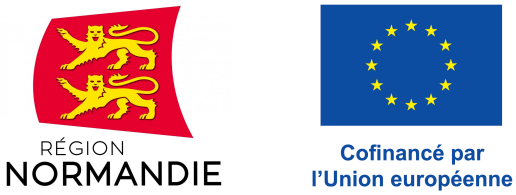
Basics on PyLGRIM

Contact : antoine.tonnoir@insa-rouen.fr, yannick.fargier@univ-eiffel.fr, cyrille.fauchard@cerema.fr, raphaël.antoine@cerema.fr

About

This work is partially supported by DEFHY3GEO project, this project is co-financed by the European Union with the European regional development fund (ERDF) and by the Région Normandie (France).



Dependancies

To be able to run the *PyLGRIM* codes, you will need to install `python3` and the following library :

1. scipy,
2. numpy,
3. and matplotlib.

This can be achieved with `pip install`. Also, it is necessary to have installed GMSH to build meshes from points Digital Elevation Models (DEM). You might need to modify line 374 of the file *generateMsh.py* in folder *1_DEM* to give the proper path to GMSH software on your computer. Note that all input data files (and some output files) are in the folder *Data*, and to visualize the results, you will need Paraview. This elements installed, you can safely use *PyLGRIM* codes.

A first example : synthetic case of study

Let us describe a first case of study. We will describe step by step the use of PyLGRIM to treat a simple example.

I. Mesh generation : We begin by running the script *generateMsh.py* in folder 1_DEM. Starting the script with command `python3 generateMsh.py`, the program asks wether :

1. you wish to build a 3D mesh with a topography described by a files containing (x, y, z) coordinates of the ground topography. In that case, you can specify the file name of the topography in variable *DEMfile* with appropriate path, and the localization of the electrode position in variable *ListElectrodeLoc* (this variable might contains a list of electrode positions files).
2. or you wish to build a 3D mesh with a topography described as a function $f(x, y)$. In that case, you need to specify the chosen function f in the beginning of the programme (function *topoF*) and the localization of the electrode position in the bloc line 140.
3. or you wish to build a 3D mesh using an already generated *.geo* file. This last case is useful is the situation where we only wish to modify some parameters of the mesh, without recomputing everything as in option 1 or 2.

For this example, we will choose the option 2, to build a synthetic case. The preprogrammed example contains one electrodes line made of 32 electrodes linking point $(0.15, 0, 0)$ to $(0.85, 0, 0)$. The topography is step like. At the end of the execution of the program, three files should have appeared in the folder *Data* :

— *meshDEMFlat.geo* : it corresponds to the geometric description of a cube used to generate the 3D meshes, see Figure 1 on the left,

— *meshDEMFlat.msh* : it corresponds to associated the mesh file, with flat topography, see Figure 1 in the center

— *meshDEM.msh* : it corresponds to the mesh file we are interested in, which take into account the topography, see Figure 1 on the right.

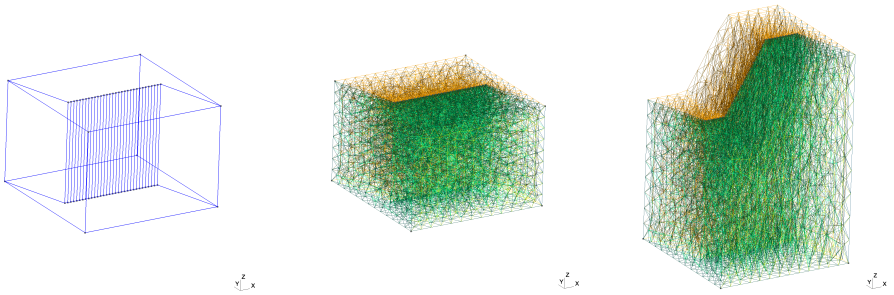


FIGURE 1 – Mesh files generated

To obtained these results, the parameters h and $coeff$ have been equal to 0.02 and 4 respectively (see lines 84 and 85). The first parameter, h, gives the mesh size around electrode position (for instance, if 0.1m separates each electrodes, then we have (around) 5 meshes between two electrodes). The second parameter $coeff$ modifies the mesh size far from the electrode lines. More precisely, at the corner of the computational domain, we have a mesh size of order $h \times coeff$ (consequently, if it is equal to 1 then the mesh size is the same everywhere). Also, the parameters Lx, Ly and $minZ$ are equal to 0.15, 0.5 and -1 respectively. The parameters Lx and Ly respectively extend the domain of interest (that is to say the line of electrodes) of a length $2Lx$ in direction x and $2Ly$ in direction y.

Remark 1. *When using the script* generateMsh.py*, it is recommended to check that the generated mesh file is correct. To do so, we shall open it in GMSH and check no error appears. Also, it is useful to know that for direct simulation, PyLGRIM can handle mesh up to* $\sim 150 \times 10^3$ *tetraedra (for* $\mathbb{P}_1$ *elements), whereas for inverse problem, it is better to limit to* 75×10^3 *tetraedra (for* $\mathbb{P}_1$ *elements).*

II. Electrical Resistivity experiment simulation : Once we have generate our mesh file, we can simulate Electrical Resistivity experiments. We will now use the second script `simuERI.py` in folder *2_SIMU_ERI*. To begin, in order to consider heterogeneous subsoil, we will run the script *genConductivite.py* with the command `python3 genConductivite.py`. This command will create in the same folder a file named *conductivite.data* containing the information of the subsoil conductivity. It also create a VTK file named *resistivity.vtk* represents the corresponding resistivity ($\rho = \frac{1}{\sigma}$) and which can be visualize with Paraview as shown on Figure 2. This medium corresponds to a two layers medium with two spherical anomalies (see left Figure). Note that modifying the file *genConductivity.py* enables to consider other heterogeneities. Note also that, for the moment, we generate conductivity file, though considering resistivity would be very easy.

Now we have defined the conductivity (or equivalently the resistivity), we can simulate ERI experiments. We execute the file *simyERI.py* with command `python3 simuERI.py` and the program first ask which order of finite element we want to use ($\mathbb{P}_1$ or $\mathbb{P}_2$). For this example, we will take 2. The

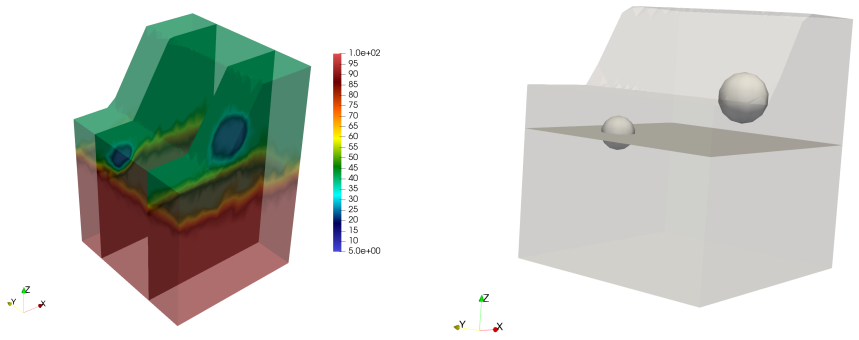


FIGURE 2 – Resistivity in the subsoil

program automatically adapts the mesh and then proposes three options of Boundary Conditions to considered unbounded subsoil :

1. Infinite elements, see paper,
2. Mixte boundary conditions,
3. or simple Dirichlet boundary conditions, that only can be accurate if they are far from the source terms.

In our example, let us take the first option. Then, the program compute the measurement matrix of the potential V at each electrode position different from the electrode injection position. At the end of the execution, the program shows this matrix and the apparent resistivity (à revoir, pour l'instant ce n'est pas exactement ça pour les deux lignes d'électrodes). Also, vtk files of the potential are written in the folder *2_SIMU_ERI/ResVTK/*. Thus, we can obtain representations of the potential as on Figure 3. Also, at the end of this second script, apparent resistivity is written

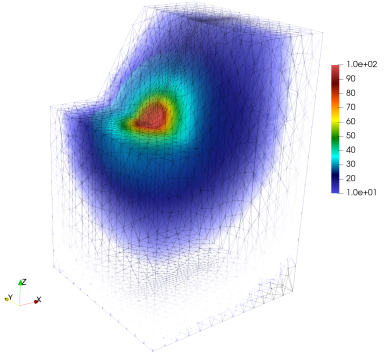


FIGURE 3 – Computed Potential injecting at one electrode.

in a file named *resistivityMes.txt* together with electrode positions in a file named *nodesInjLoc.txt* in the folder *Data/*.

III. ERI data inversion : Let us know details how to invert data. We consider here the same topography as before and the same subsoil resistivity. To generate data [1], we have used two situations :

— A first line of 32 electrodes equally spaced between $(0.15, -0.2, z_1)$ and $(0.85, -0.2, z_2)$, where z_1 and z_2 are the altitude of the corresponding points,

1. The generation of data can be skipped since the necessary files are already available in the folder *Data/*, we only need to generate the mesh file

— A second line of 32 electrodes equally spaced between $(0.1, 0.4, z_1)$ and $(0.75, -0.3, z_2)$.

In each case, we have generated a mesh file (with only one of this two lines), as explained in procedure **I.**. To do so, we have taken $h = 0.01$, $coeff = 16$, $Lx = 1.3$, $Ly = 1.3$ and $minZ = -1$. We have modified the line of electrodes description in the code between lines $140 - 185$. For each generated mesh, we have compute the apparent resistivity using procedure **II.** (note that we need to use each time *genConductivity.py* before launching *simuERI.py*). We used as before 2nd order finite elements and infinite elements for the boundary conditions on the artificial boundaries. To construct two different data files (and avoid overwriting), we have renamed the generated files (see lines 157 and 158 at the end of *simuERI.py*.

The data being generated, i.e. the four files *nodesInjLocStep1.txt*, *nodesInjLocStep2.txt*, *resistivityMesStep1.txt* and *resistivityMesStep2.txt*, we can now focus truly on the inverse problem. One feature of PyLGRIM is the possibility to invert several electrodes lines together. Let us generate the mesh file we will use. In *generateMsh.py*, we will consider the following parameters $h = 0.01$, $coeff = 15$, $Lx = 0.3$, $Ly = 0.3$ and $minZ = -1$. Also, in the electrode lines description, we consider the two lines described above. Then, the generated mesh is represented on Figure 4 as well as the .geo file.

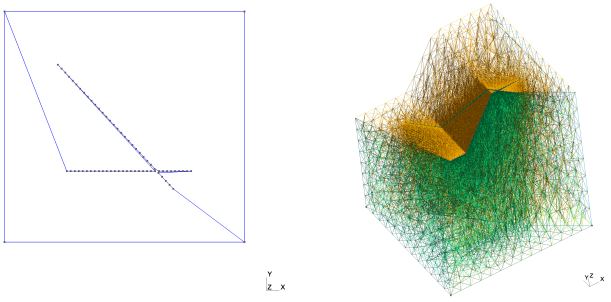


FIGURE 4 – Mesh files for inversion process.

Let us now run the last script in folder *3_Inv_ERI/* using command `python3 invERI.py`. As previously, we can choose the order of finite element. Here, to begin we will use first order elements. Then, the program asks the BC we wish to use. The choices are the same as for ERI simulations. We will use also the first choice. Also, it is proposed to use an a priori files. In our case, we do not have a priori file so we will choose '1. no'. Finally, the program proposes the regularization term we want to use :

1. Tichonov regularisation,
2. or smooth regularization (penalizing the gradient of the resistivity).

For this test, let us take the second option. And finally, the program asks the chosen minimization process :

1. Gauss-Newton algorithm,
2. or stepest gradient descent (which does not work at the moment).

To end this example, we make the first choice and... wait ! The program runs the computations. At each iterations of the minimization process, the program gives the residual error vs the residual error of the previous iterations. Also, it writes in folder *ResVTK/* the reconstructed conductivity and resistivity. On Figure 5, we have represented the reconstructed conductivity at iterations 1 and 2. The associated error are 9.8 and 0.38. The program has been interrupted at the end of the second iteration and we will restart it using the last result as an a priori guess. We have copy-paste the last file *resitivity2.vtk* in folder *Inv2Lig_o1_synth_coarsed/*. Now, we restart the program using this time 2nd order Finite Elements and an an priori guess (modifying line 35 of *invERI.py*

to called *Inv2Lig_o1_synth_coarsed/resitivity2.vtk*). Also on Figure 5, we have represented the reconstructed solution obtained starting with the a priori guess (second line) at iterations 0, 4 and 7.

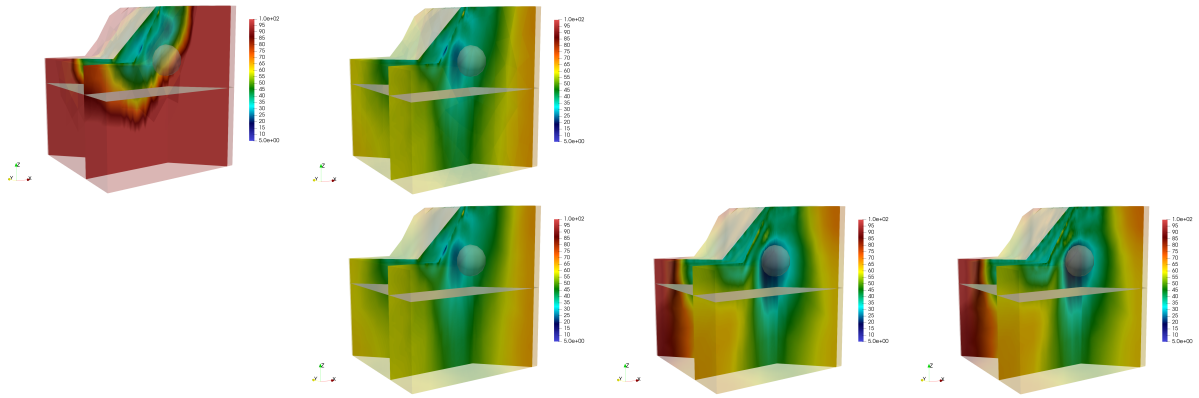


FIGURE 5 – Resistivity reconstruction at various iterations.

On Figure 6, we have represented the threshold mesh resistivity between 3 and 30 (on the left) and between 3 and 25 (on the right). On the first line, we present the result obtained at iteration 7 and considering the the two electrode lines. On the second line, the results are obtained using only one eletrode line. As we can see (and expect), we get better results considering the two lines (in particular for the spherical shape of the biggest spherical defect).

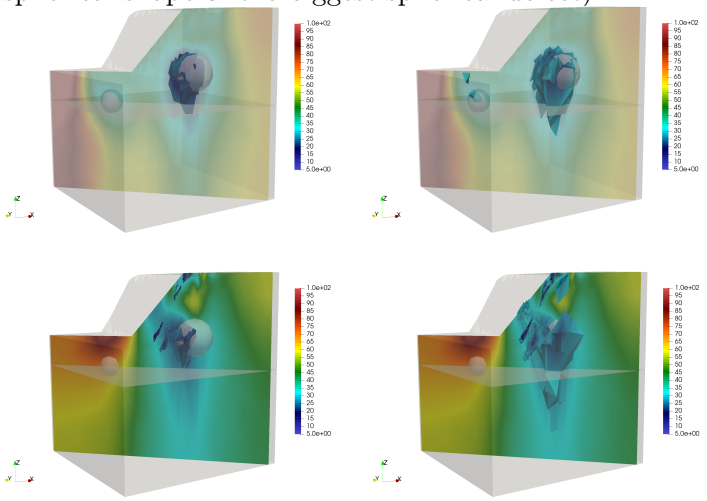


FIGURE 6 – Threshold of the resistivity reconstructed with two lines (top) and one line (bottom).

A second example : real case of study

Let us describe now a second case from real data.

I. Mesh generation : Once again, we begin with the script *generateMsh.py* in folder 1_MNT. This time, we will choose the first option and we will put the following parameters : $coeff = 15$, $h = 1.5$, $Lx = Ly = 10$ and $minZ = -30$ (see lines $84-91$). The preprogrammed example contains 3 lines of electrodes as represented on Figure 7. The computation takes few minutes due to the recursive algorithm used to determine the proper ordering of electrodes to mesh the domain.

The generated mesh file, *meshDEM.msh* (in folder *Data/*) contains almost 33×10^3 points and 195×10^3 elements (tetraedra). Though this is a suitable mesh for direct problem, we will modify

the parameter $coeff$ to get a coarsed mesh, more suitable for inverse problem. Let us then change the value of $coeff = 20$, $h = 2$, $Lx = Ly = 20$ and $minZ = -40$, and let us relaunch the program. This time, to avoid full recomputation, we will use option 3 (build a mesh from *.geo* file), which is much faster since we do not recompute electrode lines ordering. The new mesh file contains this time 96×10^3 elements, which will be suitable for inverse problem (using $\mathbb{P}_1$ elements). This procedure can be used to modified the parameters h, $coeff$, Lx, Ly, Lz and $minZ$ without recomputing everything.

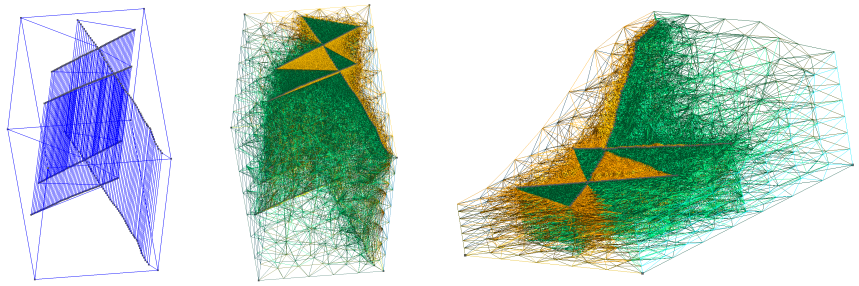


FIGURE 7 – Mesh files generated

III. ERI data inversion : For this example, we directly go to step 3 since we have experimental resistivity data. To take into account the VNC topography with the experimental measurements, we have used lines 47 and 75 to specified the electrode resistivity data and electrode positions respectively. Then, we run the script with the same options as in the previous example (and without a priori guess). The obtained conductivity at step $1, 2$ and 3 of the minimization process are represented on Figure 8, and the associated error are 1.64, 1.13 and 1.07.

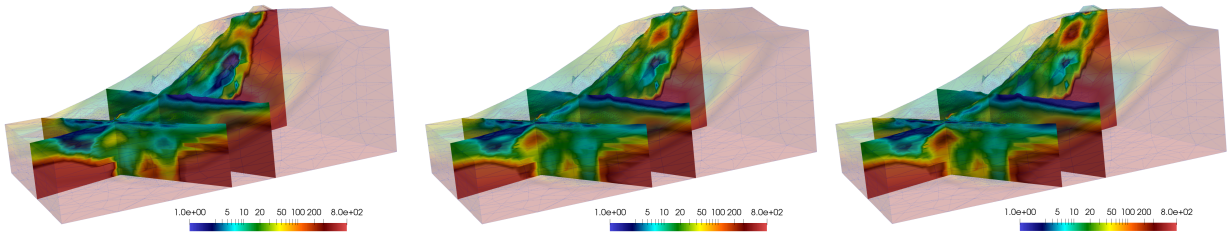


FIGURE 8 – Reconstructed resistivity at iteration $1, 2, 3$.