FE2D_DomainDecop

Release 1.0

Jianbo Long

Sep 29, 2021

CONTENTS:

Overview

This is the documentation page for the FORTRAN code developed by Jianbo Long in support of research in domain decomposition methods at Prof. Ronald Haynes's group. For any questions, please contact me at jl7037@mun.ca.

Currently, the FORTRAN code only acts as a "forward solver" in a domain decomposition algorithm, i.e., the code only computes the solution to a given partial differential equation (PDE) over the global domain or a local subdomain of the problem. For computing the global solution, an initial boundary condition is required; whereas for a subdomain solution, the boundary values of the subdomain are extracted from the global solution and the initial boundary values.

The FORTRAN code uses a finite element method and solves only 2-D PDEs. The finite element method implemented here uses unstructured triangular meshes in the spatial discretization of the problem domain. In the current code, only Maxwell's equations for 2-D magnetotelluric (MT) problems are solved.

Contents

Note: Due to a technical problem from Readthedocs, the usual webpage search by `ctrl+f` for keywords on this page is not functioning properly. However, the quick search bar at the top of this page works fine. Also, `ctrl+f` search can be used in the PDF version of this documentation.

CHAPTER

ONE

GET STARTED

To run the code, follow these steps:

Step 1: Compile the code using the given Makefile. Make sure the required libraries are installed first. For details about how to compile the code, see *Compilation* below.

Step 2: Generate the mesh for the given 2-D problem. To do this, it is recommended to use the external program Triangle to generate the mesh files needed as part of input files for the code.

Step 3: Specify the modelling parameters (e.g., MT frequency). Use the templates of input files to make sure all required parameters are provided in these files. The details of input files are discussed in *Input files*. Note that the code requires **one special file** be passed to the (running) program as an argument during the running; that special file itself contains information of all other input files. For example:

```
>>> FE2D -iFWD_input.dat
```

In the above, “FE2D” is the name of the compiled FORTRAN program, “FWD_input.dat” is the name of the special input file, and “-i” is one of the flags of the program and is used to specify the name of the input file.

1.1 Compilation

To compile the FORTRAN code, use the **makefile** provided in the source files. The compilation has been mostly tested on Linux like systems. Nevertheless, the process is the same for other operating systems. The Makefile is likely to be modified so that the paths to all source code files and libraries are correctly set on a local machine. Before compiling, make sure the following tools are installed:

1. FORTRAN compiler: `gfortran` and `ifort` are good options.
2. MUMPS solver: this is a direct solver for matrix equations (also see *modelling_parameter.in*). For download and installation of MUMPS, please see its manual here. Typically, its dependencies, such as BLAS, LAPACK and SCOTCH, need to be installed first. The installation guide of MUMPS is an excellent reference for all details of installing. Once MUMPS is installed, modify the makefile to make sure the compiler can find all these libraries. MUMPS version 5.3.3 has been used in the current FORTRAN code here. Since MUMPS is contantly developed and maintained by its developers, this version is recommended to install, but any subsequent version of MUMPS should work as well. Earlier versions of MUMPS **are not recommended** to use to run this code. When a newer version of MUMPS is installed on your machine, the FORTRAN code may need to be modified.

1.2 Mesh generation

The required mesh files for the code are .node, .ele, .edge and .neigh files generated from Triangle. The file names and their locations are defined in *mesh_parameter.in*. For the requirements of the FORTRAN code, each region of the problem domain needs to be assigned an integer marker when generating the mesh files; this is typically done in a .poly file that is required for generating the mesh. The main purpose of using regional markers is to allow modellings over complex domains where the physical or material property is heterogeneous across the whole computational domain.

Generated mesh files, along with regional marker information, are provided as inputs to the modelling code. For details of regional marker requirements in this code, see *mesh_parameter.in*.

1.3 Input files

There are a few input files required for running the forward solver. Examples of the templates in the following can be found in the Git repository.

1.3.1 `FWD_input.dat`

This is the special file passed to the program when running the code. A template of the file is given as:

```
# --put all input files for the forward modelling in one place. This file is␣
↪the direct input file for modelling program.
# -- "Name Value" format; "value" has the file name which can include the path␣
↪info.

input_path          "/media/jack/NewVolume/3DEM_Jianbo_test_data/MT2D_3layer/"␣
↪    # path info for all files defined here (if they are in one place)
input_modelling     "modelling_parameter.in"       # general modelling␣
↪parameters
input_EM_generic    "EM_generic_parameter.in"
input_mesh          "mesh_parameter.in"
```

As we can see, a general rule is that any texts after "#" are considered as comments. The information in the file is listed in a "**Name Value**" format: "**Name**" is the parameter name recognized by the code and is not allowed to change; and "**Value**" is the value of that parameter. There should be at least a space between "**Name**" and "**Value**".

This special file in the above example has five string parameters (i.e., parameter value should be a string) and all parameter names are case-sensitive:

- `input_path`: the path for all other input files defined here, for example, the file `modelling_parameter.in` defined as the value to the parameter `input_modelling` should be found using this path.
- `input_modelling`: its value is a file name; that file contains general modelling parameters and their values.
- `input_EM_generic`: its value is a file name; that file contains general parameters related to EM problems (MT is a special case of EM).
- `input_mesh`: its value is a file name; that file contains information of mesh files used by the code.

All paths and file names should be double or single quoted.

1.3.2 modelling_parameter.in

An example template of the file is:

```
# a list of general parameters for EM modelling (format in each line: Name␣
↪Value)
    DataType             CM     # 'CM' (complex) or 'RE' (real)

# --data output control
#    Measurement_file    '/media/jack/NewVolume/3DEM_Jianbo_test_data/MT2D_
↪3layer/measurement_sites.node'     # a .node file containing pre-defined␣
↪measurement sites
    outputfilepath      '/media/jack/NewVolume/3DEM_Jianbo_test_data/MT2D_
↪3layer/FE_out/global/'

# --- finite element method
    FEdegree            1    # 1 (linear) or 2 (quadratic)
    global_FE_search    f    # logical parameter;  "t/T": yes (to use a global␣
↪FE search algorithm); "f/F": no

# --- linear system solver
    Iter_solver         f
    iter_solver_name    ''    # 'GMRES', 'BCGSTAB'
    solver_verbose      f
# --- domain decomposition
    Domain_mode      1    # 1 (global solution) or 2 (subdomain solutions)
    BoundaryValueFile   '/media/jack/NewVolume/3DEM_Jianbo_test_data/MT2D_
↪3layer/MT_bc.txt'   # file containing global boundary values
```

For MT problems, always choose `DataType` as "CM" (i.e, complex-valued data type). The parameter `outputfilepath` controls where to place all output files from the program. For finite element algorithm configurations, a combination of linear degree (`FEdegree` is "1") and not using global search (`global_FE_search` is "f") provides the fastest computation.

For the matrix equation solver, two Krylov iterative methods, GMRES and BCGSTAB (Bi-CG stabilized), are provided here. To use an iterative solver, the iterative solver parameter `Iter_solver` should be set as "t" or "T" (i.e., true) and give `iter_solver_name` the value of the chosen iterative solver. If `Iter_solver` is "f" (i.e., false), then a direct solver will be used and the string parameter `iter_solver_name` can be set as an empty string (i.e., its value is just a pair of single or double quotes). In this case, verbose information of the direct solver can be turned on/off by setting `solver_verbose` as "t"/"f".

For the domain decomposition part, currently the code can solve for either a global solution or local solutions (on subdomains). The choice is made through the parameter `Domain_mode`: 1 for global solution and 2 for local solutions, as shown in the template file. Before attempting the local solutions, a global solution must be already available. To solve the global domain problem, the boundary values are read from file first. A boundary value file that provides **initial values at all degrees of freedom** (i.e., including both boundary and interior degrees of freedom) is supplied via the string parameter `BoundaryValueFile`. Details of the boundary value file are discussed in *Boundary Value File*. When `Domain_mode` is set to be 2 (i.e., to solve for local solutions), the value of `BoundaryValueFile` will be ignored.

1.3.3 Boundary Value File

An example of the boundary file is:

```
2000
1   value1  value2 value3 value4
2   value1  value2 value3 value4
3   value1  value2 value3 value4
.
.
.
```

In this file, the format follows a .node file (see mesh file discussions in *Mesh generation*) and there should be no comments. The first line has only one integer value (e.g., “2000” here), which is the number of records in the rest of the file. These records list the initial values of the function that is solved in the problem at **all nodal positions** (in 2-D domains, the nodes in the mesh are usually the positions of degrees of freedom). The order of these rows/records must follow exactly the same as in the .node file.

The first column data is the indexes. For MT problems, additional 4 columns of data are provided in this file, as seen in the above template file. The first two columns list the **real** and **imaginary** parts of the complex-valued function in TE mode. The last two columns list the initial complex values in TM mode (again, these initial values contain the boundary values and non-boundary values). If just one particular mode equation needs to be solved (by default, both equations are solved in the code), then simply supply the corresponding boundary values with zeros. However, all 4 columns of data values are still required in this file.

A convenient way to prepare this boundary value file is to read the .node file generated from meshing, in which all boundary nodes can be marked, assign desired boundary values, and write the results to this file. For example, in *Running example 1: MT modelling*, all boundary nodes are marked by “1” and all interior nodes are marked by “0” in the example .node file, which is generated by Triangle.

1.3.4 EM_generic_parameter.in

An example template is:

```
# a list of general parameters for EM(Name Value)
freq               0.01   # Only one frequency case, unit Hz

#-- a list of conductivities (unit: S/m), or material property, for different␣
↪mesh regions
condRegionFile     '/media/jack/NewVolume/3DEM_Jianbo_test_data/MT2D_3layer/
↪list_regional_cond.txt'
```

In this file, the regular frequency (in Hz) and a conductivity list file name need to be defined. The file name can include an absolute or relative path. For the content of the conductivity list file, see *list_regional_cond.txt*.

1.3.5 mesh_parameter.in

An example template is:

```
# a list of mesh parameters for EM modelling (Name Value), the place of mesh␣
↪files needs to be specified here.
nregion            3                         # the region number found in␣
↪regionMarkFile will take precedence over this value
regionMarkFile    "/media/jack/NewVolume/3DEM_Jianbo_test_data/MT2D_3layer/
↪list_regionMark.txt"        # can include path info
meshfilepath      "/media/jack/NewVolume/3DEM_Jianbo_test_data/MT2D_meshes/"  ␣
↪  # directory of mesh files
basefilename      "MT_2D_3layer.1"          # the base part of names of mesh␣
↪files (i.e., without ".node", ".ele", etc)
```

The definitions of the above parameters are:

- `nregion`: integer parameter; number of homogeneous regions of conductivity (or material property) in the mesh.
- `regionMarkFile`: the file name of list of regional markers in the mesh. Each homogenous region is assigned an integer marker value in order to distinguish itself from other regions. The order of listing the marker values can be specified by the user when generating the mesh. See *list_regionMark.txt* for an example of this file.
- `meshfilepath`: string parameter; the path of all mesh files (see *Mesh generation*).
- `basefilename` string parameter; the "base" part of the file names of mesh files. By default, all mesh files generated using the above-mentioned external program should share this part in their names. For example, if the node file is named "foo.node", then `basefilename` should be "foo".

1.3.6 list_regional_cond.txt

An example template is:

```
#  conductivities (S/m) for different mesh regions (including air layer)
L  3
1  1.e-8
2  0.01
3  0.01
```

In this file, the conductivities (i.e., material properties, unit: S/m) of different uniform regions are defined. The first non-comment line must begin with "L", which is followed by the number of actual regions in the mesh/problem domain. In this example, there are 3 regions in the mesh. Subsequent lines list the conductivity of each region. The order of this list must be the same order that is used to define different regions (see *mesh_parameter.in*) during meshing.

1.3.7 list_regionMark.txt

An example template is:

```
# first line: begins with 'L', then number of uniform conductivity regions
# subsequent lines: which region, region marker(only integers)
L  3
1    10
2    20
3    30
```

The meanings of the parameters here follow exactly the same as those in *list_regional_cond.txt*, except here the information is the assigned markers (integer values) for each region. These integer values should be distinct from each other and should be exactly the same integer markers used when generating the mesh (see *Mesh generation*).

1.4 Output files

By default, only modelling solutions are written to the disk making the output files. The path where all output files go is set in *modelling_parameter.in*.

CHAPTER

TWO

DOMAIN PARTITION (COMING UP)

CHAPTER

THREE

RUNNING EXAMPLE 1: MT MODELLING

The first example demostrating how to run the code is to simulate the MT data over a 3-layer conductivity model, which is illustrated in the following diagram:

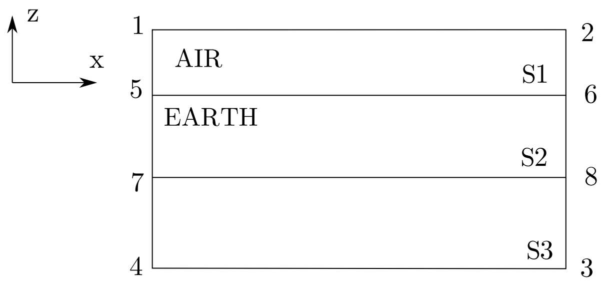

Fig. 1: A diagram of the 3-layer MT conductivity model.

To solve for the MT equations for both TE and TM modes on the global domain, these steps can be followed:

1. Prepare the mesh files. The first thing is to define the computational domain before meshing, which is typically done using a .poly file when using Triangle. An example .poly is here. Following the manual of Triangle, generate all necessary mesh files (click here for example mesh files), as shown in the following snapshot of the domain:
2. Prepare all input files based on the mesh and modelling parameters. Most introduction of the input files is actually based on this example. Refer to the details of every individual input file about setting each parameter. A good practice is to place all input files of running parameters into one directory and put all mesh files into another. Also prepare a new directory for output files from the running. On operating systems other than Linux-like systems, the output directory specified in the input files may not be created successfully. In this case, one can just manually create the directory first.
3. Once the FORTRAN program is compiled, open a terminal and execute the program like:

   ```
   >>> FE2D -iFWD_input.dat
   ```

   Again, “FE2D” is assumed to be the name of the compiled FORTRAN program, “FWD_input.dat” is assumed to be the name of the special input file, and “-i” is the necessary flag to specify the name of the input file.

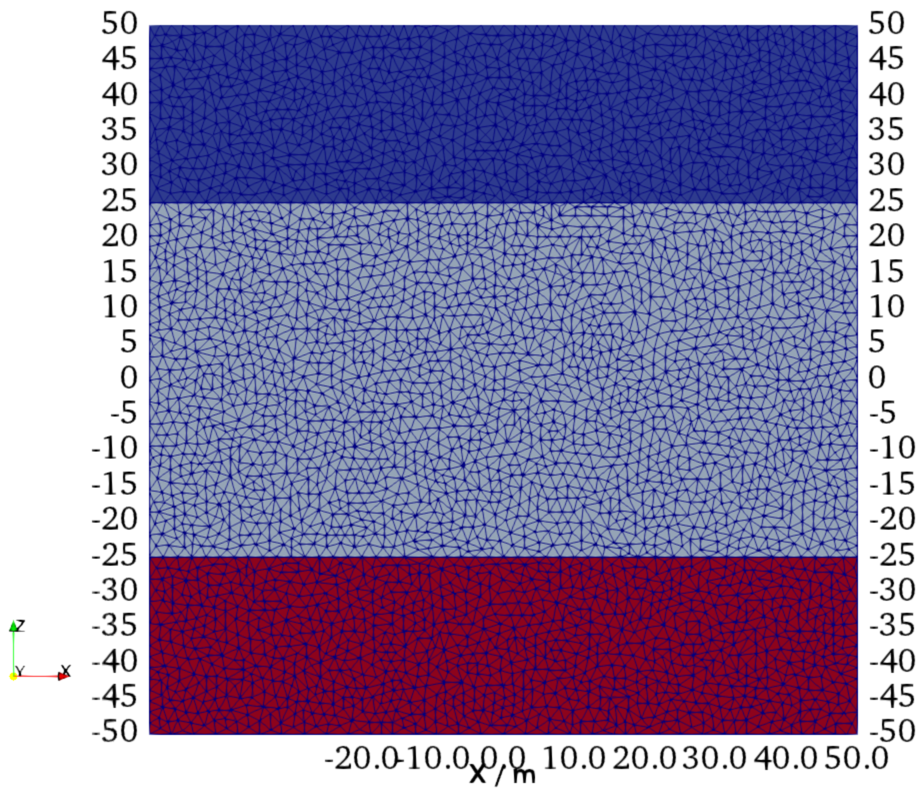

Fig. 2: Triangle mesh of the 2-D 3-layer model.

CHAPTER

FOUR

INDICES AND TABLES (COMING UP)

- genindex
- modindex
- search