## Documentation of SFD\_EXCMG

Contents

Introduction 1

Installation 1

Input files 2

Output files 4

Example run 4

References 7

单击此处输入文字。

**Introduction**

The SFD\_EXCMG is an **open-source** program for **large-scale** MT forward modeling, which utilizes an **extrapolation multigrid method** to accelerate the solving of linear systems arising from **staggered-grid finite difference** (SFD) discretization of the curl-curl equation. The program is developed for complex geo-electrical settings, thus it supports **arbitrary anisotropic** conductivity and **undulating topography**. One can control the total number of levels and the tolerance on the finest mesh to see the differences. The formulations for calculating the background field are from Josef and Fernando (2002). One can refer to the paper by Pan et al.(2024) for more details of the novel extrapolation multigrid method.

**Caution:**

1.This version of SFD\_EXCMG only supports **mesh coarsening** and **semi-uniform mesh** input, which means the mesh is uniform in the central part and with fixed mesh extension factor in each direction (x,y,z), e.g., 1.08 for x and y, 1.15 for z.

2.The **AGMG** solver developed by Notay (2010) is contained in this program as an optional solver, however, we do not recommend it, since the curl-curl type equation is rather **ill-conditioned**, and AGMG is mainly developed for elliptic problems. One can test it if he/she is interested, but the convergence of the solver is not ensured unless some regularization term is added (Li et al, 2022). The AGMG can only be called in the **Linux version**, for now.

3.There are **2** examples contained in this package: the **DTM1.0** model (from the dublin website) and the **Cascadia** model (from the WSINV3DMT program). If the users want more, they can directly contact the author.

4.The program can only accept **homogeneous half-space** or **layered medium** as the background, models with vertical electrical interfaces like COMMEMI2D-0 (Zhadnov et al, 1997) are not supported.

4.The code only supports **serial** running mode for now, and in the future, it will be parallelized using MPI and OpenMP to support multi-frequency and multi-source modeling, which could benefit the inversion of large amounts of MT data.

**Installation**

The package can downloaded from the following link:

https://github.com/eqfwrg424535/sfd\_excmg.git

The SFD\_EXCMG code is written with Fortran 90, and one can easily build the program on Windows or Linux devices.

For Windows users, they can build the program using powerful tools and softwares like Visual Studio, equipped with some certain Fortran compilers. We recommend the users to use Intel compilers like Intel Visual Fortran Compiler XE 19.0 or OneAPI Toolkit (2021.0 or newer), since the program is dependent on the direct solver mkl\_pardiso from Intel libraries.

For Linux users, the program can be built using standard Makefile and make command. And again, Intel compilers should be installed in advance. The Makefile is already contained in the package. Before making, one should modify the following paths to their own, in case of compilation error.

mkllib=/opt/intel/oneapi/mkl/2022.0.2/lib/intel64

mklinc=/opt/intel/oneapi/mkl/2022.0.2/include

Please double check that you have compiled the code; that you are running the code on the same system as the one you used to compile it; that you are running the code correctly; that your file formats are correct. Please refer to this user guide for details.

If you have any questions, please feel free to contact us with the following e-mails:

[jinxuanwang@cug.edu.cn/](mailto:jinxuanwang@cug.edu.cn/) jinx1219741@163.com

**Input files**

One should offer the following two files to run an MT model:

*fname.fwd; fname.dat*

where *fname.fwd* is the main control file that contains the modeling control parameters, mesh information and model conductivities, and *fname.dat* is the input file containing frequency and receiver information for forward modeling.

The file formats are similar to those in ModEM (Egbert and Kelbert, 2012), while the conductivities are stored in a *ncell\*6* array for anisotropic medium, with *ncell* stands for number of cells.

**.fwd file**

The *.fwd file* is composed of a header of description followed by a list of model parameter values. Below is an example, with comments for explanation of each line.

#The DTM1.0 model generated with Python3

5 1.000000e-10 1 1 # grid layers(>=3), tolerance(<=1e-8), solver type(1--bicgstab;2--qmr;3--gpbicg;4--bicg;5--AGMG), output type(1--impeadances; 2--apparent resistivity and phase)

256 256 200 56 0 # nx, ny, nz, nair, indicator for topography

1.14e+04 1.08e+04 1.03e+04 9.81e+03 9.34e+03 8.89e+03 ...... # x-dir dissection

......

1.14e+04 1.08e+04 1.03e+04 9.81e+03 9.34e+03 8.89e+03 ...... # y-dir dissection

......

5.00e+02 5.00e+02 5.00e+02 5.00e+02 5.00e+02 5.00e+02 ...... # z-dir dissection, excluding air

......

1 # the conductivity type(1--isotropic; 2--axial anisotropic; 3--arbitrary anisotropic)

1.00e-02 # the conductivities (S/m)

1.00e-02

1.00e-02

1.00e-02

1.00e-02

1.00e-02

......

**.dat file**

The *.dat file* contains the information of frequencies and receivers, and an example is as follows:

4 # the number of frequency

0.001000 # the list of frequencies(Hz)

0.010000

0.100000

1.000000

17 # the number of receivers

-3.750000e+04 0.000000e+00 0.000000e+00 # the coordinates of receivers (x,y,z)

-3.250000e+04 0.000000e+00 0.000000e+00

-2.750000e+04 0.000000e+00 0.000000e+00

-2.250000e+04 0.000000e+00 0.000000e+00

-1.750000e+04 0.000000e+00 0.000000e+00

-1.250000e+04 0.000000e+00 0.000000e+00

-7.500000e+03 0.000000e+00 0.000000e+00

-2.500000e+03 0.000000e+00 0.000000e+00

0.000000e+00 0.000000e+00 0.000000e+00

2.500000e+03 0.000000e+00 0.000000e+00

7.500000e+03 0.000000e+00 0.000000e+00

1.250000e+04 0.000000e+00 0.000000e+00

1.750000e+04 0.000000e+00 0.000000e+00

2.250000e+04 0.000000e+00 0.000000e+00

2.750000e+04 0.000000e+00 0.000000e+00

3.250000e+04 0.000000e+00 0.000000e+00

3.750000e+04 0.000000e+00 0.000000e+00

The Python script for generating the input files is also offered in this package, one can read them for reference.

Beside the above parameters, one can tune some others in the *global\_para.f90*, e.g, the tolerance and interval of conducting divergence correction, whether output the convergence history or not, the preconditioner type (SSOR as default, Jacobi and ILU for option), etc. One can also use the *output\_div* switch to output some information of the temporary***E***, e.g, the divergence of the electric field, if he/she is interested in the divergence correction technique.

**Output files**

The main output file (.res) contains the responses at the receivers, ordering with frequency-impedance type (zxx,zxy,zyx,zyy)-receiver. One can also choose to output the tippers (tzx,tzy), for some other uses. Besides, as declared above, the convergence history of the smoothing process can also be exported.

The *.res* file is organized as follows:

# x-coor, y-coor, z-coor, freq, imp, real(z)/rho\_a, imag(z)/phi\_a

-1.100000e+04 -1.100000e+04 0.000000e+00 1.000000e+00 1 -6.159788e-04 -9.965504e-05

-1.100000e+04 -9.000000e+03 0.000000e+00 1.000000e+00 1 -1.132267e-03 -4.210711e-05

-1.100000e+04 -7.000000e+03 0.000000e+00 1.000000e+00 1 -1.029098e-03 -3.466318e-04

-1.100000e+04 -5.000000e+03 0.000000e+00 1.000000e+00 1 -8.413799e-04 -4.948424e-04

**.....**

The first 3 columns are the coordinates of the receivers. The 4th column is the frequency, and the 5th column is the type of data, with 1~6 representing the 4 components of impedance tensor (or the apparent resistivity and phase according to output type) and 2 components of the magnetic tipper. The last two columns are the values of the corresponding data. For output type 1, the real and imaginary parts of an impedance or tipper component, and for type 2, the apparent resistivity () and phase(◦).

An example of the convergence history file is given below.

122 # the number of iterations

1.00000e+00 # relative residual of each iteration (not necessarily starts from 1)

5.34555E-05

.......

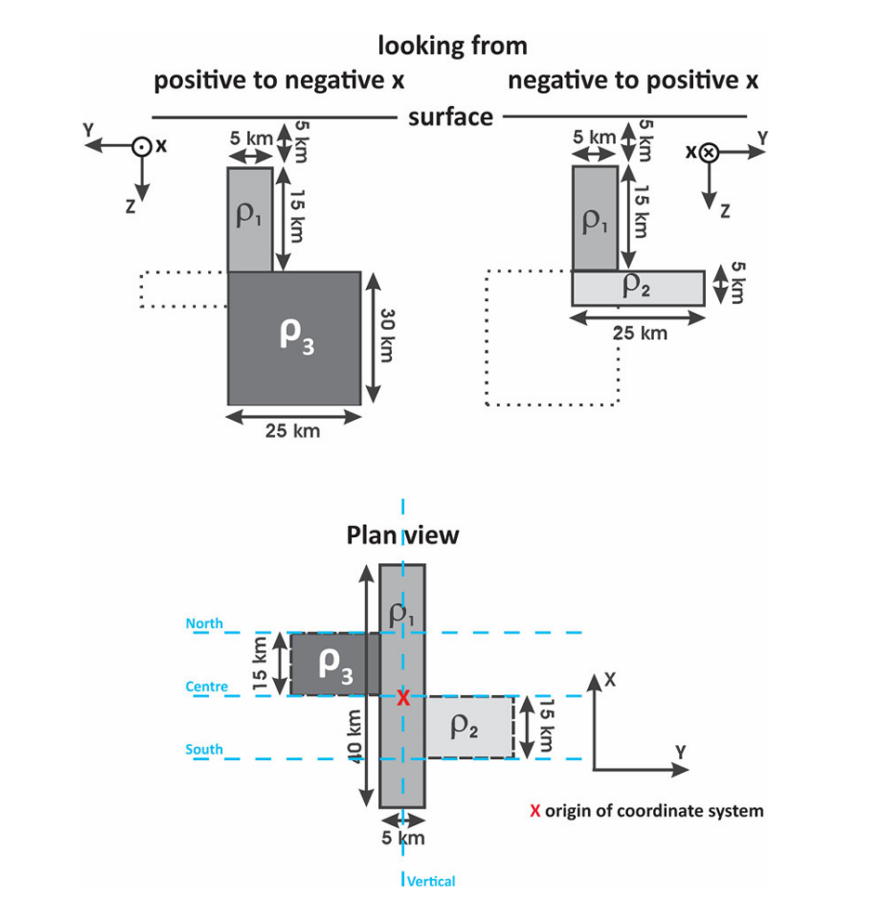
**Example run**

The first example is the DTM1.0 model from the dublin website, and the sections are shown in Fig.1. The conductivities of each block are listed in Table 1. The example is stored in the directory ./example/dtm1. The mesh used for this example is 128×128×192, with 64 layers of air. The number of grid layer is 4, and the tolerance is 10-10. Once one got the code compiled and every input file ready, they can simply run the code with

./main -f example/dtm1

where -f points the path of the model, and the program will automatically search for the input files in the given path, and run with the customized parameters in the .fwd file.

The following results were from the test on a high-performance computing server with Intel(R) Xeon(R) Gold 6248R CPU @ 3.0GHz CPU and 384 Gb RAM.



**Figure 1.** The Dublin Test model 1(DTM1) model (Marion et al, 2013). On top two sections are shown, one across body 3 (left) and one across body 2 (right). Additionally, a plan view of all three bodies is shown. The background is a 100 m homogeneous half-space and the dimensions and resistivities of the blocks are listed in Table 1. The blue dashed lines represent the four profiles with 5-km site spacing (North, Centre, South: 16 sites each, Vertical: 11 sites).

**Table 1.** Dimensions and resistivity values of the three blocks in the Dublin Test Model 1 (DTM1). For x-, y-and z-direction, the extent of the blocks is specified.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | x (km) | y (km) | z (km) | resistivity () |
| Body 1 | -20~20 | -2.5~2.5 | 5~20 | 10 |
| Body 2 | -15~0 | -2.5~22.5 | 20~25 | 1 |
| Body 3 | 0~15 | -22.5~2.5 | 20~50 | 10000 |

The results will be exported in the *dtm1.res* file, and one can plot the subsurface responses with it. The results of 0.1 Hz along x=0 are shown in Fig. 2, and users can find that the results of our algorithm correspond well with those from references (Pan et al, 2022). The biggest relative error of apparent resistivity is around 3.2%, and the biggest phase error is 1.38°̊.

The iteration numbers and computational time for this example are give in Table 2. And one can find that our EXCMG algorithm is rather efficient, since it requires less than 30 iterations on the finest grid, much fewer than the other two. For the lowest frequency (0.001Hz), the computational time of EXCMG is around 12.8% that of SSOR-BiCGStab, and 26.7% that of SSOR-GPBiCG. The memory usage of this example is around 9.08 Gb.



**Figure 2.** The subsurface responses of DTM1.0 along y = 0 at 0.1 Hz, compared with those from our previous work (Pan et al., 2022).

**Table 2.** The iteration numbers and computational time (in seconds) of SSOR-BiCGStab, SSOR-GPBiCG, and EXCMG (with 4 levels) for the DTM1.0 model at 4 different frequencies.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| f/Hz | SSOR-BiCGStab | | SSOR-GPBiCG | | EXCMG | |
| Iter | CPU/s | Iter | CPU/s | Iter | CPU/s |
| 0.001 | 247 | 719.90 | 106 | 348.41 | 13 | 90.93 |
| 0.01 | 142 | 356.98 | 229 | 749.73 | 19 | 78.27 |
| 0.1 | 146 | 363.24 | 132 | 424.66 | 28 | 98.80 |
| 1 | 146 | 363.61 | 111 | 359.97 | 14 | 58.74 |

Another example is the Cascadia model from the WSINV3DMT program, extracted from real-world MT explorations. One can find the corresponding input files in the directory ./examples/cascadia, along with the Matlab script for generating the model. The results will not be shown in this documentation, and one can contact the author for the data and results if he/she is interested.

**References**

dublin website, 2008. DTM1.0: The dublin model proposed at the 2nd annual st.patrick’s workshop on mt 3d inversion. [www.dias.ie/mt3dinv/3D\_forward\_model.html](http://www.dias.ie/mt3dinv/3823D_forward_model.html)

Egbert, G.D., A., K., 2012. Computational recipes for electromagnetic inverse problems. Geophysical Journal International 189, 251–267.

Guo, R., Wang, Y., Egbert, G., Liu, J., Liu, R., Pan, K., Li, J., Chen, H., 2022. An efficient multigrid solver based on a 4-color cell-block Gauss-Seidel395 smoother for 3D Magnetotelluric forward modeling. GEOPHYSICS 87, E120–E133.

Josef, P., Fernando A.M., S., 2002. Magnetotelluric impedances and parametric sensitivities for 1-D anisotropic layered media. Computers & Geosciences 28, 939–950.409.

Li, J., Liu, J., Guo, R., Liu, R., Wang, Y., Chen, H., 2022. Extension of the regularization technique to controlled-source electromagnetic modeling in general anisotropic conductivity media. Geophysics 87, E243–E251.

Marion P.M, Pilar Q, Alan G. J, et al. 2013. Magnetotelluric 3-D inversion—a review of two successful workshops on forward and inversion code testing and comparison. Geophysical Journal Internationl 193, 1216–1238.

Notay, Y., 2010. An aggregation-based algebraic multigrid method. Electronic Transactions on Numerical Analysis 37, 123–146.

Pan, K., **Wang, J**., Hu, S., Ren, Z., Cui, T., Guo, R., Tang, J., 2022a. An efficient cascadic multigrid solver for 3-D magnetotelluric forward modelling problems using potentials. Geophysical Journal International 230, 1834–1851.

Pan K., **Wang J.**, Liu Z., Ou Z., Guo R., Ren Z. 2024. An efficient cascadic multigrid method combined with regularization technique for 3-D electromagnetic finite-element anisotropic modeling. Geophysics, 89(6), E241-E253.

Siripunvaraporn, W., Egbert, G., 2009. WSINV3DMT: Vertical magnetic field transfer function inversion and parallel implementation. Physics of the Earth and Planetary Interiors 173, 317–329.

Zhdanov M S, Varentsov I M, Weaver J T, et al. 1997. Methods for Modelling Electromagnetic Fields Results from COMMEMI—the International Project on the Comparison of Modelling Methods for Electromagnetic Induction. Journal of Applied Geophysics, 37(3-4): 133–271.