bmt1dinv
Programs for 1D Bayesian inversion of magnetotelluric data considering model discrepancy

Hoël Seillé & Gerhard Visser [*][†]

January 28, 2022

1 Introduction and overview of the programs

The present document describes the workflow to perform 1D Bayesian inversions of magnetotelluric (MT) data considering dimensionality discrepancies. The workflow is described using synthetic MT soundings as example. Refer to the paper "Bayesian inversion of magnetotelluric data considering dimensionality discrepancies" (Seillé and Visser, 2020) for more details about the algorithm and the workflow. The synthetic soundings presented in the paper are the ones used in this manual.

The workflow is divided into three steps (see Fig. 1):

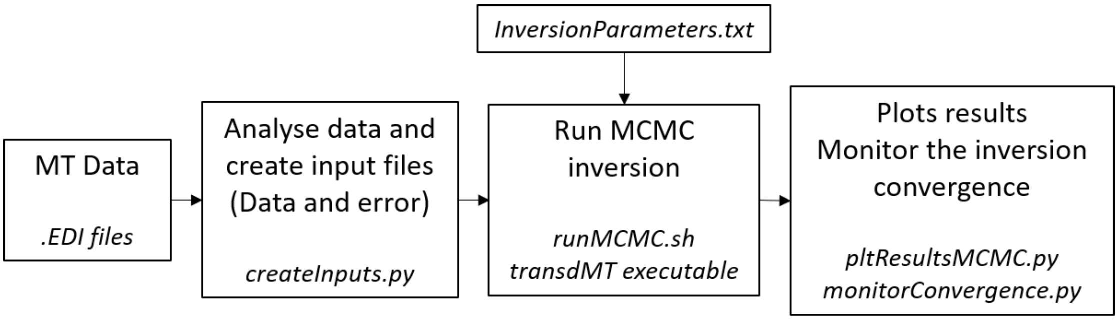


Figure 1: Workflow to run the bmt1dinv programs.

[*] CSIRO Deep Earth Imaging FSP, Perth, Australia

[†] CSIRO Mineral Resources, Perth, Australia

- **Create Input Files**: The MT data is analysed and prepared for the inversion using the **createInput.py** python script. Using input MT data in the form of *.edi* files and an existing "Dimensionality Discrepancy model", this program computes the phase tensor parameters in order to estimate the "site-specific likelihood" functions reflective of uncertainty caused by the presence of 2D and 3D effects. This uncertainty is summed up to the processing noise variance/covariance matrix and stored alongside the data (determinant of the full impedance tensor) into in a *.csv* file for use into the probabilistic algorithm.

- **MCMC Inversion**: The inversion is ran using the **runMCMC.sh** shell script that calls the *transdMT* inversion executable that performs the trans-dimensional Markov chain Monte Carlo probabilistic inversion. The *.csv* files created by the previous program is used as the data input file by the executable. Parameters of the inversion are specified in the **inversionParameters.txt** file. The output of the inversion are binary files which contains the ensembles of models and responses and the statistics of the inversion. Optionally the complete statistics of the inversion (including the pre burn-in phase) can be recorded.

- **Plot Results of the Inversion**: The outputs of the inversion are read and analysed using two python scripts. The **plotResultsMCMC.py** script reads and plots the posterior ensembles and the statistics of the inversion. The **monitorConvergence.py** script plots the complete statistics of the inversion (including the pre burn-in phase) if this option was selected for the inversion. It allows to monitor and assess the convergence of the inversion in terms of likelihood.

2 Dependencies

This section describes the python libraries required by the python scripts to run:

- numpy
- pandas
- scipy
- matplotlib
- sys
- os
- re
- numba

The present suite of Python scripts can be run on any operating system with python 3.6 installed. The **transdMT** executable is only working in a Linux environment at the moment.

3 Directory structure

The program has the following directory structure:

```
bmt1dinv/
   docs/          --> manual
   projects/      --> project directories (inputs/outputs)
   scripts/       --> main scripts
   src/           --> source scripts and files
```

Users only need to run the scripts located in the *bmt1dinv/scripts/* folder. All projects have the same directory structure (see Fig. 2). The *empty project* directory can be copy-pasted and used as a blank directory for a new project.

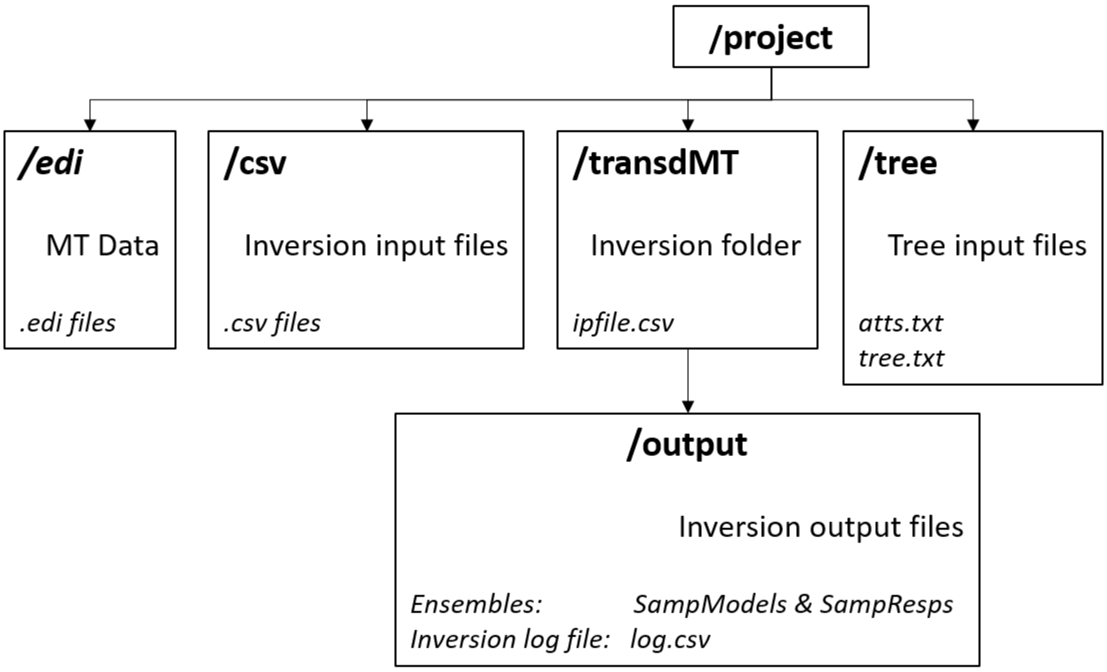


Figure 2: Project directory structure.

4 Create Input Files

The **createInput.py** script uses *.edi* files as input files and computes the determinant of the full impedance tensor and the uncertainty to consider within the inversion (model discrepancy plus processing uncertainty). Most of the processed MT data is found in the form of EDI files (Wight, 1987). It contains data in form of the full complex impedance

Tensor Z at each frequency, and the variance associated to each element of the tensor. The variance is real and common to both real and imaginary parts of each element of Z, assuming a circularly-symmetric complex normal distribution. The script that reads the *.edi* files accepts a standard format, the same as the one used by the WinGLink software when exporting the EDI files. If the full noise covariance matrix is available, it would be possible to use it, but this option has not been implemented yet.

The (edited) *.edi* files have to be manually placed into the */edi/* directory of a specific project. Files associated to the model discrepancy uncertainty estimation (*tree.txt* and *atts.txt*) are located in the *tree/* directory. These files were created for a specific cover thickness estimation application, (see Seillé and Visser, 2020). Refer to section 8 for details on these files and how to create your own.

Before creating the inversion files a careful edition of the data is sought. Obvious outliers should be removed for two reasons. 1) It can compromise the inversion and 2) outliers in the Phase Tensor parameters might be assimilated by the Dimensionality Discrepancy Estimation (DDE) algorithm as data having a high-order dimensionality (i.e. be considered as 2-D or 3-D data). If encountered at high frequencies the rest of the frequency range will also be considered as having a high-order dimensionality. It is why it is crucial to remove outliers or data suspected to be noisy at this stage of the workflow. In the presence of this type of outliers the input data should be reedited and the process of creating the input file repeated. The inclusion of a median filter to smooth the phase tensor parameters allows to remove these outliers. The length of this filter can be modified using the *medfiltsp* option.

The script's options can be defined in the header of the script itself. The project name has to be specified in the script. The options are:

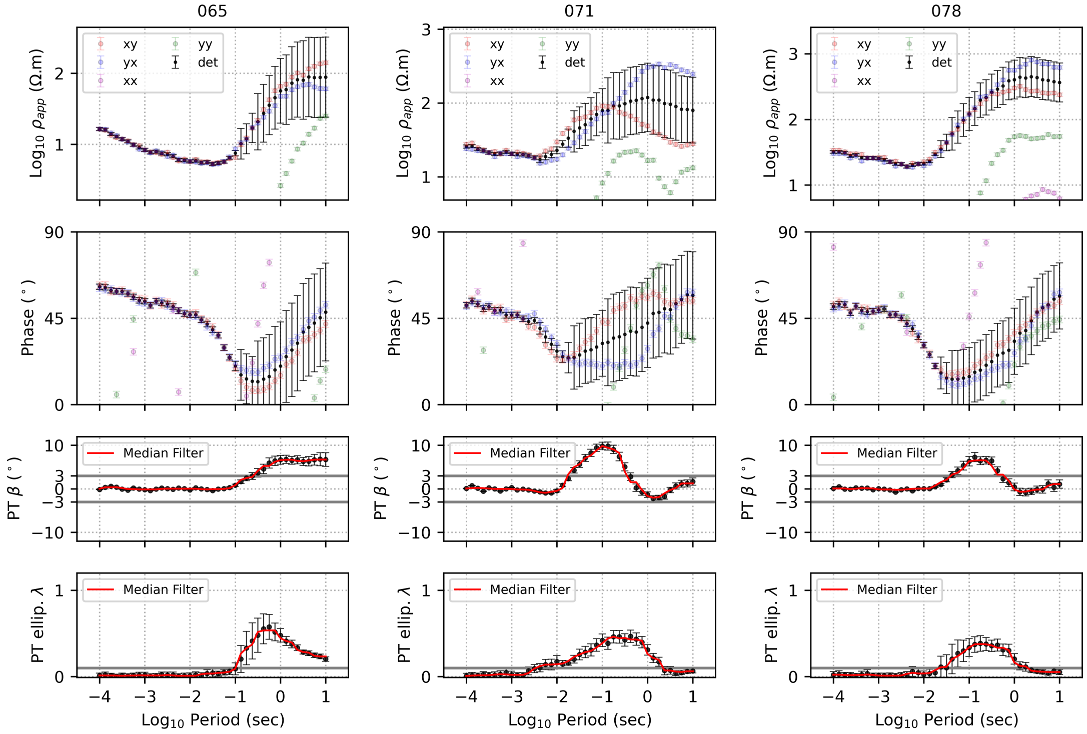
- **Error Floor *EF*:** when EF = -1, the error that will be assigned to the data will consider the model discrepancy using the PT parameters. If EF ¿ 0, the error value indicated is used as an error floor across the complete frequency range. If using an error floor, the resulting csv file will have appended the string 'EF'.

- **Phase Tensor parameters median filter length *medfiltsp*:** The value defines the lenght of the window that will be used to perform the median filter for both PT parameters. It is in log scale, along the frequency axis. For example, *medfiltsp = 0.5* filters along half a frequency decade.

- **Plot the input data:** Option to save the data plots.

- **Save .csv files:** Option to save the *.csv* files.

The input data images are saved in the same folder as the textit.edi files. It serves at a first QC to ensure that no outliers are present in the Phase tensor (PT) parameters β and λ.



(a) MT065

(b) MT071

(c) MT078

Figure 3: Plots created by the DDE.py script

5 MCMC Inversion

The **transdMT5** executable takes as input the textit.csv file and outputs an ensemble of models that fits the data within its estimated uncertainty.

Usage of the executable:

```
    main gamma ipfile runc ipfile outpath [-s seed] [-p samples]
    [-h min max mean std] [-u min max] [-t startT]
```

For example, from *projects/example/*:

```
../../src/transdMT7 2 ./transdMT/ipfile.csv 50000
  ./csv/065.csv ./transdMT/outfolder -s 100 -u -2 6
```

This command will run 1 Markov chain with 50.000 iterations for site 065, using an uniform resistivity prior between -2 and 6 log10 ohm.m, and a depth dependent prior as specified in ipfile.csv. The results (ensemble of responses and models) will be found in the outfolder. Running a single chain with few iterations can provide a quick way to check if the inversion/data is working properly.

When running the inversion using multiple chains (and multiple MT sites) the **runMCMC.sh** shell script located in the */scripts/* folder can be used, using this command in a terminal:

```
     sh runMCMC.sh project_name
```

The script will run in batch the inversion for all the input *.csv* files located in the *project/csv/* folder. The *inversionParameters.txt* file allows to specify the parameters to use within the inversion using this format:

```
number_of_chains_running_in_parallel=5
total_number_of_chains=20
iterations_per_chain=500000
ipfile=ipfile.csv
gamma=2
samples_per_chain=100
prior='-u'
min_resisitivity_log10=-2
max_resisitivity_log10=6
save_full_MCMC_convergeance=true
```

- **gamma:** 1 for random layer widths, 2 for more equal layer widths
- **prior:** use '-u' for uniform log10(res) prior; use '-h' for truncated gaussian log10(res) prior.

- **min/max resisitivity log10:** for uniform prior only
- **mean/std resisitivity log10:** for gaussian prior only

This example is run using an uniform prior on the resistivity, 20 chains are run, 5 at a time in parallel, with 500.000 iterations in each chain.

The script merges merges the results from all the chains into single files after the inversion. The ensemble of models *sampModels* and responses *sampResps* for each site, along with the input csv files will be generated in the *project/trandMT/outfolder*. If the *save full MCMC convergeance* option is selected, a *log.csv* file is also saved in the same folder.

5.1 Depth dependent prior file

The depth dependent prior file defines the prior information on the expected number of layers within certain depth intervals. Using this prior allows to encourage complexity within certain intervals, but also discourage the apparition of unnecessary layers in intervals that are not constrained by the data. It is defined as follow:

```
division,rate
5,1
3000,5.5
10000,2
100000,2
```

The *division* column defines the depth of the bottom of an interval. The *rate* column defines the number of layers within that interval. If a uniform prior wants to be used, the file has to contain only one line, with the depth of the bottom of the model and the prior number of layers for the entire model.

6 Plot Results of the Inversion

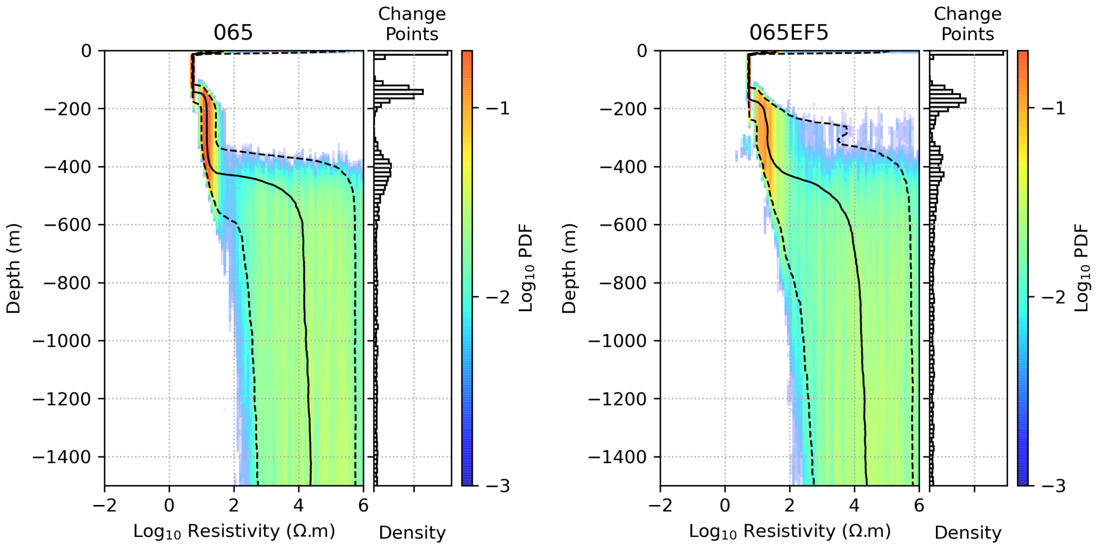
The script **plotResultsMCMC.py** plots the posterior distributions for the models and the responses. The statistics and evolution of the RMS, the likelihood and the number of layers after the burn-in phase can be plotted.

The project name has to be specified in the script. The options are:

- **Plot Models:** Plots the posterior model distribution as a 2D histogram.
- **Plot Responses:** Plots the posterior response distribution as a 2D histogram. If the *plot fast responses* option is selected, 100 models from the ensemble will be plotted. This option makes the plot much faster.
- **Plot Inversion Statistics:** Plots the evolution and distribution of the RMS / normalised log likelihood / number of layers during the post burn-in phase.

- **plot niblettBostick:** Plots the approximate Niblett-Bostick depth-transform in the response plots.

The images created by the script will be saved in the *project/transdMT/outfolder* folder. Examples of model posteriors, response posteriors and inversion statistics are shown in in Fig. 4, Fig. 5 and Fig. 6 respectively.



(a) Site-Specific Likelihoof Fcts

(b) Error Floor 5%

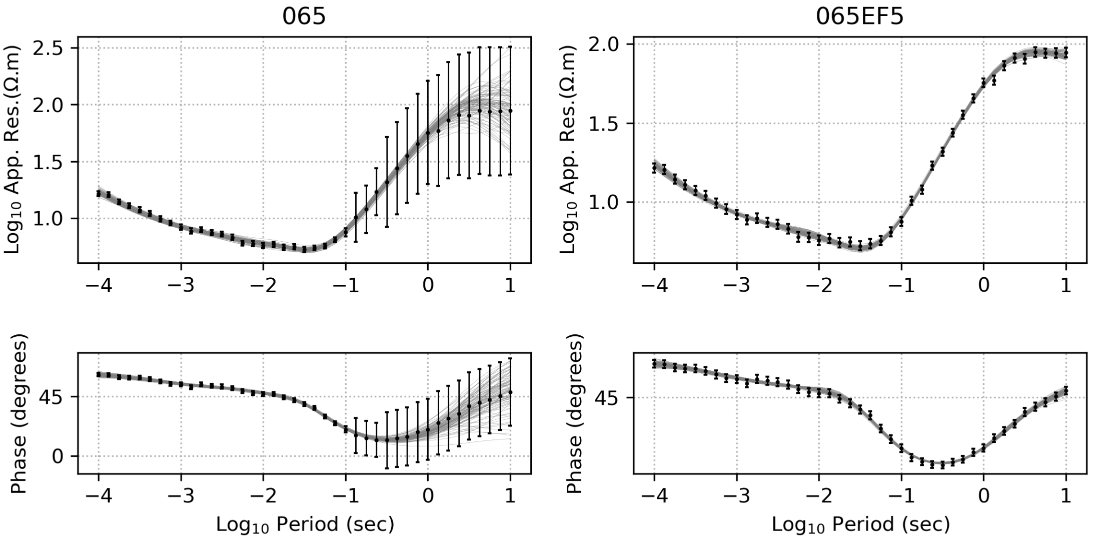
Figure 4: Plot created by the *plotResultsMCMC.py* script: Models posteriors.

7 Monitor the inversion convergence

An important criteria to assess if the inversion has reached convergence is to observe a stationary state of all the chains in the post burn-in phase. Also, the variability within each chain and between the chains should be similar. Quantitative measures of convergence has not been implemented, the assessment of the convergence is done graphically, plotting the evolution of all chains for all the iterations.

The **monitorConvergence.py** script plots the evolution of the normalised log likelihood and the number of interfaces for all the iterations for all the chains. The option to save the convergence results has to be chosen during the inversion (see Section 5). Fig. 7 shows the results for MT site 065.

When using real data, 10^6 iterations and 60 chains was generally satisfactory.



(a) Site-Specific Likelihood Functions

(b) Error Floor 5%

Figure 5: Plots created by the *plotResultsMCMC.py* script: Responses posteriors.

8 Custom Tree

This section describes the attributes file format and the tree file format, and how a custom tree can be constructed.

8.1 Attribute files

The attributes txt.file is constructed as follow: 3 columns, comma separated:

```
    condition number, condition name, condition
```

Example:

```
1,’elip1’,elip > 0.1
2,’elip2’,elip > 0.2
3,’elip3’,elip > 0.3
...
7,’beta1’,abs(beta) > 1
8,’beta2’,abs(beta) > 2
...
```

The condition numbers have to match the ones in the tree file.

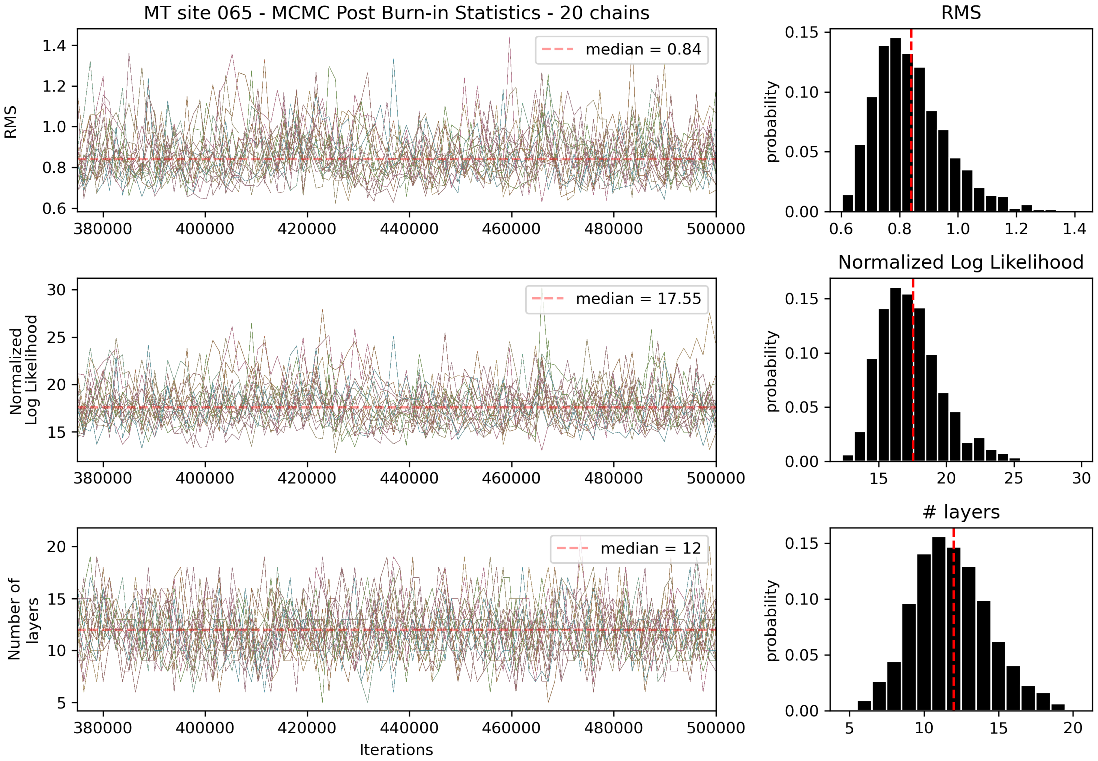


Figure 6: Plot created by the *plotResultsMCMC.py* script: MCMC inversion statistics for MT site 065.

8.2 Tree file

The tree file is constructed as follow:
1st line: variogram
2nd line: variogram
from 3rd line to the end: 3 columns, white space separated

```
tree root number, tree node characteristic (split or leaf),
       std dev (if leaf) or condition number (if split)
```

Example:

```
1.8999999999999997
0.7983039513625659
0 Split 2
1 Split 8
2 Split 5
3 Split 9
```

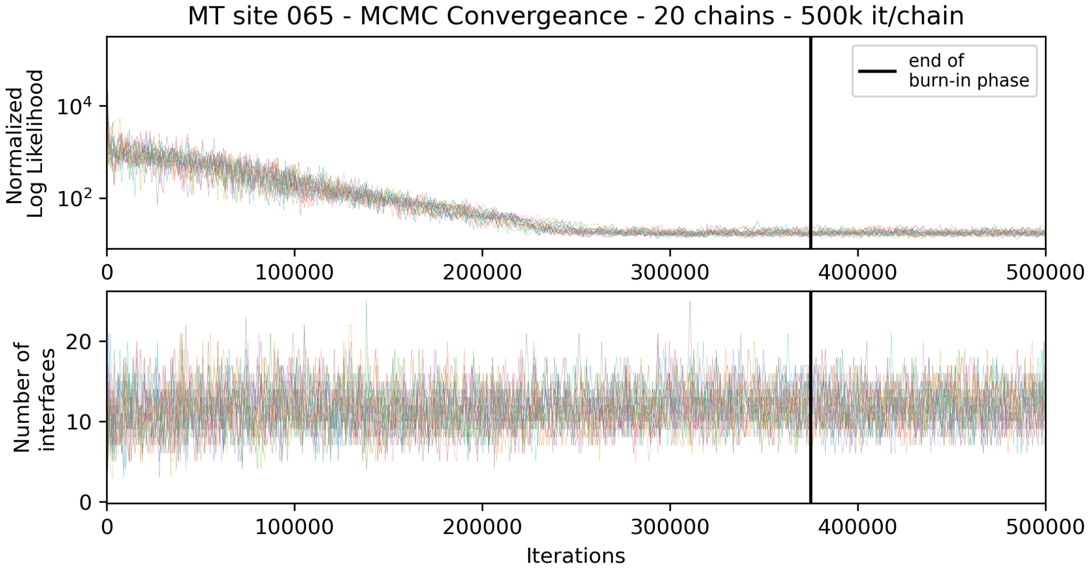


Figure 7: MCMC convergence plot for MT site 065.

```
    4 Leaf 0.575751
    4 Leaf 0.537254
    ...
```

The standard deviations (in log scale) for each data point is found at the leafs of the decision tree given the conditions established on the dimensionality attributes.

The tree goes first to the "True" binary condition result.

The variograms parameters define how the errors are correlated across adjacent frequencies.

8.3 Construction of a simple tree

Here is how would look like a simplistic tree defined using 2 attributes:

Attributes file

```
1,'elip1',elip > 0.1
2,'beta1',abs(beta) > 3
```

Tree file

```
2
1
0 Split 1
```

```
1 Split 2      --> if elip > 0.1
2 Leaf 0.6     --> if abs(beta) > 3 (3D)
2 Leaf 0.2     --> if abs(beta) < 3 (2D)
1 Split 2      --> if elip < 0.1
2 Leaf 0.6     --> if abs(beta) > 3 (3D)
2 Leaf 0.03    --> if abs(beta) < 3 (1D)
```

Note that the standard deviation defined here and the variogram parameters set here are arbitrary.

9 Notes

While the algorithm is designed to include either a DDM or to use error floors, it will be possible in a future version to include both simultaneously: when the Earth is relatively assimilable as uni-dimensional, the errors (processing + DDM) might still remain very low. This could be incompatible with the user's understanding of other sources of uncertainty (underestimated processing errors, physical assumptions). Then, this could be accounted for incorporating them within an error floor, used simultaneously with the DDM.

References

Seillé, H. and Visser, G. (2020). Bayesian inversion of magnetotelluric data considering dimensionality discrepancies. *Geophysical Journal International*, 223(3):1565–1583.

Wight, D. E. (1987). MT/EMAP Data Interchange Standard. Technical report, SEG.