PyMERRY

Python iMprovement of Electrical Resistivity tomography ReliabilitY

User Manual

Article associated in SEG GEOPHYSICS

Gautier, M., S. Gautier, R. Cattin, 2023, PyMERRY: a python solution for improved interpretation of electrical resistivity tomography images, Geophysics.

Maxime GAUTIER[1], Stéphanie GAUTIER[1], Rodolphe CATTIN[1]

[1]University of Montpellier, Géosciences Montpellier – CNRS, Montpellier, France.

E-mail:

maxime.gautier@umontpellier.fr

stephanie.gautier-raux@umontpellier.fr

rodolphe.cattin@umontpellier.fr

	Version	Date (mm/dd/yyyy)
PyMERRY Software	v1.1	06/08/2023
User Manual	v1.0	06/18/2023

Contents

1 Foreword

Thanks for using PyMERRY: Python iMprovement of Electrical Resistivity tomography ReliabilitY.

Please cite the corresponding article if you are using PyMERRY in your work:

> *Gautier, M., S. Gautier, R. Cattin, 2023, PyMERRY: a python solution for improved interpretation of electrical resistivity tomography images, Geophysics.*

PyMERRY has been developped with PyGIMLi, please cite the corresponding article if you are using PyMERRY or PyGIMLi in your work:

> *Rücker, C., T. Günther, and F. M. Wagner, 2017, PyGIMLi: An open-source library for modelling and inversion in geophysics: Computers & Geosciences, **109**, 106–123, doi: 10.1016/j.cageo.2017.07.011.*

By reading the user manual you will learn how to install the Anaconda suite, configure a virtual environment suitable for PyMERRY, prepare your data and run the code.

The version v1.0 of the PyMERRY code has been tested on the following platforms:

- Windows 10 and 11.
- Linux Ubuntu and Fedora.
- Mac OS Catalina 10.15.16 (Intel).

2 Installation

Before using the PyMERRY module follow the instructions below to install Anaconda Suite and manage a suitable environment for PyMERRY.

PyMERRY is written in Python3. The following Python packages are the main dependencies of PyMERRY.

Library name	Use
PyGIMLi	Geophysical tools library
Pandas	Dataset manager
Numpy	Numerical array tools
Matplotlib	Graphical library
tqdm	Progress bar
spyder	Complete Python IDE (if used)
pyqtwebengine	UI module nessessary for Spyder

2.1 Download and install the Anaconda Suite

Go to the Anaconda website at https://www.anaconda.com/ and download the Anaconda Suite version compatible with your operating system.

Execute the downloaded file and follow the instructions on the screen to install Anaconda Suite.

2.2 Install a virtual environment for PyMERRY

For more information about the installation of PyGIMLi see the PyGIMLi website at: https://www.pygimli.org/installation.html

For more information about virtual environment managing see the Conda website at: https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#activating-an-environment

2.2.1 Windows / Linux

Use the provided .yml files available in the "installation" directory to create a virtual environment suitable for PyMERRY. Follow the instructions:

1. Run the Anaconda Prompt application. (If you don't know how to open Anaconda Prompt, follow the direction 1 to 5 at the "1st option" paragraph explained in the section 4.3.1.)
2. Navigate to the installation directory with the commands:
   - Go to the "dir" directory: `cd dir`
   - Get one directory back: `cd ..`

   Once in the right directory the terminal displays:

   ```
   (base) …/…/main_directory/installation>
   ```

3. Execute the following command to set the virtual environment named "pymerry" with the appropriate .yml file compatible with your operating system (installation could take several minutes):

   ```
   conda env create –f pymerry_env_MyOS.yml
   ```

   The user could be invited to press the [y] key to confirm the installation during the process.

PyMERRY has been tested on virtual environment created on Windows / Linux with the following command line in a terminal:

```
conda create –n pymerry –c gimli –c conda–forge pygimli=1.4.3 pandas pyqtwebengine spyder tqdm
```

2.2.2 Mac OS

To set the "pymerry" virtual environment follow the directions below:

1. Launch your Terminal App.
2. Navigate to the installation directory with the commands:
   - Go to the "dir" directory: `cd dir`
   - Get one directory back: `cd ..`

   Once in the right directory the terminal displays:

   ```
   (base) …/…/main_directory/installation>
   ```

3. Execute the following command to set the virtual environment named "pymerry" with the appropriate .yaml file compatible with your operating system (installation could take several minutes):

   ```
   conda env create –f pymerry_env_macos.yaml
   ```

   The user could be invited to press the [y] key to confirm the installation during the process.

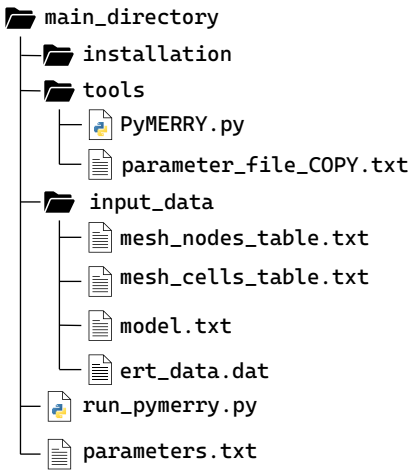
PyMERRY has been tested on virtual environment created on Mac OS (intel) with the following commands lines in a terminal:

```
conda create -n pymerry -c gimli pandas pyqtwebengine spyder
conda activate pymerry
conda install -c gimli pygimli
conda install -c conda-forge tqdm
```

3 Manage directories and files

You need to respect the correct file tree structure to run PyMERRY. The main directory includes:

- The “installation” directory, which contains .yml files for installing the “pymerry” python virtual environment.
- An input data directory named “input_data”. It contains inputs files.
- The “tool” directory with the “PyMERRY.py” script.
- A parameter file named “parameters.txt”.
- The “run_pymerry.py” script for running PyMERRY.



- Do not change the name of directories or files. If a name is modified, the same modification have to be done in the “run_pymerry.py” and “parameters.txt” files at lines where the files are called.
- The “parameter_file_COPY”.txt is available in the “tools” directory. This file has not to be modified and is available for user if the structure of parameter file is accidentally modified.

4 Run PyMERRY

To run PyMERRY, four steps are requested:

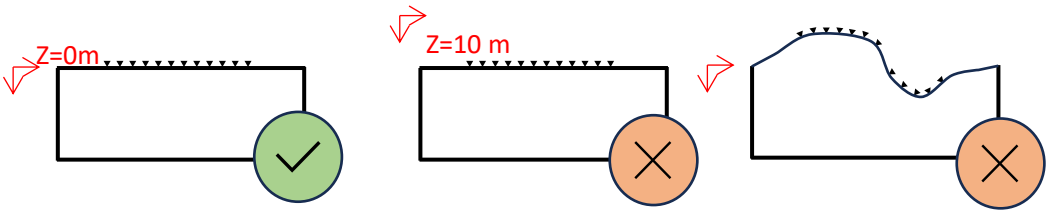
- Manage your main directory properly as described in section 3.
- Prepare your input data files describing the mesh used during the inversion process, the model from the inversion, and the field data. See section 4.1.
- Choice your parameters by updating the “parameters.txt” file. See section 4.2.
- Run the “run_pymerry.py” script. See section 4.3.

4.1 Input files

The “input_data” directory contains the input files needed to launch PyMERRY. Each file is described in detail in the following.

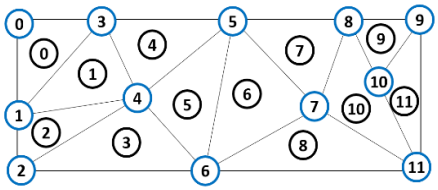
4.1.1 Mesh

- Only **2D** and **rectangular** cross-sections are supported by PyMERRY.
- Only a **triangular** meshing is supported by PyMERRY.
- Topography is not yet supported. **For all nodes on the surface boundary** of the meshed domain and **electrodes, the y-coordinate must be set to 0 meters.** Each electrode position must have a corresponding node.
- Axes for ar 2D mesh in PyMERRY or PyGIMLi are (x, y) and not (x, z).



To complete the procedure, PyMERRY requires the mesh that was used. The mesh is defined by two files: one containing the nodes table and the other describing each mesh cell (the cells table).

Nodes table



The "mesh_nodes_table.txt" file in the input_data_directory contains the 3D coordinates and corresponding numbers of each node in the mesh. It has four columns with no headers and a column delimiter of ";". In 2D, use the 2nd and 3rd columns for x and y, and let zero in the 4th column for z. The number of rows in the file corresponds to the number of nodes, and the columns are organized as follows:

1	2	3	4
Node ID (int)	x coordinate (float)	y coordinate (float)	z coordinate (float)

```
0;0.00;0.00;0.00
1;0.00;-4.00;0.00
2;0.00;-6.00;0.00
3;3.50;0.00;0.00
4;4.55;-3.00;0.00
5;8.00;0.00;0.00
6;7.00;-6.00;0.00
7;12.5;-5.30;0.00
8;13.5;0.00;0.00
9;16.5;0.00;0.00
10;14.0;-3;0.00
11;16.5;-6.00;0.00
```

mesh_nodes_table.tx

```
0;0.00;0.00;0.00
1;0.00;-4.00;0.00
2;0.00;-6.00;0.00
3;3.50;0.00;0.00
4;4.55;-3.00;0.00
5;8.00;0.00;0.00
6;7.00;-6.00;0.00
7;12.5;-5.30;0.00
8;13.5;0.00;0.00
9;16.5;0.00;0.00
10;14.0;-3;0.00
11;16.5;-6.00;0.00
```

mesh_cells_table.tx

Cells table

The cells table contains cells numbers and nodes numbers for each cell. The file is labelled "mesh_cells_table.txt" and is placed in the "input_data_directory". This is a .txt file with 4 columns and no headers. The column delimiter is ";". The number of lines – 1 (Python uses zero indexes array) is the number of cells.

Columns are:

1	2	3	4
Cell ID (int)	Node 1 ID (int)	Node 2 ID (int)	Node 3 ID (in)

```
0;86.552934
1;92.781449
2;95.489338
3;92.049281
4;92.271355
5;102.567012
6;105.500156
7;86.766207
8;102.711571
9;105.450398
10;102.528208
11;100.162781
```

model.txt

4.1.2 Resistivity model

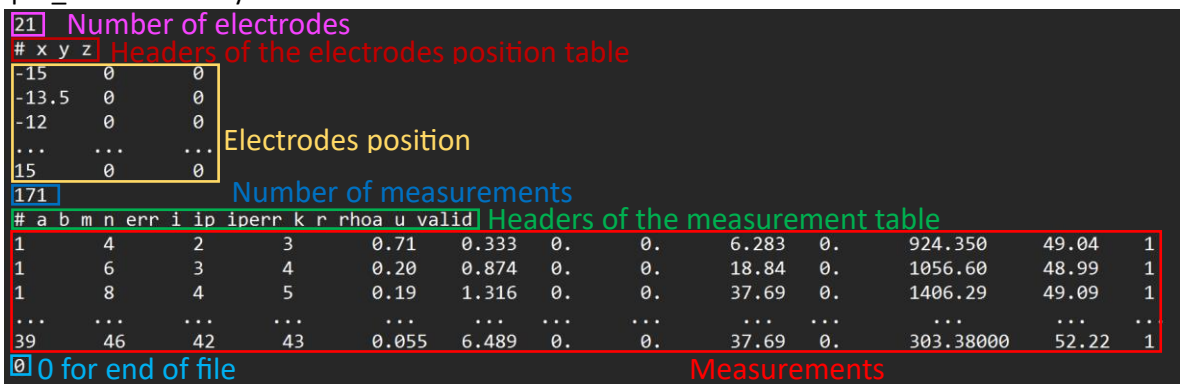
The resistivity model is the resistivity distribution provided by the inversion code on the given mesh. It is set in PyMERRY through a .txt file named “model.txt”. The file contains two columns delimited by the “;” character. The number of lines is the number of cells.

Columns are:

1	2
Cell ID (int)	Resistivity value (ohm-m) (float)

4.1.3 ERT Data

ERT field data are given to PyMERRY with a .dat file named “ert_data.dat” placed into the “input_data” directory. This file is structured as follow:



ert_data.dat

The electrode position table contains the coordinates x y z for each electrode of the array.

The measurement table contains (one line for one data):

- a: index of injection electrode A used in the electrode position table (start at 1).
- b: index of injection electrode B used in the electrode position table (start at 1).
- m: index of potential electrode M used in the electrode position table (start at 1).
- n: index of potential electrode N used in the electrode position table (start at 1).
- err: standard deviation on measurement.
- i: current injected (if used).
- ip: induced polarization (if used).
- iperr: induced polarization error (if used).
- k: geometrical factor of the quadrupole (including the 2 π term).
- r: measured resistivity (if used).
- rhoa: apparent resistivity (ohm-m) measured.
- valid: validity of the data (set 1 for valid and 0 for not valid).

PyMERRY supports Wenner, Wenner-Schlumberger, Dipole-Dipole and gradient configurations.

4.2 Parameters file

PyMERRY needs some additional information. Before running PyMERRY reads a parameters file named “parameters.txt” in the main directory.

Users can change parameter values by modifying some lines of this file before running.

- Do not add, delete or move any line in this file.
- Only lines beginning by the "->" characters could be modified.
- Only modify these lines after the ":" character.
- If the structure of the parameter.txt file is accidentally modified. A copy of the original one is available in the "tool" directory as "parameters_file_COPY".

Parameters are described below:

1	mesh cells table	Path to input files. Delimiter could be "/" or "\".
2	mesh nodes table	
3	resistivity model	
4	data file	
5	directory to save results	Name of the directory for saving results. This directory will be created during the run.
6	resistivity-meter injection accuracy	Resistivity meter accuracy in %. Value could be found in technical notice or estimated by users.
7	resistivity-meter potential accuracy	
8	DOI Apparao (A), or Baker (B)	Choice of the theoretical depth of investigation coefficient. A – Roy and Apparao (1972) B – Baker (1989)
9	DOI Custom* (set a value (example 0.3) or none if unused)	Choose your own depth of investigation coefficient. If the choice of DOI is already set to A or B. This parameter has to be set to "none". If you enter your own value, the precedent parameter is ignored.
10	gamma	Value for plotting error bar ∈]0; 1[(float). See article for details.
11	color bar min value (ohm-m)	Minimal value of the color bar > 0. If "from model" is set, the minimal value will be the minimal value of the model.
12	color bar max value (ohm-m)	Maximal value of the color bar. If "from model" is set, the maximal value will be the maximal value of the model.
13	color bar in log scale	Set "True" for use a logscale color bar. Set "False" for a linear color bar.

4.3 Run

Once you have correctly set the input data and parameter files, you can run PyMERRY using two options: a command line in the terminal or a manual run in an IDE like Anaconda Spyder.

4.3.1 Run with command line in a terminal

Open a terminal window with the "pymerry" virtual environment activated.

1st Option: use Anaconda navigator to open a terminal:

1. Launch your Anaconda Navigator.
2. Go to the environment panel by clicking on the "Environment" button on the left.
3. Activate the "pymerry" environment by clicking on the "pymerry" button and wait for the activation.
4. Once activated click on the play button and on "Open Terminal".
5. A terminal window will appear.

2nd Option: Run your terminal directly:

1. Run your Anaconda Prompt application.
2. A terminal window will appear.
3. Execute the following command to activate the "pymerry environment":

```
conda activate pymerry
```

You are invited to write your commands at the end of the prompt line in the terminal window. Before running PyMERRY you need to navigate to the “main_directory” containing all the files.

Use the following commands to navigate:

- Go to the “dir” directory: `cd dir`
- Get one directory back: `cd ..`
- (Use `cls` or `clear` to clear the terminal window is necessary).

The terminal line is now `(pymerry) …/…/…/main_direcotry>`.

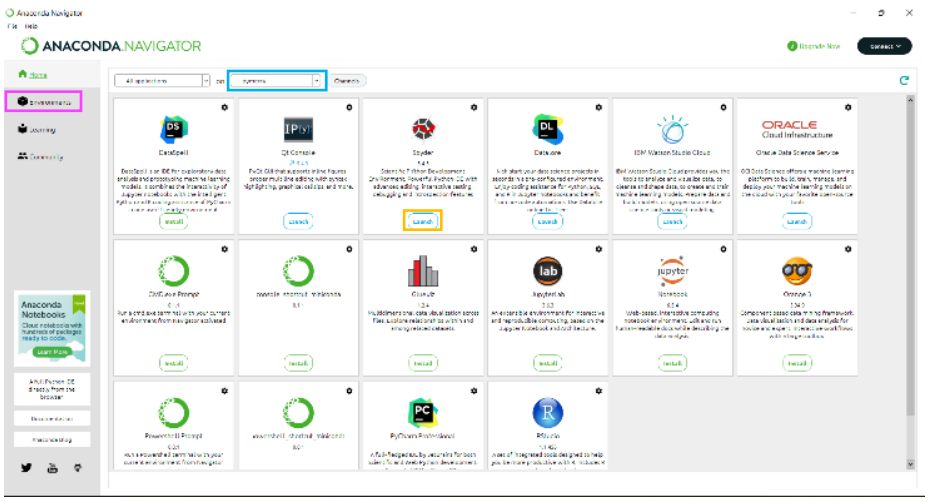
To run PyMERRY enter the following command:

```
python run_pymerry.py
```

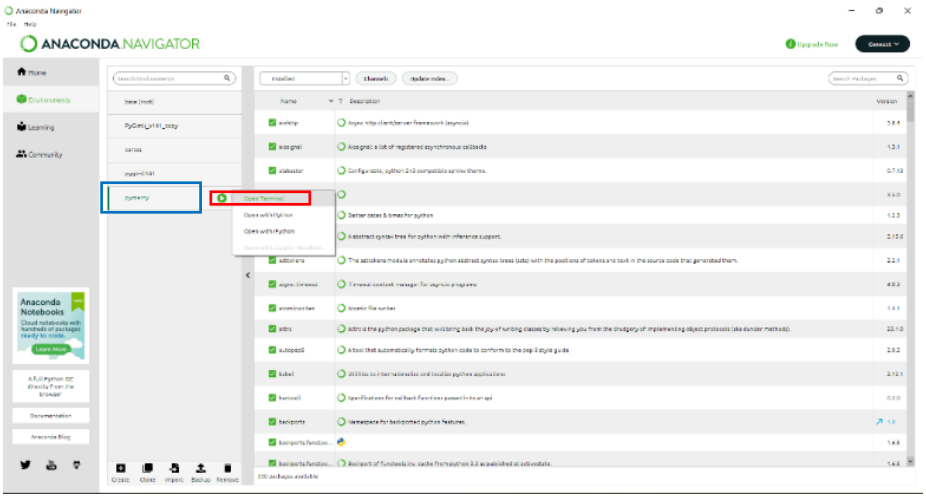
4.3.1 Run in the Anaconda Spyder IDE

PyMERRY could be run from IDE such as Anaconda Spyder. To run PyMERRY with Spyder follow the instructions:

1. Launch your Anaconda Navigator
2. Use the pull-down menu to activate the “pymerry” environment.
3. Run Spyder by clicking the launch button.
4. Open the “run_pymerry.py” file.
5. Click on the run button.



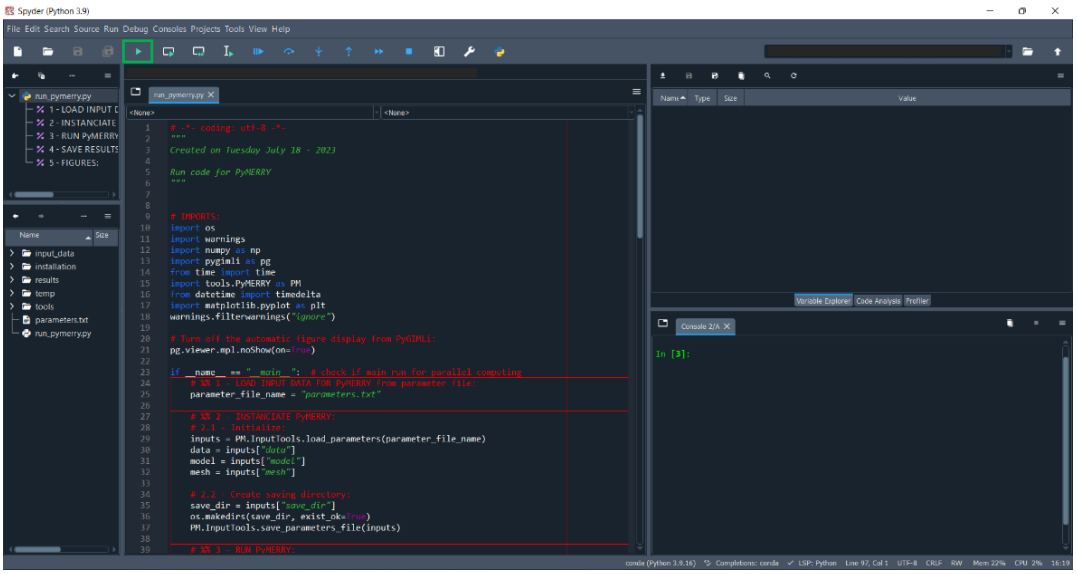
Anaconda navigator main windows.



Anaconda navigator environment window.

```
(pymerry) .../.../main_directory> python run_pymerry.py
```

Command line for run PyMERRY in a terminal.



Run PyMERRY from Spyder IDE.

5 During running

- A temporary directory named “temp” will be created in “main directory” at the beginning of the run. Do not remove this directory while PyMERRY is still running. This temporary directory will be automatically removed at the end of run.
- Results will be saved in a directory named by the user at the corresponding line in the parameters file.
- Computation time is provided at the end of the stages of the process. PyMERRY uses parallel processing and utilizes all the available processors, with the exception of two on your computer, to optimize computation time.

The execution could take several minutes. Some information about the different steps will be printed in the terminal (or in the console if Spyder is used) at each completed step.

A progress bar showing the started tasks (not completed tasks) is available if you run the code with a Linux operating system. It is not available on Windows because of the “spawn” stating method with multiprocessing parallel run with Python.

6 Output files

At the end PyMERRY saves the results in .csv files in the saving directory. The name of this directory has been set in the parameters file.

Two .png files are created to display the coverage map plot and the error bar plot for a quick check.

The user is invited to use the results files to create their own graphics.