An Open-Source Variational Inference Code for Electrical Resistivity Tomography

ERTVI User Guide

Jiahe Yan[1,2] and Andrew Binley[1]

[1] Lancaster Environment Centre, Lancaster University, Lancaster LA1 4YQ, UK

[2] College of Geo-Exploration Science and Technology, Jilin University, Changchun 130026, China

August 2025

1. Introduction

ERTVI is an open-source code used for quantifying resistivity distributions and associated uncertainties in electrical resistivity tomography (ERT) using variational inference (especially SVGD or sSVGD). The code is modified from the open-source package **VIP - Variational Inversion Package** developed by Dr. Xin Zhang and Prof. Andrew Curtis from The University of Edinburgh (https://github.com/xin2zhang/VIP). Therefore, the users who use this code should follow the license of VIP. The ERT simulation code is a modified version of R2 developed by Prof. Andrew Binley. The standard version of R2 can be downloaded at http://www.es.lancs.ac.uk/people/amb/Freeware/R2/R2.htm.

2. Requirements

2.1 VIP

A guide for *VIP* installation can be found in https://github.com/xin2zhang/VIP. A summary is provided below:

pip install Cython

pip install dask

pip install h5py

cd VIP

sh setup.sh install

cd ..

Please use the VIP package included with this code, as we have made minor modifications to fit the characteristics of ERT.

2.2 wine

The ERT simulation code is a modified version of R2 and is provided as an EXE file for Windows. Because the code is to be run on Linux systems, users should install *wine* in the system. There are many tutorials about *wine* installation available on the internet and it may already be pre-installed on your system, so users can install it themselves if necessary, but should first verify that *wine* is installed and functioning correctly before running the code:

wine --version

2.3 g++

Similar to *wine,* a simple procedure for installing and verifying *g++* is as follows:

sudo apt install g++

g++ --version

2.4 Eigen

A C++ library for linear algebra: matrices, vectors, numerical solvers, and related algorithms *Eigen* is required, which should have already been included in the folder. It can be downloaded from https://eigen.tuxfamily.org/index.php?title=Main_Page.

3. Instructions

3.1 R2

You should provide a R2 forward program in a folder with the name "R2". It is the most important part of this code. Detailed information on configuring the R2 input files is available at http://www.es.lancs.ac.uk/people/amb/Freeware/R2/R2.htm. The folder "R2" must include the files *mesh.dat, protocol.dat, R2.in* and *resistivity.dat*. The *meshtype* in *R2.in* is recommended to be 3 or 6 for using triangle or quadrilateral meshes. Please note that you should provide a script *run.sh* in folder "R2", containing the command "*wine R2_J.exe*". Once the R2 files are properly configured, the following scripts can be run for verification:

cd R2

sh run.sh

cd ..

The program *R2_J.exe* is a modified version of R2 that not only generates forward transfer resistances but also the Jacobian matrix of the transfer resistances with respect to the logarithm to base *e* of conductivity, saved in a binary file *R2_forward.bin*. *R2_forward.bin* will be created if the process completes successfully, if not, you should check and correct for mistakes that may exist in your *mesh.dat, protocol.dat, R2.in* and *resistivity.dat* files. Please remember to remove *R2_forward.bin* after testing, since it is not needed once the program has completed.

3.2 config.ini

In this section, you should determine the parameters provided for *VIP*. An example is provided in the folder "examples". Here are some instruments for *config.ini*:

basepath: the base path of your file, e.g. *{path_to_your_folder}/ERTVI*

nparallel: the number of threads you would like to use in your code. This is set in the interest of computational efficiency. It must be the divisor of the number of "particles" *nparticles*. For example, if you would like to use 800 "particles", you can use 2, 4, 5, 8, 10, ... , 200, 400, 800 for *nparallel*. This parameter needs to be set according to your computer's performance. If your computer is extremely powerful (and has sufficient memory and storage), you can set *nparallel* to the maximum, in theory! **Please note this parameter, as it will be used in subsequent steps.**

nobs: the number of observations, which should be the same number as in *protocol.dat* in *R2*

nelement: the number of elements, which should be the same number as in *mesh.dat* in *R2*

obsfilename: the filename of the observations. The file *obsfilename* should contain a vector of transfer resistance with each line corresponds to that in *protocol.dat* in *R2*

a, b: error model parameters describing the standard deviation of measurements, refer to the R2 manual for more details

targetRMS: the target or desired misfit, usually default to 1.0

scalingfactor: the particle updates can be very large if you use a small weight *b* (e.g. 0.002), which may lead to an unstable inversion process. A scaling factor is used to adjust the data weight and reduce the target RMS. For example, with 0.2% noise, setting $b = 0.002$ may cause instability. To address this, a scaling factor of 10.0 is recommended, using a virtual weight $b = 0.02$. In this case, the program reduces the *targetRMS* to 0.1 (but you still should set the *target RMS* = 1.0). If a large data weight *b* is used (e.g., 0.05), the scaling factor can be set to 1.0

method, kernel, diag, optimizer, transform, priortype, nparticles, iter, burn_in, thin, stepsize, final_decay: Please refer to ***VIP User Guide*** for more information

inputpath, outpath: The input and output paths for ERVI. All input and output files are suggested to be put in these two folders

prior, init: The filename of the prior or initial distributions for VI, also refer to **VIP User Guide. Please note that the parameters should be provided in form of the logarithm to base *e* of conductivity. But the final output is the logarithm to base 10 of resistivity.**

seed: The integer seed of the random generator

3.3 Multiple R2 folders

Create multiple R2 forward simulation folders using:

sh mkR2.sh nparallel

where *nparallel* is the value the same as *nparallel* in *config.ini.* If everything goes well, you will find many R2 folders with the name from *R2_0* to *R2_(nparallel-1)* in the working directory. Each folder is independent of the others, so they can run completely in parallel. These folders will be used to calculate forward transfer resistances and Jacobian matrices for VI. You should not delete the original R2 folder because the program will also read the first two columns of *resistivity.dat* from it (and you may also need to use it in further analysis). If you would like to modify the files in the R2 folders or use another *nparallel*, you should remove these folders first and then create new folders:

sh rmR2.sh nparallel(old)

sh mkR2.sh nparallel(new)

3.4 multiR2.out

Open the "*main.cpp*" in the folder, find "*int n_threads = XX;*". You should modify the value to be the same as *nparallel* in *config.ini.* Then, compile the C++ code *main.cpp* using

make

You will get an executable file *MultiR2.out.* The file is used to control multiple folders from *R2_0* to *R2_(n-1)* and runs R2 forward simulations in parallel. Each folder will provide an output file *R2_forward.bin*. Then, the executable file reads the *.bin* files and outputs a binary file *lossgrad.bin* in the working directory, which will be used by Python codes to continue VI for ERT.

3.5 run.sh

Modify run.sh as follows:

PYTHONPATH={path_to_your_folder}/ERTVI python ERTVI2d.py -r 0

The program does not support the "continue" function, but if the user would like to use this, please modify the codes referred to in the VIP examples.

Now, run the program using

sh run.sh

If everything goes well, the program will run smoothly and output files *samples.txt* (contains all resistivity distributions at each element in the mesh, with a size of *nparticles * nelements*) and *samplemeanstd.txt* in the folder *results*. Then, what you need to do is to analyze these samples as you so wish!

Be careful when running the code on the solid-state drive, because the program will involve massive Jacobian matrix reading and writing processes, which may cause damage to your hard disks! The authors are not responsible for any damage and errors caused by the code and its use. This problem will be addressed in the next version of the code which will involve much less I/O operations. For more information and future updates, please visit https://github.com/YanJiahe1998/ERTVI.

4. Examples

Synthetic and field experiments are included in the folder *"examples"*, which can also be used as the template for the code. Note that the field data in Chenqi watershed originated from a project supported by the UK Natural Environment Council (NERC) Grant NE/N007409/1 awarded to Lancaster University.

5. References

If the code is helpful for your research, please cite the following references:

Yan, J., Zeng, Z., Tso, C. H. M., Cheng, Q., & Binley, A. (2025). Uncertainty in Hydrogeophysics: Electrical Resistivity Tomography with Variational Inferences. *in preparation*

Zhang, X., & Curtis, A. (2024a). VIP - Variational Inversion Package with example implementations of Bayesian tomographic imaging. *Seismica*, 3(1). https://doi.org/10.26443/seismica.v3i1.1143

Zhang, X., & Curtis, A. (2020). Seismic Tomography Using Variational Inference Methods. *Journal of Geophysical Research-Solid Earth*, 125(4). https://doi.org/10.1029/2019JB018589