ModEM: User's Guide: Revision 2.0 for Version 1.2.0

Gary Egbert, Anna Kelbert, Naser Meqbel (COAS/OSU)

April 15, 2025

Contents

1 Overview

ModEM was designed as a flexible electromagnetic modeling and inversion system, written in Fortran 95. Although the code can be (and has been) extended for inversion of more general types of EM data (e.g., controlled source, DC; see Meqbel and Ritter, 2015), here we describe the stable, core system developed for 2D and 3D magnetotelluric (MT) problems. While a primary design goal of the system was to allow simplified extension with regard to data types, modeling codes, parameterization and regularization, and inversion search algorithms (see Egbert and Kelbert, 2012; Kelbert et al. 2014), issues of code modification and development are beyond the scope of this document. The stable MT program has a command-line interface, which controls available program options, and specifies required and optional input and output files. This document is focused on use of these options and files to accomplish the most common MT modeling and inversion tasks.

2 Obtaining ModEM

If you are reading this far, you probably already have obtained ModEM, but for completeness, we summarize here how this is done. ModEM 2D and 3D MT modeling and inversion codes are currently available for non-commercial academic use, subject to the license agreement (see COPYRIGHT). The latest stable version of ModEM can always be obtained from our GitHub repository:

`https://github.com/MiCurry/ModEM-Model`

It is advisable that you obtain the code through the GitHub repository, to keep your version up-to-date with bug fixes and updates.

3 Getting Started: Compiling and Running ModEM

3.1 Directory Structure

The following files and sub-directories will be found:

COPYRIGHT Please get familiar with the Copyright before using this code!

README Explains how to obtain, install and update the code.

doc/ Provides additional documentation, including this user guide.

examples/ Provides a careful set of examples that are known to work.

f90/ The code base, makefiles and configuration scripts.

matlab/ Auxiliary Matlab scripts for file conversion, and others.

3.2 Compiling ModEM

The f90 directory contains several prototype makefiles, as well as some configuration scripts (`****.config`) that can be used for creating makefiles. These configuration scripts are primarily useful after substantial code modifications, and most users will not have cause (or for that matter, any desire) to use these scripts. For almost all purposes you can use the provided makefiles, perhaps with some minor editing. There are several makefile variants for both 2D and 3D MT. For example, Makefile3d for compiling a serial version of the code, and Makefile3d.MPI for a parallel version. *Follow these instructions carefully when modifying Makefiles!* Copy the makefile before editing, to avoid conflict with standard versions in the svn archive, i.e.,

```
cd f90; cp Makefile3d Makefile
```

Then open the copy of the Makefile in your favorite editor, and make sure that the (uncommented) compiler is set appropriately for your system). Some paths may also need to be modified. The `make` command will compile the 3D MT (or 2D MT) code for you (`Mod3DMT` and `Mod2DMT`, respectively).

ModEM can be compiled for a parallel or serial execution. Make sure that you use the appropriate compiler `preprocessing` flag to compile the desired MPI (parallel) or serial version. Also make sure that the `MPICH` library is already installed on your machine, if you want to compile for MPI version. Here are examples of the compiler `preprocessing` flag and directive used to compile the MPI version:

- PGI compiler: -Mpreprocess —> Serial; -Mpreprocess -DMPI —> Parallel
- Intel compiler: -fpp —> Serial; -fpp -DMPI —> Parallel
- GFortran: -cpp-input —> Serial; -cpp-input -DMPI —> Parallel

3.3 Running ModEM: The basics

Running the compiled program with no command line arguments will print a brief summary of usage:

```
Output information to files, and progress report to screen (default).
Usage: Mod3DMT -[job] [args]

[READ_WRITE]
 -R  rFile_Model rFile_Data [wFile_Model wFile_Data]
  Reads your input files and checks them for validity;
  optionally also writes them out
[FORWARD]
 -F  rFile_Model rFile_Data wFile_Data [wFile_EMsoln rFile_fwdCtrl]
  Calculates the predicted data and saves the EM solution
[COMPUTE_J]
 -J  rFile_Model rFile_Data wFile_Sens [rFile_fwdCtrl]
  Calculates and saves the full J(acobian)
[MULT_BY_J]
 -M  rFile_Model rFile_dModel rFile_Data wFile_Data [rFile_fwdCtrl]
  Multiplies a model by J to create a data vector
[MULT_BY_J_T]
 -T  rFile_Model rFile_Data wFile_dModel [rFile_fwdCtrl]
  Multiplies a data vector by J^T to create a model
[INVERSE]
 -I NLCG rFile_Model rFile_Data [lambda eps]
  Here, lambda = the initial damping parameter for inversion
           eps = misfit tolerance for the forward solver
OR
 -I NLCG rFile_Model rFile_Data [rFile_invCtrl rFile_fwdCtrl]
  Optionally, may also supply
      the model covariance configuration file   [rFile_Cov]
      the starting model parameter perturbation [rFile_dModel]
  Runs an inverse search to yield an inverse model at every iteration
[TEST_COV]
 -C  rFile_Model wFile_Model [rFile_Cov]
  Applies the model covariance to produce a smooth model output
[TEST_ADJ]
 -A  J rFile_Model rFile_dModel rFile_Data [wFile_Model wFile_Data]
```

```
  Tests the equality d^T J m = m^T J^T d for any model and data.
  Optionally, outputs J m and J^T d.
[TEST_SENS]
 -S  rFile_Model rFile_dModel rFile_Data wFile_Data [wFile_Sens]
  Multiplies by the full Jacobian, row by row, to get d = J m.
  Compare to the output of [MULT_BY_J] to test [COMPUTE_J]

Optional final argument -v [debug|full|regular|compact|result|none]
indicates the desired level of output to screen and to files.
```

The `-F [FORWARD]` and `-I [INVERSE]` options will suffice for simple applications. For some of the more advanced applications discussed below the `-C [TEST_COV]` option is also needed. The `-R [READ_WRITE]` option can be useful for testing whether your input files are in the correct format. The -v option controls the level of program output; the default is "regular"; if you want to reduce outputs ("none" is *not recommended!*) you can experiment with other output levels; "full " and "debug" would be primarily useful for developers. All of the other options (`-J, -M,` etc.) are primarily useful for development, testing, and some specialized applications that are beyond the scope of this basic documentation.
Sub-directory `examples/` contains some sample input files with instructions for simple cases. These should be sufficient to test your installation, and also will provide a template for your own initial use. The `doc/` sub-directory contains some further information on the ModEM system, beyond the summary of usage provided here.

All of the various options have a required set of arguments, which define input and output file names (`rFile`, and `wFile` in the usage summary, respectively). There are also optional arguments, mostly also specifying additional input or output file names. In all cases the order of arguments must be exactly as indicated in the "Usage" block, so to give an optional argument at the end of the list, all previous optional arguments must also be provided. Thus, for example, with the `-F` option, if you want to specify the input `rFile_fwdCtrl` file, you must also specify the output `wFile_EMsoln` file, even if you do not specifically want the full set of output EM fields, which this option will result in. Before providing a more detailed discussion of the various options, we consider file formats.

4 File Formats

There are four principal file types: Model (providing grid+conductivity; input or output); Data (input or output); Covariance (required to specify topography or bathymetry, or to "freeze" conductivity; e.g., in the ocean, or to modify default smoothing parameters; input); EM solution (electric fields; input or output). There are also two simple optional control files. Some additional files associated with the non-standard (development/testing) options (`-J, -M, -T`, etc.) are not discussed here. The Model, Data, Covariance, and control files are all flat ASCII files. The EM solution file is very large, and thus uses a binary format. In typical usage (e.g., nested modelling; see below), the EM solution files from one run are used as inputs to a second run. Care should be taken if the files are moved between computers, as binary formats may possibly be incompatible. There are matlab functions that can read the binary EM solution files, at least when all work is done on a fixed computer (see section on matlab I/O) We summarize formats of the basic ASCII input and output files next.

4.1 Data Files

ModEM data files use the so-called "list format". This ASCII format (with essentially each observation a separate item with full meta-data, in a list) results in larger files than necessary, but is very flexible. As data files are typically not so large (at least compared to other files encountered in 3D MT inversion) the size is not usually much of an issue. Note that data files are required both as inputs and outputs, for both `-F` and `-I` options. In the case of forward modeling on input the data file provides essentially meta data: the list of periods to be run, as well as a list of the types of responses to compute (impedance, vertical field TF, etc.), and locations where these predicted responses will be evaluated. The list format is specific to ModEM, so it requires a detailed description.

ModEM "list format" data files contain one or more blocks of data. Each block corresponds to a single "type" of data (e.g., the full 2×2 impedance tensor), generally for all sites and all periods. Each block consists of an 8-line header followed by the actual data. As an example, the start of a single data block is as follows:

```
# Cascadia data from XML files, error floor 5%, no topography
# Period(s) Code GG_Lat GG_Lon X(m) Y(m) Z(m) Component Real Imag Error
> Full_Impedance
> exp(+i\omega t)
> [mV/km]/[nT]
> 0.00
```

```
> 45.276 -119.634
> 10 109
1.000000E+00 006-006    0.000    0.000    41000.000    39375.000        0.000 ZXX
-1.700416E+03    3.455034E+02    6.167116E+02
1.000000E+00 006-006    0.000    0.000    41000.000    39375.000        0.000 ZXY
1.056768E+04   -1.259106E+04    6.167116E+02
1.000000E+00 006-006    0.000    0.000    41000.000    39375.000        0.000 ZYX
-1.761286E+04    1.617083E+04    6.167116E+02
1.000000E+00 006-006    0.000    0.000    41000.000    39375.000        0.000 ZYY
2.497673E+03   -1.202580E+03    6.167116E+02
1.000000E+01 006-006    0.000    0.000    41000.000    39375.000        0.000 ZXX
-9.602744E+02    3.031632E+02    1.981138E+02
1.000000E+01 006-006    0.000    0.000    41000.000    39375.000        0.000 ZXY
3.743773E+03   -3.790398E+03    1.981138E+02
1.000000E+01 006-006    0.000    0.000    41000.000    39375.000        0.000 ZYX
-6.208288E+03    5.470545E+03    1.981138E+02
1.000000E+01 006-006    0.000    0.000    41000.000    39375.000        0.000 ZYY
2.279075E+03   -1.333596E+03    1.981138E+02
```

The actual data are written in a list following the 8-line header. Each data line (which are split over two lines in the example displayed here, but should be on a single line in the actual data file) in the list contains one data component appropriate to the data type, following the format described in the second line of the header. For example, for the `Full_Impedance` data type in the example above, `ZXX, ZXY, ZYX` and `ZYY` would be data components; for each period/site these data components would be listed on a separate line. Thus a complete set of data components for a data type for even a single site will typically require multiple lines (4 in the example above); however, real and imaginary parts of an intrinsically complex data type (such as impedance components) are given together on a single line, with a common error bar.

Some (or all) components can be missing at any sites and/or for any periods, and any ordering of the list elements is allowed. This is an advantage of the format: it is easy to edit a file and delete or add observations, or to have a program that does this. Note however, that if components are missing for all sites (e.g., you are inverting only off-diagonal impedances for all, or a large group of sites), it is usually better to use a more restrictive data type to minimize internal storage and computations (e.g., use `Off_Diagonal_Impedance` in place of `Full_Impedance`). Multiple data blocks are allowed (typically corresponding to different types), so if a large group of sites had only off-diagonal impedances, and another group had full impedances, it would be most effi-

cient to use two blocks, with distinct data types.

The first two lines of the header are descriptive only (i.e., are primarily for a human reading the file, as indicated by the # at the start of the line. But note that this symbol is not parsed by the program.)

Line 1: # comment

Comment line, intended to describe your data; will be read and replicated in output responses. Max 100 chars.

Line 2: # comment

Comment line, intended to describe the contents of each data item in the block; will be ignored by the code.
The remaining lines give details about the data type. Not all of these are used at present.

Line 3: > data_type

Keyword for the data type stored in this data block. Only specific keywords are supported. For each data type there are a set of additional keywords denoting specific components for the type. These may be real or complex.

For the 3D MT program these are:

Keyword	Allowed components	Real or Cmplx
Full_Impedance	ZXX,ZXY,ZYX,ZYY	Complex
Off_Diagonal_Impedance	ZXY,ZYX	Complex
Full_Vertical_Components	TX,TY	Complex
Full_Interstation_TF	MXX,MXY,MYX,MYY	Complex
Off_Diagonal_Rho_Phase	RHOXY,PHSXY,RHOYX,PHSYX	Real
Phase_Tensor	PTXX,PTXY,PTYX,PTYY	Real

For 2D MT these are:

Keyword	Allowed components	Real or complex
TE_Impedance	TE	Complex
TM_Impedance	TM	Complex

Line 4: > sign_convention

This defines the sign convention assumed for time dependence, which is effectively assumed in Fourier transform (or other) data processing, which can be plus or minus. Here this is written as, "exp(+i omega t)" for clarity. In fact, ModEM searches for a minus sign in this line; if not found, plus is assumed. The data are then converted to the sign convention used internally in the code (which is determined at compile time by the global variable `ISIGN`). Any output data files will follow the convention used in the input file, although internal calculations will follow the convention defined in `ISIGN`.

Line 5: > units

Units for this data type. Internally in the code, the SI units for E/B are used for MT impedances. However, the user is given a choice for this data type. Supported options are:

1. SI units for E/B: [V/m]/[T] (used internally in ModEM)
2. practical units for E/B: [mV/km]/[nT]
3. SI units for E/H: [V/m]/[A/m] = Ohm

The output data will be given in the same units as the input, with the program converting to the internally used units after input/before output. The units supplied in the data file must of course match the actual data. Dimensionless data (such as the vertical magnetic field transfer functions, or inter-station transfer functions) are identified by empty "[]" units. Apparent resistivity and phase are loosely viewed as being dimensionless. It is assumed that apparent resistivity is given in ohm-m; there is no user option on units for this data type.

Line 6: > orientation

of coordinate axis. *THIS IS NOT PRESENTLY IMPLEMENTED IN THE CODE!!* At present all impedances are assumed to be rotated into the same Cartesian coordinate system used by the model grid. This header information may be useful for other programs that plot data, so it would be good practice to set the orientation to a meaningful value, even though the inversion code does not reference this number.

Line 7: > geographic_origin

Latitude and longitude for the (0, 0, 0) point in the data file. Not used in ModEM, but could be useful for plotting, and to match the origin of the model to that of the data. See discussion of coordinate origins in description of model file formats. Can set to zero if unknown since it does not effect program function.

Line 8: > nperiods nsites

These two integers give the number of periods and sites in the data set. More properly, these give the *maximum* number of periods and sites. The program will parse the list of data components, and figure out the actual numbers. Thus it will never hurt to set these numbers to a larger value than you actually need. However, if these numbers are smaller than required to read your data block, a segmentation fault will occur. If unsure, use the command

```
./Mod3DMT -R model_input data_input model_output data_output
```

to validate your model and data files. This will read your files, store them in the internal dictionary, data and model structure, write them out again, and exit. If this does not work, you need to increase the size of the parameters. (This should be fixed!)

In the example given above, each line in the actual data list contains period, site code (up to 12 chars), geographic latitude and longitude, X/Y/Z location in meters, component code (e.g., ZXY), real and imaginary parts, and the error bar. Geographic latitude and longitude are not used by the program, which requires the user to make sure that the correct locations (i.e., X/Y/Z in meters are provided, registerd to the grid; see further discussion in Sec. 4.2.

Of course each specific data type requires a specific format, and consistent header entries, such as units (described above). List element formats for currently supported data types are summarized here.Some data file examples are provided in the `examples/` directory. The `matlab/ioAscii/` directory contains read and write routines in Matlab (at present these reading/writing routines are restricted to the most common data types:

- `Off_Diagonal_Impedance`
- `Full_Impedance`
- `Full_Impedance_plus_Full_Vertical_Components`

For complex impedances and vertical components the data format is:

`Period(s) Code GG_Lat GG_Lon X(m) Y(m) Z(m) Component Real Imag Error`

For apparent resistivity and phase the data format looks like:

`Period(s) Code GG_Lat GG_Lon X(m) Y(m) Z(m) Component Value Error`

For inter-station transfer functions, information about the reference site is also required:

`Period(s) Code GG_LatLon XYZ(m) Ref_Code Ref_LatLon Ref_XYZ(m) Component Real Imag Erro`

For 2D MT, the data format is identical, but the X coordinate is not used or stored in the program.

4.2 Model Files

ModEM support two model file formats, based on previously used formats for 3D resistivity (not conductivity!) models: The first (and the current default) is based on that used in Weerachai Siripunvaraporn's WSINV3DMT program. There are a few differences in the ModEM implementation: (1) ModEM does not support the option to use a set of resistivity codes (i.e., integers 1-9, used to specify a small number of fixed resistivity values); (2) we have added the option to store the resistivity values in natural log or linear formats; (3) an extra line at the end of the file is used to specify a coordinate origin, as discussed below. The second format is identical to that used for 3D models in WingLink (referred to in the ModEM code, perhaps inappropriately, as the "Mackie format".) These two formats are similar, but not identical. At present switching between these model formats requires editing and recompiling the program.

(For completeness here is a very brief summary of the modifications required: read routines are in `f90/3D_MT/modelParam/modelParamIO/*.inc`. All of these files are included in the compiled ModelSpace module (with compiler directives #include). The names of the read and write routines are `read_modelParam_WS(mackie)` and similarly for write. The read/write routines in use are overloaded on `read_modelParam` (and similarly for the write routine; this is done in `f90/3D_MT/modelParam/ModelSpace.f90`), and calls to these generic read/write subroutines are then used in the program to input or output model parameter objects. Both the "WS" and "Mackie" read/write routines (and others that one might want to implement) must have identical interfaces. Then, by changing the names of routine overloaded in the `ModelSpace` interface, the expected file format used for model parameter I/O is changed. Yes, better approaches could be imagined!)

Both of the currently supported model file formats begin with a description of the tensor-product Cartesian grid: cell dimensions in x, y, and z are given (in meters, i.e., $\Delta x_i, i = 1, ..., N_x; \Delta y_i, i = 1, ..., N_y; \Delta z_i, i = 1, ..., N_z;$), followed by a list of linear or log

resistivity values. The two supported formats differ primarily in the ordering of the list of resistivity values.

A critical issue which is independent of file format, is how coordinates implicit in the model grid are related to coordinates specified in the separate data file. The arrays $\Delta x, \Delta y, \Delta z$ define a natural Cartesian coordinate system for the grid, with origin at the outer corner (of model parameter cell (1,1,1)). We refer to this here as the "natural grid coordinate system". The origin of the coordinate system used in the data file (the "data coordinate system" does not have to be (and in fact most often is not) the same. The relationship between the two origins must obviously be defined. This is done in ModEM by giving, in the model file, the coordinates (in natural grid coordinates) of the origin of the data coordinate system. A common approach might be to center the grid on a point chosen near the center of the array of observations. Specifically, this could correspond to the location given by the arithmetic mean of site coordinates, or perhaps the coordinates of a centrally located site. In the data file Cartesian coordinates of all data sites will be specified (in meters) relative to this point. Then, if the coordinates of the reference point given in the model parameter file corresponds to the center of the grid, the array of sites will be centered on the numerical grid. If no origin is specified in the model file (i.e., the final lines are missing, as would be the case if an actual WSINV3DMT model input file were used), the horizontal (x,y) origin is taken to be center of the grid, and the vertical origin the top of the first model layer (nominally the surface of the Earth, but see Sec. 4.3.2). It is the user's responsibility to ensure that a sensible data origin is used, and that data locations are given consistently relative to this origin. When there are geographically fixed features (e.g., a coastline, topography) that need to be represented in the numerical model, it is of course also essential to ensure that consistent projections and coordinate systems are used for model and data file setup. Tools for input setup (e.g., Grid3D; N. Meqbel) will commonly take care of these details. One issue to be aware of is that if you change the grid discretization, and the origin somehow changes (e.g., relative to a fixed feature like the ocean) coordinates in the data file might need to be checked/modified. Another issue to be aware of is the vertical coordinate of the origin. For a flat Earth (no topography) it is natural to take the surface to be z = 0. However, if topography is included then the choice is less clear: 0 might be sea level, or the highest land elevation point in the domain. We discuss this further in Sec. 4.3.2.

Following are further details on file formats:

WS format (default): annotated example:

```
# Written by JavaScript within EM-WebApp   <--(1) header line
```

```
59 54 25  0 LOGE  <-- (2) grid dimensions, and more
165357 152127 99214.01  ... <-- (3) grid dimensions: x
157079 144512 119380  ... <--     grid dimensions: y
100 120 140 160 ...  <--     grid dimensions: z
    <-- (4) resistivity codes (blank line often)
 -1.20397276458951 -1.20397276458951  ... <-- (5) cell resistivities
 ...
 ...
 ...
-1183956 -1088486 0 <-- (6) coordinate origin (last lines optional)
0   <-- (7) coordinate rotation (not used)
```

Explanation/Notes:

(1) Header is not used by program; useful for user to describe model in some fashion. Here, the program that wrote the file is documented.

(2) In addition to the three integers defining grid size (N_x, N_y, N_z), there is a fourth integer in this line, for consistency with the original WSINV3DMT model file format. In the original, this was used to specify the number of integer resistivity codes, with nCodes = 0 used to indicate that resistivities are specified as actual values. This is the only option supported by ModEM, and this integer should be left set to 0. (Otherwise the program will stop). The next entry on the line is a character string specifying LINEAR, LOGE or LOG10 resistivity. In fact the inversion always works in the natural log domain, even if input and output use the linear option. Note that output files will match the convention used in the input file.

(3) The next lines give the grid spacing, in meters. Format is free, and the list for any of the three arrays can span multiple lines. Use a new line for each of x, y, and z.

(4) The next line is where resistivity codes were to be specified in the WSINV3DMT files. This line is left blank here, as ModEM does not support the resistivity code option.

(5) Now the list of cell resistivities (or integer codes, if this option is being used) are given. This is free format, and the full list of cell resistivities will typically span many lines. To fill the array rho(Nx:-1:1,1:Ny,1:Nz) use these read statementsi (note the reversal of order of the index ix)

```
DO iz = 1,Nz
```

```
  DO iy = 1,Ny
     DO ix = Nx,1,-1
         READ(10,*) rho(ix,iy,iz)
      ENDDO
   ENDDO
ENDDO
```

(6) These three numbers provide the origin (x, y and z), in "natural grid coordinates" used for Cartesian data locations. This line is an addition to the WSINV3DMT file format, and is optional; default is 0. In any event, one has to make sure that the XYZ corrdiantes of the sites are refernced to the same origin (see 4.1).

(7) The last (also optional) line in the file is for orientation of the model grid. This can be useful for plotting programs, but is not used by ModEM. It is important to note that while it is possible to rotate the grid to any orientation, all transfer function types must be rotated into the grid coordinate system: x in the impedance is the same as x in the model grid. Again, it is the users responsibility to ensure consistency between grid and data rotations.

For 2D MT, the WingLink (Mackie) model format is used, with the same modification for natural log resistivity. The WingLink 3D format can also be used for model input/output to ModEM, but this requires slight code modification, and recompiling, as discussed above.

4.3 Model Covariance File

Model regularization is handled through a positive-definite smoothing operator, that can be viewed as a prior covariance on the model perturbation–i.e., the deviation of the model parameter (log resistivity) from an assumed model. A prior (or prejudice) model is required at present in the ModEM regularization approach. In practice most users will usually want to use something simple, such as a half space of constant resistivity, but any resistivity model can be used for the prior. In the simplest usage, no covariance file is required; there are default values for all parameters. However, for many purposes (including topography, or a fixed conductivity feature such as an ocean) the covariance file must be used. The covariance file also allows the user to control smoothing length scales separately in all directions, and as a function of depth in each of the vertical layers of the model. An additional parameter defines vertical smoothing independently. One can also switch off (or reduce) the smoothing across a boundary. Air and ocean cells are also

specified through the model covariance file.

Example covariance files are provided in `examples/3D_MT/`, with file extension `.cov`. The file has a 16-line header (Fixed length!) that is ignored by the code. This is followed by grid dimensions, and parameters that define the auto-regressive smoothing scheme. These α parameters should be between 0 and 1; 0 implies no smoothing, larger values correspond to longer smoothing (correlation) length scales. Typical values that we use are 0.1 - 0.3. The smoothing can be specified independently for each vertical level, and for x and y directions. A single parameter defines the vertical smoothing. Note that the smoothing is applied without reference to the actual variable grid spacing. In addition to the parameters α, one can specify how many times the smoothing is repeated (nSmooth). Typically this number is 1-3. Repeating the smoothing implies longer decoration length scales—these increase length scales by a factor of sqrt(nSmooth). Using a smaller α and more smoothing repetitions will result in a more "circular" effective smoothing kernel. With nSmooth = 1 the covariance will be somewhat anisotropic, with longer length scales associated with grid coordinate axes (i.e., features may tend to be elongated in x and y directions, relative to a 45 degree angle). Default values for the smoothing parameters (used if no model covariance file is provided) are $\alpha = 0.3$ and nSmooth = 1. More circular smoothing, with similar length scales would be achieved with $\alpha = 0.2$ and nSmooth = 2.

The covariance file also is used to subdivide the model domain, by defining an integer index for each cell in the grid. The index can be used to enable freezing of some regions (so that cells in the region are not changed during the inversion), or to turn off (or reduce) smoothing across the boundaries between adjacent regions. To turn off smoothing, one provides "exception rules" which define how the covariance should smooth across a boundary between regions tagged with two different indices. All of this information (array of indices for all grid cells; exception rules) is provided after defining the smoothing parameters. The most common situation is including the ocean (with conductivity fixed; index = 9) and inclusion of topography (where cells that are in the air should be frozen to the very high air resistivity; index 0). It is possible to freeze other parts of the model domain, for example in hypothesis testing applications. Thus, to include an ocean or topography in an inversion, it is necessary to provide a covariance file.

Following is a simple example of a covariance file for a small model:

```
+---------------------------------------------------------------------------+
| This file defines model covariance for a recursive autoregression scheme. |
| The model space may be divided into distinct areas using integer masks.   |
```

```
| Mask 0 is reserved for air; mask 9 is reserved for ocean. Smoothing between |
| air, ocean and the rest of the model is turned off automatically. You can   |
| also define exceptions to override smoothing between any two model areas.   |
| To turn off smoothing set it to zero. This header is 16 lines long.         |
| 1. Grid dimensions excluding air layers (Nx, Ny, NzEarth)                   |
| 2. Smoothing in the X direction (NzEarth real values)                       |
| 3. Smoothing in the Y direction (NzEarth real values)                       |
| 4. Vertical smoothing (1 real value)                                        |
| 5. Number of times the smoothing should be applied (1 integer >= 0)         |
| 6. Number of exceptions (1 integer >= 0)                                    |
| 7. Exceptions in the form e.g. 2 3 0. (to turn off smoothing between 3 & 4) |
| 8. Two integer layer indices and Nx x Ny block of masks, repeated as needed.|
+-----------------------------------------------------------------------------+

21 28 11    <-- Grid dimensions excluding air layers (Nx, Ny, NzEarth)

0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 <-- Smoothing in the X direction
0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 <-- Smoothing in the Y direction
0.1   <-- Vertical smoothing

2     <-- Number of times smoothing is applied

1     <-- Number of exception rules
2 4 0.     <-- Exception rule #1: smoothing across region 2, 4 boundary
                this should be a real number; 0. means no smoothing
                repeat these lines for each exception
1 11   <-- Start index array specification; see more details below.

9 9 9 9 9 9 9 9 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
9 9 9 9 9 9 9 9 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
9 9 9 9 9 9 9 9 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
9 9 9 9 9 9 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
9 9 9 9 9 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
9 9 9 9 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
9 9 9 4 4 4 4 4 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 4 4 4 4 4 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 4 4 4 4 4 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 4 4 4 4 4 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4
```

```
9 9 9 9 4 4 4 4 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 9 9 4 4 4 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
```

The index array is specified through a series of blocks, which specify indices for a range of vertical layers. First the vertical layer limits are given (k1, k2) then an array temp(i,j). Then index(i,j,k1:k2) = temp. Here the array temp (for a single range of layers) is read as

```
DO i = 1,Nx
   DO j = 1,Ny
      read(fid,*) temp(i,j)
   ENDDO
ENDDO
```

Blocks are repeated as necessary — a single block is used in the simple example, indicating that the regions extend over all 11 layers (top-to-bottom) of this very small grid.

For 2D MT, Weerachai Siripuvaraporn's model covariance is used. No configuration file is currently allowed.

4.3.1 Including an Ocean

To include an ocean domain, which will have conductivity that is not changed ("frozen") during the inversion run, the covariance file must be used. As noted in the header to the example covariance file index 9 is used to denote the ocean domain that will be frozen to the value specified in the prior model. Note that you also need to specify the conductivity of seawater that you want to use in the prior model; this would typically be roughly 0.3 ohm-m (but in some cases might be quite different – e.g., when modeling a highly

saline body of water such as the Dead Sea). The same index 9 would be used to freeze a structure in hypothesis testing. Future versions of ModEM will allow multiple indices to denote frozen structure, but at present you have to use index 9.

Here is an example of the covariance mask blocks used to specify an ocean:

```
1 1   <-- first layer

9 9 9 9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

2 2   <-- second layer

9 9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

3 11   <-- remaining layers

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

In this example there are just 2 ocean layers, and the remainder of the model is indexed by the default mask 1. Assuming a 100 ohm-m prior (for example) the same pattern would be given in the prior model, with cells that are marked as 9 set to 0.3 (or ln(0.3)) and those marked with 1 set to 100 (or ln(100)). (But note that the order of elements in the arrays differs between covariance and model files!)

4.3.2 Including Topography

To include topography you need to add extra layers to the top of the model, with some (typically most) cells in these layers marked as air. As with ocean cells, the air cells need to be indicated in both the prior model (set to air resistivity–a large positive number "Note: We use resistivity in the model file") and in the covariance file, marked with index 0. Here is a simple example of the covariance file for a simple circular hill (just showing the mask blocks).

```
1 1   <-- top layer -- mostly air +_  top of hill

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
2 2   <-- second layer -- mostly air +_ more hill

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

3 13   <-- remaining layers

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Here we have essentially added the hill by including two additional layers (in the 11 Earth layers in the original example model). In this example all of the Earth cells are marked by the default index 1, and only the air cells in the additional layers have a different mask index, namely 0. Of course you can use multiple indices with topography (e.g., including also an ocean).

If you do include topography you need to be careful about the definition of the vertical coordinates of data sites (specified in the data file), and make sure that you define the data origin (given in "natural grid coordinates" in the model file) consistently. The key points are these: (1) z is positive downwards; (2) the model parameter now explicitly includes the extra layers used to define topography. Thus, the origin for the "natural grid coordinates" (see 4.2) is the outer corner at the top of the uppermost topography layer–this corresponds to z=0 in these coordinates. (3) data sites should be projected onto the surface of the (model) Earth.

One approach would be to define the data origin at sea level (or perhaps at the first layer that has no air cells). Then in data coordinates all of the layers above this surface would have negative z coordinate, and any data sites sitting on this topography should be projected to this level. If you choose the data origin as the highest point in the model (i.e., coinciding with natural grid z-coordinate = 0) then all data sites will have non-negative z-coordinates. A very common error in using topography is to project sites to the wrong z-coordinate, so that sites are effectively inside the Earth, or in the air. The inversion obviously will not work correctly if you do this, so if you are using topography and having problems, double check that you are defining vertical levels consistently in the model and data files. Again, using tools for model and data file setup (such as Grid3D) can help ensure consistency–if these are used correctly!

5 Forward Modeling and Inversion Options

5.1 Forward Modeling

```
Usage: -F  rFile_Model rFile_Data wFile_Data [wFile_EMsoln rFile_fwdCtrl]
```

Required files are:

- `rFile_Model` : input model grid/resistivity file
- `rFile_Data` : input data file (template: defines data types/locations to return)
- `wFile_Data` : output data file (as defined from input)

The optional files are:

- `wFile_EMsoln` :output file for electric field solution (all periods and the 2 polarizations), defined on staggered grid edges
- `rFile_fwdCtrl` : forward solver control file. This allows more control over how the iterative solver works. This file is required for nested modeling, as this is where you specify the file name for the larger scale regional solution used for boundary data (last line below; see section 6.2). The format for the forward solver control file is as follows:

```
Number of QMR iters per divergence correction : 40
Maximum number of divergence correction calls : 20
Maximum number of divergence correction iters : 100
Misfit tolerance for EM forward solver        : 1.0e-7
Misfit tolerance for EM adjoint solver        : 1.0e-7
Misfit tolerance for divergence correction    : 1.0e-5
Optional EM solution file name for nested BC  : nested.esoln
```

The first 6 parameters are numbers and are all required if you use this option. The last line gives the file for nesting and is optional. If it is present, electric fields in the file are used to provide boundary data; if it is not present, default boundary values are used. A hash mark # can be inserted at the start of the line instead The order of lines is fixed, so to use nested modeling you have to specify values for the other 6. Default values are given above. Instead of specifying a file name for forward model control, the user can specify a single real number. ModEM will interpret this as the value of eps = misfit tolerance for the forward solver (I.e., the 4th line in the file).

5.2 Inversion

The simplest usage specifies only the search algorithm, the prior (and starting) model file, and the data file. Default values for forward modeling, inversion control parameters, as well as model covariance, are then used. Additional optional arguments allow more control over the inversion. The general form for the command line is

`Usage: -I NLCG rFile_Model rFile_Data [InvCtrl FwdCtrl CovCtrl StartModel ]`

As usual, arguments have to be in the specified order. Thus, to specify StartModel all other optional arguments must be given.

The required arguments are:

- `NLCG` Inversion search approach (other supported options DCG; only NLCG has been extensively tested and used to date). This argument is required by the -I option, even though most users only use the `NLCG` option.
- `rFile_Model`: input model grid/resistivity file : This is the prior model, which by default is used also as the starting model.
- `rFile_Data` : input data file (template: defines data types/locations to return)

Optional arguments are:

- `InvCtrl`: inversion control file, which has the format

```
Model and data output file name    : Modular
Initial damping factor lambda      : 1.
To update lambda divide by         : 10.
Initial search step in model units : 100.
Restart when rms diff is less than : 2.0e-3
Exit search when rms is less than  : 1.05
Exit when lambda is less than      : 1.0e-4
Maximum number of iterations       : 120
```

The values given above are the default values. The first line is used to define the root for output files (model parameter and data) created by the inversion; these are discussed in more detail below, under Inversion Outputs. As with the forward modeling control file, there is a simpler form for this optional argument, which modifies only a single (the most commonly modified) parameter, i.e., lambda = the initial damping parameter for inversion.

- `fwdCtrl` : forward solver control file, described above under the -F option.
- `CovCtrl`: The covariance file. This is used to define the model covariance(see section 4.3 and related discussion).
- `StartModel` : The starting model file. This must be specified to start from something other than the prior. Proper use of this option is discussed below, in Sec. 6.1.

5.3 Inversion Outputs

For the NLCG inversion 4 files are produced for each each step in the iterative search process. The files produced have the name `root_NLCG_###.rho`, `root_NLCG_###.prm`, `root_NLCG_###.dat`, and `root_NLCG_###.res`, where `root` is the output file root specified in the inversion control file, and `###` is the NLCG iteration number. The output file root defaults to `Modular` if the inversion control file is not used. The ***.rho file is the model parameter file, giving the resistivity estimate at the current iteration. The ***.prm file is also in the model parameter file format, but represents the transformed model parameter ($\tilde{\mathbf{m}} = \mathbf{C_m}^{-1/2}(\mathbf{m} - \mathbf{m_{prior}})$, as discussed below inn Sec. 5.4. The ***.dat file contains predicted data, in the ModEM list format (identical to the input data file), and the ***.res file contains residuals, again in the ModEM list format. (This file is not useful and will be eliminated in future ModEM releases). In addition to the suite of model parameter and data files produced, the inversion provides an ongoing report on results of the various modeling and search steps. These are displayed to standard output and should be monitored by users. Key diagnostics of the search process are recorded in the log file `root_NLCG.log`.

Similar files are produced by the DCG inversion, with the obvious name change. Note in this case each iteration corresponds to a full Gauss-Newton step, so there will be many fewer of these output files. Howwever, this option has seen little actual use so far.

5.4 Apply Model Covariance

This option is used to apply the model covariance (smoothing operator) to a model parameter, and to invert this operation (roughening). The forward covariance operator is primarily used for testing; the inverse is required for some specialized applications discussed below in Sec. 6.1. The format for this option is:

```
-C INV rFile_Model wFile_Model [CovCtrl rFile_Prior]
```

The required arguments are:

- `INV`: this is the inverse (roughening) operator, most often used. `FWD` is the argument used to apply the covariance (smoothing) operator.
- `rFile_Model` and `wFile_Model` are the input and output model.

Optional arguments are:

- `CovCtrl` : the covariance file; if this is not specified default covariance parameters are used.

- `rFile_Prior` : the prior model file; in general this is required for the most common use of the -C option.

Explanation: In the case of the INV option the output file contains the transformed model parameter $\tilde{\mathbf{m}} = \mathbf{C_m}^{-1/2}(\mathbf{m} - \mathbf{m_{prior}})$, where $\mathbf{m}$, and $\mathbf{m}_{prior}$ are the input and prior model parameters, provided in files `rFile_Model` and `rFile_prior`, respectively. If the optional prior model file name is not provided $\mathbf{m_{prior}}$ defaults to 0. The FWD option reverses the transformation, computing $\mathbf{m} = \mathbf{C_m}^{\mathbf{1/2}}\tilde{\mathbf{m}} + \mathbf{m_{prior}}$.
WARNING: the inverse computation may be poorly conditioned if the smoothing is too strong, so it is a good idea to proceed with caution, and check your model after applying the inverse covariance. The need for caution is one reason why we ask the user to do several steps themselves, instead of packaging this as a simpler to use black box.

6 Inversion: Beyond the Basics

Some things that most users will want to do with the inversion require use of the optional arguments, in multi-step procedures. These include restarting the inversion, and nesting a forward or inverse calculation within a larger, more coarsely discretized model domain.

6.1 Restarting the Inversion

The inversion can be started from any model parameter file, not just the prior model. There are many reasons why a user might wish to do this. The simplest case would be if the inversion has stopped due to a time limit on a computer system, or some other reason, and the user wants to continue the run. In this case, to restart the inversion from last iteration (e.g., iteration # 60) use the following command line:

```
ModEM -I NLCG rFilePrior rFile_Data InvCtrl FwdCtrl CovCtrl root_NLCG_060.prm
```

One might also want to change some inversion control or covariance parameters. For example, it is possible that an inversion that has stalled in terms of decreasing the normalized RMS, might be moved out of a local minimum by restarting the inversion with a larger value of the damping factor lambda. Restarting the inversion after removing structures (e.g., setting a conductive body to the prior resistivity), and possibly freezing the model to some value, will be useful for hypothesis testing.

The key point that must be understood to start the inversion from something other than the prior is this: As already noted in the discussion of the -C option, the inversion search is conducted in the "transformed" model space, i.e., we directly minimize the L2

norm of $\tilde{\mathbf{m}} = \mathbf{C_m}^{-1/2}(\mathbf{m} - \mathbf{m_{prior}})$, and then obtain the actual (physical) model parameter as $\mathbf{m} = \mathbf{m}_{prior} + \mathbf{C_m}^{1/2}\tilde{\mathbf{m}}$. The starting model file has to be given in the transformed model space, i.e., what is output in the ***.prm file. For a simple restart of the inversion (picking up where the search has gone so far) you would use the name of the appropriate file from the last iteration completed (for example `Example_NLCG_060.prm`) as the `StartModel` (last optional argument) when restarting the inversion with the -I option. The same procedure would be used when restarting with a larger value of the damping parameter lambda.

Some other restarting cases, where either the model is edited, or the covariance parameters (including frozen sub-domains) involve an additional step. In these cases you have to first create the appropriate ***.prm file. We give two slightly different examples here; the basic idea is the same in both (and other cases). Suppose you wish to modify the inversion result, removing some structure to see if it is required. Let's call the modified model parameter file `HypTest.rho` (containing log resistivity, modified with some tool from the inversion result). As a first step you could obviously run the forward code, using `HypTest.rho` as the input model. To go further, you might want to restart the inversion from the modified model. This must first be transformed (roughened), using the -C inverse option:

```
ModEM -C INV HypoTest.rho HypTest.prm CovCtrl rFilePrior
```

Then restart the inversion:

```
ModEM -I NLCG rFilePrior rFile_Data InvCtrl FwdCtrl CovCtrl HypoTest.prm
```

Here `CovCtrl` is the covariance parameter file, and `rFilePrior` is the prior model file, which in the application just described would be the same as used for the initial inversion.

Going one step further in the hypothesis test example, one might want to remove a structure (e.g., a conductive feature) and then freeze the region where conductivity has been set to background levels (thus verifying whether data can be fit without the tested structure). To do this the covariance file would have to be modified to now include the frozen region. This modified covariance file would then be used in both steps given above (-C and -I).

As another example, one might want to change the covariance (e.g., changing smoothing length scales, etc.), and restart from results obtained with an earlier run to (hopefully) accelerate convergence. This would be done in the same way, except the input model parameter file on the first (-C) step would be something like `Example_NLCG_060.rho` — i.e., convert the final model parameter file to the transformed (***.prm) form appropriate to the new covariance, and then use this for the starting model.

6.2 Nested Modeling

Boundary conditions for the ModEM forward model are specified tangential fields on domain boundaries (top, bottom, and sides). By default, boundary data are computed from the prior model conductivity (at present using solutions to a series of 1D MT problems). When this simple default is used it is advisable to keep boundaries distant from the interior model domain of interest. ModEM also supports a *nesting* option, where boundary data are taken from a 3D forward solution in a larger (and typically more coarsely discretized) domain. This approach requires computing a forward solution on this larger domain, and saving the electric field solution vector in file `wFile_EMsoln` (first optional argument in `-F` option). The model parameter file for this large domain could be obtained from running the inversion, or from a prior model (for example containing a realistic ocean over an area too large to use for the final high resolution inversion run). Note that there is no option to output the electric field solution with the `-I` option, so if an inverse model from a large grid is used for nesting, the forward model must be rerun (on the large grid) using the final inversion result.

Use of the nested option, for either forward (`-F`) or inverse (`-I`) calculations is triggered by presence of the final (optional seventh) line in the `fwdCtrl` file (see section 5.1). The name of the electric field solution file (i.e., `wFile_EMsoln` used when running `-F` on the large grid) should be specified on this line to turn on the nesting option. NOTE: the periods (including order) in the data file used for the large grid `-F` run must be identical to those used for the nested forward/inverse modeling. The simplest approach is to use the same data file (`rFile_Data`) for both large and nested runs, but only the periods (not the sites) have to be the same.

Here is a simple example illustrating the use of nested modeling for both forward modeling (`-F` option) and inversion (`-I` option). Assume that you have constructed a large coarse grid which includes e.g., an ocean. Let us call the large coarse grid "Large_Grid.mod" and the data file "Data_file.dat". In the first run we need to let ModEM compute the electric field solution and save it in a file. Thus, in the first run we need the `-F` option with the optional output `wFile_EMsoln`

```
ModEM -F  Large_Grid.mod Data_file.dat temp_Predicted_Large.dat EMsoln_large.soln
```

After running this command line, ModEM will produce 2 files: 1) temp_Predicted_Large.dat, the predicted data for the large grid (not important), 2) EMsoln_large.soln, the electric field solution. To run the nested modeling using either forward (`-F`) or inverse (`-I`) options you need to first to modify the forward solver control file `rFile_fwdCtrl` to include the name of the electric field solution obtained from the first run:

```
Number of QMR iters per divergence correction : 40
```

```
Maximum number of divergence correction calls : 20
Maximum number of divergence correction iters : 100
Misfit tolerance for EM forward solver        : 1.0e-7
Misfit tolerance for EM adjoint solver        : 1.0e-7
Misfit tolerance for divergence correction    : 1.0e-5
Optional EM solution file name for nested BC  : EMsoln_large.soln
```

For the second run (either forward (`-F`) or inverse (`-I`)) construct a smaller and generally finer grid and call it e.g., "Small_grid.mod". To run forward modeling use:

```
ModEM -F Small_grid.mod Data_file.dat Predicted_small.dat EMsoln_small.soln FwdCtrl
```

and for the inversion:

```
ModEM -I NLCG Small_grid.mod Data_file.dat InvCtrl FwdCtrl CovCtrl
```

IT IS VERY IMPORTANT TO NOTICE HERE THAT IN THE FIRST AND SECOND RUNS WE USED THE SAME DATA FILE "DATA_FILE.DAT". THIS IS BECAUSE ModEM DOES NOT INTERPOLATE OVER PERIODS. THUS, TO MAKE SURE THAT THE SAME PERIOD LAYOUT SAVED IN "EMSOLN_LARGE.SOLN" IS USED IN THE SECOND RUN TO EXTRACT THE BOUNDARY VALUES AT THE EDGES OF THE NESTED GRID, USE THE SAME DATA FILE.

7 Change Log for Release 1.2.0, April 12, 2019

This release incorporates a substantial update to the code, in preparation for some additional capabilities related to boundary conditions. However, the updates that are of relevance to the user community are few.

7.1 Flexible Air Layers

Air layers can now be set in the forward solver configuration file. Options can be printed to screen using `./Mod3DMT -F` command without the additional input arguments. The new capability can be executed by adding two more lines to the configuration file (lines 8 and 9), after the nested modeling file. If you are not using the nested modeling option but would like to change the air layers, you should still include line 7, but leave the name of the EM solution file in line 7 blank. Flexible air layers are only implemented for 3D.

Three options for the air layers are: `mirror`, `fixed height`, `read from file`. For backwards compatibility, the default is '`mirror 10 3. 30.`', so that the code should still produce identically the same results to the previous version, unless the air layers are changed.

Option 1: mirror Mirrors the air layers based on the uppermost layers of the Earth resistivity model. In the code, air layers are counted from top to bottom (top is 1, bottom is N). The logic is as follows,

$$airlayer((N+1)-j) = \alpha^{j-1} * earthlayer(j), \tag{1}$$

where α and the number of air layers N are supplied by the user, and j is an integer index from 1 to N. When $j = 1$, the bottom air layer N equals the top earth layer 1. When $j = N$, the top air layer 1 equals $\alpha^{N-1} * earthlayer(N)$. Finally, the third user-supplied parameter `MinTopDz` specifies the minimum width of the top air layer, in km. If the top layer happens to be thinner than specified by this parameter, it is adjusted to this value. The parameters are specified in the following order: N, α, `MinTopDz`.

```
Air layers mirror|fixed height|read from file : mirror
Number of air layers and min top dz in km     : 10 3. 30.
```

This procedure is unnecessarily complex, and has created an unreasonably large air domain for most grid setups in the past. It is maintained now as the default for backwards compatibility. However, this option is not recommended.

Option 2: fixed height In this setup, the width of the air layers increases logarithmically until it reaches the user-defined height. Additionally, the user defines the total number of the air layers, and the logarithmic exponent is calculated automatically from these two parameters. This is the recommended option.

```
Air layers mirror|fixed height|read from file : fixed height
Number of air layers and max height in km     : 12 1000.
```

Option 3: read from file In this setup, the user specifies the total number of the air layers, and their width from top to bottom, in km, directly in the forward solver configuration file. This provides additional flexibility for advanced users, allowing in particular to compare modeling results more directly to those produced by alternative codes or analytical solutions.

```
Air layers mirror|fixed height|read from file : read from file
Number of air layers and dz top to bottom km  : 10 500. 200. 100. ... 2. 1. 0.5
```

Recommended air layers option is '`fixed height 12 1000`'. The users will likely find out that different air layer setups create substantial differences in the modeled impedances. We suggest that this issue is carefully explored by the users before they switch to a different setup.

7.2 LOG10 Resistivity Model File Option

To exercise this new option, create a WS or Mackie format model file with `LOG10` resistivity instead of `LOGE` or `LINEAR`. This is specified in the second line of the resistivity model input file (the first line is a comment). Internally in the code, we are keeping `LOGE` for all computations; but we convert `LOG10` or `LINEAR` to `LOGE`, make note of the user preference, then convert back for output. This feature is particularly useful for synthetic models, since it makes the input/output files more readable.

Having said that, we should note that we appreciate the inadequacy of an ASCII input file format for modern high performance 3D modeling situations. An option utilizing modern binary input/output formats will be added in a future release.

7.3 New Unit Testing Features for Code Developers

This release includes a collection of fully developed and debugged symmetry tests, gradient test, and the full Jacobian test. The code now passes all tests; all symmetry tests (including S, i.e., the forward solver) are passed with arbitrary random perturbations as input. The symmetry test `[TEST_ADJ]` has many sub-options, which can be explored by typing `./Mod3DMT -A` command without the additional input arguments. Options to generate random perturbation files for model [-A m], data [-A d], RHS [-A b], E-soln [-A e] are also included.

```
[TEST_GRAD]
-g  rFile_Model rFile_Data rFile_dModel [rFile_fwdCtrl rFile_EMrhs]
Runs the ultimate test of the gradient computation based on
Taylor series approximation.
[TEST_ADJ]
-A  J rFile_Model rFile_dModel rFile_Data [wFile_Model wFile_Data]
Tests the equality d^T J m = m^T J^T d for any model and data.
Optionally, outputs J m and J^T d.
[TEST_SENS]
-S  rFile_Model rFile_dModel rFile_Data wFile_Data [rFile_fwdCtrl]
Multiplies by the full Jacobian, row by row, to get d = J m.
```

`Compare to the output of [MULT_BY_J] to test [COMPUTE_J]`

7.4 Other Comments

Relative to the previous stable release, the value of `LARGE_REAL` that defines the error bars for modeled data has changed from 2.0e15 to 1.0e13. This does not affect any computations but it may be confusing to the user as these values are output in place of the error bars for modeled MT data. Otherwise, impedances are exactly the same unless the air layers are changed. With modified air layers values are noticeably different and probably more accurate... because now the diagonals are exactly zero over a 1D area. Tippers are less noticeably affected by the air layers modifications.

In addition to these user facing changes, this code has undergone substantial development, specifically to allow for flexibly boundary conditions in the future. We have also cleaned up the handling of non-zero sources in EMsolve3D.f90, specifically checked the physics and the maths, and added comments to clarify things.

8 Known Issues

[POSSIBLY NOT AN ISSUE] As of revision 485 on July 24, 2014 , apparent resistivities in 3D MT are specified in the file as actual resistivity, but are converted to the log domain before inversion in the program. There is no way presently to invert apparent resistivity itself, only logs. Probably OK, but obtuse.

[POSSIBLY OBSOLETE] Also, in the stable version that most users have there is a bug in error propagation for the apparent resistivities.