# TransdEM:

An open-source package for transdimensional Bayesian inversion of electromagnetic data in 1D layered media.

Version 1.0

January, 2022

# Contents

## 1 Introduction

This document describes how to use the Julia[1] software package **TransdEM** perform transdimensional MCMC inversion for different EM data in 1D layered media. Here's a list of features currently supported in **TransdEM**:

- Forward modeling of MT/CSEM/TEM responses in 1D layered media
- Transdimensional MCMC inversion of 1D MT/CSEM/TEM data
- Statistical analysis of posterior probability distribution about model parameters (e.g., mean, median, and mode)
- Transdimensional MCMC samplings with parallel tempering

[1]http://julialang.org/

## 2 Installation

### 2.1 Installation of Julia

The package is compatible with Julia v0.7 or later versions (The LTS version v1.0.5 is recommended).

#### 2.1.1 Windows systems

Go to the Julia download page to download the Windows command line version (.exe) and install it.

#### 2.1.2 Linux systems

Although Julia is a cross-platform language, we strongly recommend to run **TransdEM** under Linux rather than Windows. This is because some of the third-party packages utilized by **TransdEM** such as **Dipole1D** are more straightforward to compile under Linux. There are three ways to install Julia on Linux:

- **Using precompiled binaries (recommended)**. Go to the Julia download page to download the generic Linux binaries (.tar.gz file). Then make sure that the Julia executable is visible for your system. To do this, first extract the .tar.gz file to a folder on your computer. Then you can either add Julias bin folder to your system PATH environment variable, or create a symbolic link to julia inside a folder which is on your system PATH, for example, by using the following command:

  ```
  sudo ln -s <where you extracted the julia archive>/bin/julia /usr/local/bin/julia
  ```

- **Compiling from source**. Assume that Git has been installed already, then we can grab the Julia sources from GitHub by using the following command:

  ```
  git clone git://github.com/JuliaLang/julia.git
  ```

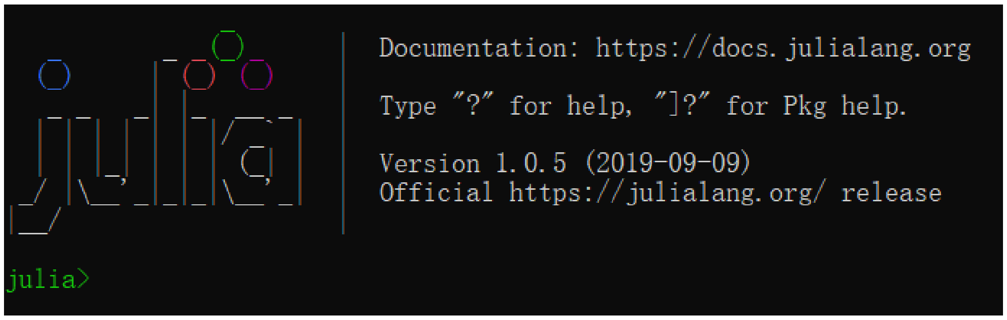
  This will download the Julia source code into a julia directory in the current folder. The Julia building process needs the GNU compilation tools g++, gfortran, and m4, so make sure that you have installed them. Now go to the Julia folder and start the compilation process as follows:

```
cd julia
make
```

- **Using PPA for Ubuntu Linux**. Particularly, for Ubuntu systems (Version 12.04 or later), there is a Personal Package Archive (PPA) for Julia that makes the installation painless. All you need to do to get the stable version is to issue the following commands in a terminal session:

```
sudo add-apt-repository ppa:staticfloat/juliareleases
sudo add-apt-repository ppa:staticfloat/julia-deps
sudo apt-get update
sudo apt-get install julia
```

After a successful installation, Julia can be started by double-clicking the Julia executable (on Windows) or typing "julia" from the command line (on Linux). Following is an illustration of Julia's command line environment (the so-called REPL):



```
Documentation: https://docs.julialang.org

Type "?" for help, "]?" for Pkg help.

Version 1.0.5 (2019-09-09)
Official https://julialang.org/ release

julia>
```

Figure 1: An illustration of Julia REPL.

## 2.2 Installation of the TransdEM package

### 2.2.1 Setting up the package environment

**TransdEM** depends on several external packages (the so-called dependencies) which are not shipped with the package .zip file. These dependencies can be automatically resolved by activating and instantiating the package environment through Julia's package manager (Pkg). Suppose that the TransdEM package is placed at `home/username/code` on Linux or at `D:\code` on Windows, you can type the following command from the Julia REPL to go to the package directory:

```
julia> cd("/home/username/code/TransdEM")
```

on Linux, or

```
julia> cd("D:\\code\\TransdEM")
```

on Windows. Then press ] from the Julia REPL you will enter the Pkg REPL which looks like

```
(v1.0) pkg>
```

, indicating that you are currently in the environment named "v1.0", Julia 1.0's default environment. To switch to the package environment, just `activate` the current directory:

```
(v1.0) pkg> activate .
```

you will get:

```
(TransdEM) pkg>
```

indicating that you are in the environment "TransdEM". The environment will not be well-configured until you `instantiate` it:

```
(TransdEM) pkg> instantiate
```

. By doing so the dependencies listed in `Project.toml` and `Manifest.toml` can be automatically downloaded and installed.

To get back to Julia REPL from Pkg REPL, press `backspace` or `^C`.

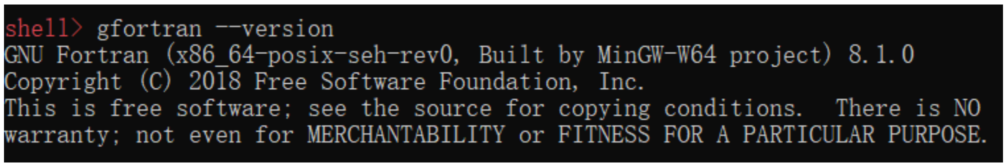
### 2.2.2 Building the em1dmod library

**TransdEM** contains several Fortran codes that are used to compute 1D MT/CSEM/TEM responses, and they must be compiled to generate a shared library (.so, on Linux) or dynamic link library (.dll, on Windows) called `em1dmod` prior to running **TransdEM**. Suppose that you have already installed the GNU fortran compiler (gfortran) in your computer system, all you need to do is going to the subdirectory of the `TBFwdSolver` module and ”including” the script `build.jl`, the Julia REPL commands of which are like:

```
julia> cd("/home/username/code/TransdEM/src/TBFwdSolver/deps")
julia> include("build.jl")
```

. If no error is reported during the building process, the library file should be generated and placed at the directory like `D:\code\TransdEM\src\TBFwdSolver\deps\lib`.

For installing gfortran on Windows systems we recommend to install MinGW, particularly Mingw-w64 for 64 bits systems. The windows installer (.exe) can be downloaded from https://sourceforge.net/projects/mingw-w64/. After installing, you need to edit the `PATH` variable. You can access the System Control Center by pressing **Windows Key + Pause**. In the System window, click **Advanced System Settings** → **Advanced (tab)** → **Environment Variables**. For Windows 10, a quick access is to enter ”Edit the system environment variables” in the Start Search of Windows and click the button ”Environment Variables”. Change the `PATH` variable (double-click on it or Select and **Edit**), and add the path where your MinGW-w64 has been installed to e.g., `C:\mingw\mingw64\bin`. This folder should contain a number of .exe-files that you can see in your explorer. To check that your Mingw-w64-gfortran is correctly installed and available, enter the shell mode by pressing ”;” from the Julia REPL and type ”`gfortran --version`”, the expected output looks like



```
shell> gfortran --version
GNU Fortran (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Figure 2: The expected output of the command ”gfortran –version”.

## 3 Running the code

### 3.1 Running single MCMC sampling

- **First**, you need to let the **TransdEM** package to be "loaded" by the current Julia environment. This is done by adding the parent directory of the package directory to `LOAD_PATH`, a global environment variable of Julia. For example, the TransdEM package is placed at `home/username/code` on Linux or at `D:\\code` on Windows, then type the following command from the Julia REPL:

  ```
  julia> push!(LOAD_PATH,"/home/username/code")
  ```

  on Linux, or

  ```
  julia> push!(LOAD_PATH,"D:\\code")
  ```

  on Windows.

- **Then**, Then go to the package directory for example: `cd /home/username/code/TransdEM` and activate the **TransdEM** environment (please refer to the section **"Setting up the package environment"**).

- **Finally**, go to the directory where the running script loaded, and run the script by typing the following command (for example) from the Julia REPL:

  ```
  julia> include("runMCMCScript.jl")
  ```

### 3.2 Running parallel MCMC sampling

To perform parallel MCMC sampling, call the parallel MCMC sampling function **parallelMCMC-sampling** instead of the single MCMC sampling **runMCMC** (please refer to the `paraMCMCScript.jl` scripts within the `examples` directory).

- **First**, you need to launch multiple worker processes by either starting Julia like:

  ```
  shell> julia -p 12
  ```

or adding processes within Julia (recommended) like:

```
julia> addprocs(12)
```

- **Then** you need to let the **TransdEM** package to be ”loaded” on all processes by following command from the Julia REPL:

  ```
  julia> @everywhere push!(LOAD_PATH,"/home/username/code")
  ```

- **Finally**, go to the directory where the running script loaded, and run the script by typing the following command (for example) from the Julia REPL:

  ```
  julia> include("paraMCMCScript.jl")
  ```

## 3.3 Running MCMC sampling with parallel tempering

In order to accelerate convergence of the Markov chains, a powerful technique known as parallel tempering can be used. To perform MCMC sampling with parallel tempering, call the parallel tempered MCMC sampling function **runTemperedMCMC** instead of the parallel MCMC sampling function **parallelMCMCsampling** (please refer to the `runPTMCMCScript.jl` scripts within the `examples` directory).

- **First**, you need to launch multiple worker processes by either starting Julia like:

  ```
  shell> julia -p 6
  ```

  or adding processes within Julia (recommended) like:

  ```
  julia> addprocs(6)
  ```

  Note that the number of processes invoked should be equal or larger than the number of parallel tempered Markov chains.

- **Then** you need to let the **TransdEM** package to be ”loaded” on all processes by following command from the Julia REPL:

```
julia> @everywhere push!(LOAD_PATH,"/home/username/code")
```

- **Finally**, go to the directory where the running script loaded, and run the script by typing the following command (for example) from the Julia REPL:

```
julia> include("runPTMCMCScript.jl")
```

## 3.4 Writing a running script

For each numerical example contained in the directory `examples`, one or more running script-s named `runMCMCScript.jl`, `paraMCMCScript.jl` or `runPTMCMCScript.jl` have been provided. These scripts are well documented. A user can modify them to get his/her own.

## 4 Files required by the package

There are two files required to run the **TransdEM** package: the data file and startup file. Each file is described in detail in the following sections.

### 4.1 Data File

The data file describes survey information. For current version, the file supported contains data from MT, CSEM and TEM surveys. Because these methods have different survey configuration and data types, their data file formats are designed to be different. Note that these data file formats are specific to the TransdEM package.

#### 4.1.1 CSEM data file format

The CSEM data file contains information about CSEM surveys, including transmitter and receiver locations, frequencies, data types, and data values and associated standard errors. Following is an example of CSEM data file:

```
# Format:          CSEMData_1.0
# Description:     inline and broadside, two frequencies.
SeaLayer:         1000.0  3.2 (optional, required for marine CSEM case)
Dipole Length:       1.0      (optional, default value < 10)
Phase Convention:    lead     (optional, default value is "lead")
Source Location (m):        1
#        X            Y         Z     Azimuth      Dip
0.0         0.0       950.0    0.0         0.0
Receiver Location (m):      20
#          X          Y            Z
500.0        0.0      1000.0
...
10000.0        0.0      1000.0
Frequencies (Hz):       2
2.5000e-01
```

```
5.0000e-01
DataType:     2
ampEx
phsEy
Data Block:    80
# FreqNo. TxNo.   RxNo.  DTypeNo.  Value        Error
1      1      1      1      4.638014e-10    2.337142e-11
1      1      1      2     -2.717600e+01    2.864789e+00
1      1      2      1      3.641894e-11    1.689530e-12
1      1      2      2     -6.338186e+01    2.864789e+00
...
2      1     20      1      1.315654e-14    6.638414e-16
2      1     20      2     -1.741810e+02    2.864789e+00
```

The following is an brief explanation of the keywords of the CSEM data file.

- **Dipole Length**: followed by a real value, representing the length (in meters) of the electric dipole source. If the length is bigger than 10, then the transmitter will be treated as a finite length line source rather than a point dipole. This is optional, and the default value is less than 10, which means by default the source is regarded as a point dipole.

- **SeaLayer**: followed by two real values, representing the thickness (in meters) and conductivity (in S/m) of the seawater layer, respectively. This is optional, and required for marine CSEM case.

- **Phase Convention**: followed by a string. Phase "`lead`" corresponds to the $e^{iwt}$ time dependence, while phase "`lag`" corresponds to the $e^{-iwt}$ time dependence. This is optional, and the default value is "`lead`".

- **Source Location**: followed by an integer, which is the number of transmitters. The $x$, $y$, $z$ locations (in meters), the rotation angle (degrees clockwise from $x$) and the dip angle (degrees positive down) of transmitters are listed below it.

- **Receiver Location**: followed by an integer, which is the number of observation sites. The

$x$, $y$, $z$ locations (in meters) of sites are listed below it.

- **Frequencies**: followed by the number of frequencies, and the frequency values (in Hz) are listed below it.
- **DataType**: specify the data types used for inversion. For CSEM the allowed **DataType**s are the amplitude and phase of the electromagnetic fields.
- **Data Block**: followed by the number of data points, the maximum value of which is $N_{freq} * N_{source} * N_{site} * N_{type}$. A table of data parameters is given below it. The first column of the data table is the frequency index for each datum, the second column is the source/transmitter index, the third column is the site index and the fourth column gives the data type index. The last two columns give the corresponding data value and associated standard error.

#### 4.1.2 MT data file format

The MT data file contains information about MT measurements. Different from the 1D CSEM data, the transmitter and receiver locations are not needed for 1D MT inversion. Following is an example of MT data file:

```
# Format:           MTData_1.0
# Description:      generated in Wed Mar 17 11:48:37 2021
Frequencies (Hz):       40
3.2000e+02
2.3957e+02
...
1.00000e-04
DataType:  2
realZxy
imagZxy
Data Block:    80
# FreqNo.  DTypeNo.    Value          Error
1        1    5.704015e-01    2.901863e-02
1        2    6.165296e-01    3.082017e-02
```

```
2         1    4.713291e-01    2.409851e-02
2         2    5.563706e-01    2.789044e-02
...
40         1    5.769041e-04    2.970454e-05
40         2    6.286499e-04    3.033997e-05
```

The following is an brief explanation of the keywords of the MT data file.

- **Frequencies**: followed by the number of frequencies, and the frequency values (in Hz) are listed below it.
- **DataType**: specify the data types used for inversion. For MT the allowed **DataType**s are the real and imaginary values of the MT impedance.
- **Data Block**: followed by the number of data points, the maximum value of which is $N_{freq} * N_{type}$. A table of data parameters is given below it. The first column of the data table is the frequency index for each datum, the second column gives the data type index. The last two columns give the corresponding data value and associated standard error.

### 4.1.3 TEM data file format

The TEM data file contains information about TEM surveys, including the size of the transmitter loop, and a table of time channels, data values and standard errors. Following is an example of TEM data file:

```
# Format:           TEMData_1.0
# Description:      generated in Tue Mar 23 15:00:14 2021
LoopWidth(m):       20  20
Data Block:    31
# timechannel        value           error
1.00000000e-06    1.827007e-02    9.400605e-04
1.25892541e-06    1.360854e-02    6.848484e-04
...
7.94328235e-04    2.079405e-09    1.053918e-09
1.00000000e-03    1.554934e-09    1.026795e-09
```

The following is an brief explanation of the keywords of the TEM data file.

- **LoopWidth**: followed by the length and width of the transmitter loop (in meters). For current version, only concentric loop configuration is supported.
- **Data Block**: followed by the number of data points. A table of data parameters is given below it. The first column gives the time channels (in seconds) of the TEM survey, while the second and third column give the corresponding data value (electromagnetic force) and associated standard error.

## 4.2 Startup File

The startup file describes the options that control the transdimensional MCMC inversion. It is composed of a list of option keywords and corresponding values. Following is an example of the startup file:

```
surveytype:       mt
datafile:         mt1data.dat
burninsamples:    300000
totalsamples:     1000000
numberoflayer:    2    30
minthickness(m):  100
zcoordinate(m):   1.0  50000.0 0.05
resistivity:      0.1 1e4 0.05
numberofbins:     400 400
credinterval:     0.9
```

The following is an brief explanation of the keywords of the startup file.

- **surveytype**: followed by a string indicating the type of EM surveys, which can be ’`mt`’, ’`csem`’ or ’`tem`’.
- **datafile**: followed by a string that indicates the file name of the data file used in the inversion. Note that the datafile could be an edi file for **MT** survey, which would be automatically converted to the required mt data format by the package.

- **burninsamples**: number of samples for burn-in period used in the estimate of the posterior probability distribution.
- **totalsamples**: number of total samples the Markov chains will produce.
- **numberoflayer**: followed by two real values, which give the minimum and maximum number of layers allowed during the inversion.
- **minthickness**: indicates the minimum layer thickness (in meters) allowed during the inversion.
- **zcoordinate**: followed by three real values. The first two give the minimum and maximum depths (in meters) of the layer interfaces allowed during the inversion, the last one is the standard deviation (in percent) when perturbing the value of the interface depth.
- **resistivity**: followed by three real values. The first two indicates the lower and upper bounds for layer resistivities (in $\Omega m$), the last one is the standard deviation (in percent) when perturbing the value of the resistivity of the layers.
- **numberofbins**: followed by two integer values, which indicate the number of depth bins and resistivity bins for analyzing the posterior distribution of model parameters.
- **credinterval**: a real value between 0 and 1, which indicates the credible interval of interest.