Software Documentation and User Manual of a Reduced Basis Occam (REBOCC) Inversion Version 1.0 for Two-dimensional Magnetotelluric Data

Weerachai Siripunvaraporn and Gary Egbert
(wsiripun@oce.orst.edu, egbert@oce.orst.edu)
College of Oceanic and Atmospheric Sciences
Oregon State University
Corvallis, 97331

January 29, 1999

Contents

Chapter 1

Software Documentation

1.1 Information about REBOCC

1.1.1 What is REBOCC ?

REBOCC is an inversion program for 2-D Magnetotelluric (MT) data. REBOCC stands for **RE**duced **B**asis **OCC**am's Inversion. It is based on an efficient variant on the OCCAM algorithm of deGroot-Hedlin and Constable (1990).

Currently, REBOCC can invert apparent resistivity (ρ_a) and phase (ϕ) of TM and TE modes, as well as the real ($\Re$) and imaginary ($\Im$) parts of the vertical magnetic transfer function (TP) -the ratio between the vertical magnetic field and the horizontal magnetic field.

1.1.2 Why uses REBOCC ?

- Fast
  Numerical experiments (Siripunvaraporn and Egbert, 1999) show that the speed of REBOCC is competitive with RRI (Smith and Booker, 1991), generally faster than NLCG (Rodi and Mackie, 1999), and significantly faster than OCCAM (deGroot-Hedlin and Constable, 1990).

- Stable
  REBOCC converges more reliably than some approximate inversions.

- Moderate memory requirement
  Large data sets can be inverted with REBOCC. It has been tested on a Sun UltraSparc I (with 288 Megabytes of RAM) on a very large problem: joint inversion of 55 stations, 41 periods and 2 responses for TM mode, 37 stations, 41 periods and 2 response for TE mode and 15 stations, 41 periods and 2 response for TP data (or total number of data = 8774) and total number of model parameter of 14800 (74×200).

- Easy to use
  Default options and keywords (provided for input parameters), generally, work well. We also provide a user manual. The program is written entirely in standard FORTRAN 77. Compilation with any standard Fortran compiler should be straightforward. To date REBOCC has been tested and run on Sun workstations running Solaris and IBMs runing AIX.

1.1.3 What is the condition on using REBOCC inversion?

REBOCC is freely available for academic use. It cannot be used for any commercial purposes without written permission from the authors.

1.1.4 Update and bug fixed ?

If you download the program, please send your name and email address to `wsiripun@oce.orst.edu` or `egbert@oce.orst.edu`, so that we can put your name into an update and bug report list.

1.2 Instruction

1.2.1 Downloading

The program can be downloaded from `ftp.oce.orst.edu` via anonymous ftp in `/pub/wsiripun/rebocc`, or point your browser to `ftp://ftp.oce.orst.edu/pub/wsiripun/rebocc` and click at the file to download.

- `rebocc.tar.gz` is for gzipped source.
- `rebocc.tar.Z` is for compressed source.

Note that in the case that you can not find the directory, please contact `wsiripun@oce.orst.edu` or `egbert@oce.orst.edu` directly for the new location.

1.2.2 Installing

After downloading the program, `gunzip rebocc.tar.gz`, or `uncompress rebocc.tar.Z` to get `rebocc.tar`. Then type `tar xf rebocc.tar` at the command line. You should get four directories.

- `data` contains examples of the data input files.
- `docs` contains documents, including this documentation and a preprint of Siripunvaraporn and Egbert (1999).
- `examples` contains examples of the startup files.
- `src` contains Fortran source codes.

1.2.3 Testing and Experimenting with REBOCC

We provide sample startup files which are ready to run in `/examples/startupfiles`. First, compile the program in directory `src`. If you find you do not have enough memory to run the inversion program with default parameter setting, see section 2.1 for details in adjusting parameters in the `parameter.h` file.

Then, make another directory, copy one of the startup files into this directory, and run the program. An example of the output directory is `/examples/tms6`.

Different startup files are provided for different circumstances. Notes on use of the startup files are in the header of each file. Experimenting with the examples along with reading the user manual provided in chapter 2 will make understanding the structure of the program simpler.

1.3 Overview of REBOCC

Here, we briefly review the inversion approach used in REBOCC. For more technical details, please refer to Siripunvaraporn and Egbert (1999).

The goal of the inversion is to find the minimum structure model subject to a desired misfit level. The unconstrained functional $U(\mathbf{m}, \lambda)$ can be written as

$$U(\mathbf{m}, \lambda) = (\mathbf{m} - \mathbf{m}_0)^T \mathbf{C_m}^{-1} (\mathbf{m} - \mathbf{m}_0) + \lambda^{-1} \{(\mathbf{d} - \mathbf{F}[\mathbf{m}])^T \mathbf{C_d}^{-1} (\mathbf{d} - \mathbf{F}[\mathbf{m}]) - X_*^2\} \tag{1.1}$$

for which stationary points (with respect to both $\mathbf{m}$ and λ) are sought.

Instead of solving the minimization problem in the model space, we transform the problem into the data space, by expressing the solution as a linear combination of "representers" (i.e., rows of the sensitivity matrix smoothed by the model covariance). This transformation reduces the size of the system of equations to be solved from $M \times M$ to $N \times N$ (where M is the number of model parameter and N is the number of data parameter). Since the number of model parameter M is often much larger than the number of data N, a significant decrease in both cpu time and memory can be achieved with this approach. More importantly, the data space formulation leads naturally to a simple approximation which can result in very significant computational savings.

Generally, MT data are smooth (in period, and for closely spaced sites, in space) and "redundant". Therefore, in the data space approach, there is no need to use all of the representers. A subset of these basis functions (of dimension L) is sufficient to construct the model without significantly loss of detail. With this approximation it is unnecessary to compute all sensitivities, and the size of the system of equations that must be solved can be significantly reduced (to $L \times L$ where $L << N$ and M).

Note that even though we construct the solution from a subset of the smoothed sensitivities, the goal of the inversion remains to find the norm minimizing model subject to fitting all of the data well enough.

With careful implementation of forward modeling and sensitivity calculations, run times of the REBOCC are only a fraction of most of the methods. In addition, by reducing the size of the system of equations and sensitivity matrix, large data sets can be inverted with REBOCC.

Because we are searching for the minimum norm model, the REBOCC inversion can be divided into two stages: Phase I for bringing down the misfit to the desired level, and Phase II for searching for the model with minimum norm while keeping the misfit at the desired level (or smoothing process). Phase II is necessary in order to wipe out the spurious features occurring while the program tries to reduce the misfit.

Since the MT inverse problem is non-linear, the desired misfit may never be reached and Phase II will never be executed. In this case we recommend user should restart the inversion process with a higher desired misfit, and a model (having a misfit close to the new desired misfit) from the previous run as a starting model.

1.4 Acknowledgments

We thank the Royal Thai Embassy via the Development and Promotion of Science and Technology (DPST), the US DOE and NSF for support. We also thank Markus Eisel for his helpful review of this user manual.

Chapter 2

User Manual

2.1 Configuring and Compiling

To minimize memory requirements we provide three different include files for use with REBOCC to invert only one ("parameter1.h"), two ("parameter2.h") or all three ("parameter3.h") data types. Note that one or two mode inversion are allowed with "parameter3.h". One of these include files has to be copied to "parameter.h" before compiling.

The file "parameter.h" sets the maximum dimension for a number of arrays used in REBOCC. Users will have to adjust the values of these parameters for the size of the inversion problem, with regard to the memory size of the computer.

- `NMODMX` is the maximum number of mode. If you copy the file from "parameter1.h", "parameter2.h" or "parameter3.h", you probably do not need to change this.
- Data dimension parameters to set:
  - `NRESiMX` is the maximum number of responses for the ith mode. The number of response should not be more than two for each mode (i.e., the ρ_a and ϕ for TM and TE, and $\Re$ and $\Im$ for the tipper).
  - `NPERiMX` is the maximum number of periods for the mode i.
  - `NSTAiMX` is the maximum number of stations for the mode i.

  These parameters are used to calculate `NNiMx` and `NNOMX` which should not be modified. texttttNNiMX is a maximum number of data for mode i calculated from the multiplication of `NRESiMX`, `NPERiMX` and `NSTAiMX`. `NNOMX` is the maximum number of data parameters, and is automatically calculated from the summation of the `NNiMX`.
- Representer dimension parameters to set: These parameters refer to the fraction of data used to define the reduced basis. An example of the subset of data selected is given in section (2.2.3). The default values are currently setting for using with $3rd$-stripe subset and up, and only need to be adjusted when having memory problems.
  - `LPERiMX` is the maximum number of periods used to calculate the representers for mode i. Generally, this value is a lot smaller than `NPERiMX` and can not exceed `NPERiMX`.

- `LSTAiMX` is the maximum number of stations used to calculate the representers for mode i. For stripe pattern (see section 2.2.3), this parameter is equal to `NSTAiMX`, and for checker patter (see section 2.2.3) this should be less.

These parameters are used to calculate `LLiMx` and `LLOMX` which should not be modified. `LLiMX` is calculated the same way as `NNiMX`, except using `LPERiMX` and `LSTAiMX`. `LLOMX` is the maximum number of representers automatically computed.

- Model dimension parameters to set:
  - `NZOMX` is the maximum number of rows (Z-levels) of the model, including air layers.
  - `NYOMX` is the maximum number of columns of the model.

  There should be NO adjustment to other parameters.

2.2 Input Files

Input information for the REBOCC inversion is split into three required input files and four optional files. The required files are ones that users must provide. If optional files are not provided, default values are used for the corresponding options.
These are the required files:

1. Startup file : This file defines all parameters used for the inversion.
2. Data file(s) : This data file contains the data, i.e. ρ_a and ϕ of TM and TE, or $\Re$ and $\Im$ of TP, used for inversion. Separate data file are required for each mode. Therefore, for example, when inverting TE and TM, users must provide two data files.
3. Starting model file : This file defines the model grid size and the initial resistivity value of each model block.

These are the optional files:

1. Sensitivity inclusion matrix file(s) : This file contains the data subset used to calculate the representer. Simple data subsets that work effectively for REBOCC will be given in section (2.2.3) and a keyword for the simple subsets can be used.
2. Distortion file(s) : This file contains the static shift indices and correction factors.
3. Prior model file : The prior model $\mathbf{m_0}$ is the model that the inversion seeks to minimize smooth deviations from. It can be the same file as the starting model. By default, there is no prior model.
4. Model control file : This file defines the free and fixed regions of the model such as the ocean, and allows for turning off smoothing across a known fault.

All input files are command driven with the following rules :

- Each line contains one command followed by parameter.
- Commands are separated from the parameters by space(s).

- Every command must be continuous. The underline "_" is used to distinguish words.
- Lines beginning with "#" are comment lines which will not be read.
- Blank lines can be used anywhere in the startup file.
- Commands in UPPER case are required, i.e., users must provide these parameters (no default values).
- Commands in lower case are the optional, i.e., users can either set values or leave them as defaults.
- Optional command lines can be omitted. Program will then use default values. In most cases, leaving parameters set to defaults should work just fine.

2.2.1 Startup File

Example of Startup File

```
#### example of long startup file ####
#  data
NUMBER_OF_MODE 1

# data input section : first mode
DATA_FILE      ../../data/data.tm
SENS_INCLUSION stripe:p=6
LEFT_OFFSET    295000.
distort_file   default

# output section
OUTPUT_FILE    tms6
initial_iterno default
logfile_screen default

# starting model section
STARTING_MODEL ../../data/INITMODEL.H100
prior_model    default
background_rho default
model_control  default
fwd_only       default

# inversion parameters section
DESIRED_RMS    1.
MAX_ITERATION  20
MAX_SMOOTHING  10
sd_rms         default
cont_high_rms  default
cont_notfound  default
cont_highnorm  default

# lagrange multiplier (lgm) control
starting_lgm   default
stepsize_lgm   default
smooth_szlgm   default
search_lgm     default
max_search_lgm default

# PCG forward modeling options
etol           default
max_pcg_iter   default

# Model Improvement
model_change   default
mnorm_change   default
parabolic_cor  default

#### example of short startup file ####
#### leaving optional commands to defaut ####
NUMBER_OF_MODE 1
DATA_FILE      ../../data/data.tm
SENS_INCLUSION stripe:p=6
LEFT_OFFSET    295000.
OUTPUT_FILE    tms6
STARTING_MODEL ../../data/INITMODEL.H100
DESIRED_RMS    1.
MAX_ITERATION  20
MAX_SMOOTHING  10
```

Descriptions of Startup File

The startup file is subdivided into seven sections.

1. Data Input Section:

- `NUMBER_OF_MODE` *integer* [1 <= `NUMBER_OF_MODE` <= 3]
  This must be the first command in the startup file, otherwise the program will terminate. It specifies the number of modes to be inverted, and must be less than 3. Then a sequence of commands (`DATA_FILE`, `SENS_INCLUSION`, `LEFT_OFFSET` and `distort_file`) for each mode are placed following the `NUMBER_OF_MODE` command. Here is an example (only for the data input section) of two mode inversion.

```
#  data
NUMBER_OF_MODE 2

#  first mode
DATA_FILE      ../../data/data.tm
SENS_INCLUSION stripe:p=6
LEFT_OFFSET    295000.
distort_file   default

#  second mode
DATA_FILE      ../../data/data.te
SENS_INCLUSION stripe:p=6
LEFT_OFFSET    295000.
distort_file   default
```

- `DATA_FILE` *filename* [less than 70 characters]
  This gives the name of the data file to be inverted. A full description of the data file format is given in section 2.2.2.
- `SENS_INCLUSION` *filename* [less than 70 characters]
  This gives the name of the sensitivity inclusion file used to determine which representers are calculated. A full description of this file is given in section 2.2.3. For simple subsets, keywords can be used instead of providing a file. A keyword *stripe:p=6* is used to represent the $6th$-stripe subset of the data or *checker:p=4:s=2* for the $4th$:$2nd$-checker subset of the data. The program will then automatically generate the file (see section 2.2.3).
- `LEFT_OFFSET` *real*
  This is the distance from the left boundary of the starting model to the origin of the coordinate system used for specifying station locations (see more details on `NUMBER_OF_STATION` of section 2.2.2). Generally, we prefer to have stations located at the boundaries of the blocks of the model.
- `distort_file` *default/filename* [less than 70 characters]
  The *default* value of this command is no correction for static distortion, otherwise *filename* must be given if the correction is required. Details of the distortion file is discussed in the section 2.2.6.

2. Output Section:

- `OUTPUT_FILE` *filename* [less than 70 characters]
  This command specifies the name of the output file. Output files are generated for each iteration, with the number of the iteration added to the filename (e.g., tms6_model.001 is the output after the 1st iteration). The model output file can also be used directly as the starting model. Details of the output file are given in section 2.3.
- `initial_iterno` *default*[0]/*integer*
  This is the initial iteration number. By default, it is set to zero. In special circumstances (e.g., when restarting the inversion with the results from a previous run as starting model) one might want to set this to a different value. Setting this parameter to a non-zero value has no effect on the number of iterations (`MAX_ITERATION`) set later.

- `logfile_screen` *default*[*yes*]/*no*
  This command controls where the program writes the logfile. By *default* (*yes*) the log file is written to both screen and file. If set to *no*, the program will write the log file to the file only. This option is useful when running the program in the background.
- `fwd_only` *default*[*no*]/*yes*
  This command specifies which problems will be solved: the forward problem or the inverse problem. By *default* (*no*), the inversion problem is solved. If set to *yes*, only the forward problem (for the initial model) will be solved.

3. Starting Model Section:

   - `STARTING_MODEL` *filename* [less than 70 characters]
     This command specifies the starting model file. Description is given in section 2.2.4.
   - `prior_model` *default*/*filename* [less than 70 characters]
     The prior model $\mathbf{m_0}$ is the model that the program will invert around. This can be used to include known geological features in the inversion. By *default* there is no prior model (i.e., $log_{10}\rho = 0$). The program uses the grid defined by the starting model. The prior model should be on the same grid. See futher details below.
   - `background_rho` *real*
     This is the estimated background resistivity value. By *default* the program chooses the average resistivity of the first layer of the starting model as the background resistivity value. The program uses skin depths for this resistivity to set length scales for the interpolation scheme.
   - `model_control` *default*/*filename*
     This command specifies the model control file. This file contains prior information such as freezing part of the model (e.g., the ocean), or adding faults. More details of the control file format are given in section 2.2.7. The *default* is no prior information about any parts of the model.

4. Inversion Parameter Section:

   - `DESIRED_RMS` *real*
     This is the desired root mean square (RMS) misfit to the data which is defined as $||\mathbf{C_d}^{-1}(\mathbf{d} - \mathbf{F}[\mathbf{m}])||$. The goal of the inversion is to find the minimum norm model subject to this RMS. After completing Phase I (reaching this RMS), the inversion starts Phase II by keeping the misfit at this desired level but reduceing the norm of the model $R = ||(\mathbf{m} - \mathbf{m_0})^T \mathbf{C_m}^{-1}(\mathbf{m} - \mathbf{m_0})||$.
   - `sd_rms` *default*[0.05]/*real*
     This is the tolerance level of the desired RMS, i.e. it expands the range of the desired RMS from `DESIRED_RMS` - `sd_rms` to `DESIRED_RMS` + `sd_rms`.
   - `MAX_ITERATION` *integer*
     This is the maximum number of iterations for Phase I of the inversion. The program will stop if the number of iterations exceeds this value, even though the desired RMS level has not been reached.
   - `cont_high_rms` *default*[*yes*]/*no*
     Setting this parameter to *no* forces the inversion (in Phase I) to stop if the RMS does not improve from the previous iteration. The *default* (yes) is to keep going even when this happens.

- `MAX_SMOOTHING` *integer*
  This is the maximum number of smoothing iterations in Phase II of the inversion after reaching the desired RMS.
- `cont_notfound` *default*[*no*]/*yes*
  By *default*[*no*] in Phase II if the inversion cannot reach the desired RMS, the inversion will stop and the results of the previous iteration should be used. Setting to *yes* will continue the inversion.
- `cont_highnorm` *default*[*no*]/ *yes*
  By *default*[*no*] the (Phase II) inversion process stops if the model norm cannot be improved from the previous iteration. If set to *yes*, the inversion process will continue even the model norm is not improved.

5. Lagrange Multiplier (λ) Section: λ acts to "trade-off" between minimizing the norm of the data misfit and the norm of the model (see 1.1). When λ is large, the data misfit is de-emphasized, leading to a smoother model. In contrast, as $\lambda \to 0$ the inverse problem becomes closer to the ill conditioned least-square inversion problem, resulting in an erratic model (see Parker, 1980). For more information, please refer to Siripunvaraporn and Egbert (1999).

   - `starting_lgm` *default*[2.]/*real*
     This is the starting log10 λ used in the first iteration only. Subsequent iterations will use information from previous iterations. The *default* is 2.
   - `stepsize_lgm` *default*[0.5]/*real*
     This is the step size used for searching for λ (never changed inside the inversion) in Phase I of the inversion. The *default* value is 0.5 covering one full decade with three values of log10 λ. The linearized search for λ works as follows. In the first iteration, the inversion will start by calculating the misfits of the models from the given λ and nearby λ ($\pm$ the step size). If the minimum misfit is not found in this range, the program will move to the left or right (depending on the search direction) until these three values bracket the minimum misfit. Then the "parabolic interpolation" (see Press et al., 1992) is used to find the minimum misfit bracketing by those three points. If the minimum misfit from parabolic interpolation seems to be less than the misfits of the three λ, the program will keep this value and start the next iteration, otherwise the program will use the λ of the center value and go to the next iteration. Setting the step size value to a smaller number will increase the number of λ used for bracketing the minimum especially in the early iterations.
   - `smooth_szlgm` *default*[0.1]/*real*
     Similar to `stepsize_lgm`, this is the step size used for searching for λ, but in Phase II of the inversion. The *default* value is 0.1. A smaller step size is preferable in this case.
   - `search_lgm` *default*[*yes*]/*no*
     Setting this parameter to *no* swithches off the λ search process described above. λ is fixed at the starting value. The inversion problem become minimizing the penalty function with λ fixed (see Siripunvaraporn and Egbert (1999) for more details). If the misfit is reduced at every iteration, the bracketing for minimum process will not be applied. This could help speed up the inversion because only one λ is used in every iteration. However, without any priori information, we do

not really know the optimal value of λ. Different values of λ can converge to different level of misfit minimum. Therefore, we do **not recommend** this strategy.

- `max_search_lgm` *default*[10]/*integer*
  This is the maximum number of steps used to search for λ per iteration. The value will be ignored if fixed λ is used. If λ is allowed to vary, we found that up to eight values of λ are used in the early iterations, and at least three λ are used for later iteration. `max_search_lgm` might have to be increased if `stepsize_lgm` is set to a smaller value.

6. PCG Forward Modeling Section: These options control iterative solution of the forward modeling problems.

   - `etol` *default*[1.E-08]/ *real*
     This is the tolerance level of the normalized relative residual, defined as `etol` $= ||\frac{\mathbf{b}-\mathbf{Ax}}{\mathbf{b}}||$, from solving the system of equation $\mathbf{Ax} = \mathbf{b}$ of the MT forward problem with the Preconditioned Conjugate Gradient (PCG) method. For general problems the *default* (etol = 1.E-08) value yields the results with a deviation of less than 2 %, from those of the finite element forward modeling program of Wanamaker (1986). Decreasing this value will increase the accuracy of the forward modeling, but at the cost of longer computational time. Note that the accuracy of forward modeling depends also on other factors, such as the vertical grid space near the surface, etc.
   - `max_pcg_iter` *default*[500]/ *integer*
     This is the maximum number of PCG iterations. If forward modeling needs more than the *default* (500 iterations) setting, this could mean the system of equation is really stiff and difficult to solve. A stiff system of equations is generally the result of a small λ, leading to a problem close to the ill-condition least square problem requiring a high number of iteration. The resulting model (if PCG converges) is usually very rough with a relatively high RMS. Therefore this kind of model should be avoid. If the `max_pcg_iter` is reached for a given λ, the program will automatically move to a higher value of λ where a smoother model can be expected.

7. Model Improvement Section:

   - `model_change` *default*[1.]/*real*
     This is the minimum normalized relative percent change of the model for two successive iteration defined as $||\frac{\mathbf{m}_{k+1}-\mathbf{m}_k}{\mathbf{m}_k}|| * 100$. By *default*, the inversion will stop if if the relative change of the model is less than 1 %.
   - `mnorm_change` *default*[1.]/*real*
     This is the minimum normalized relative percent change of the model norm for two successive iteration defined as $||\frac{R_{k+1}-R_k}{R_k}|| * 100$ (only computed in Phase II). By *default*, the inversion will stop if the relative change of the roughness is less than 1 %.
   - `parabolic_cor` *default*[5.]/*real*
     This is the threshold improvement of the misfit after using parabolic interpolation to estimate the lowest misfit inside the bracket. If the misfit improves by more than `parabolic_cor` % (*default* is 5 %), the inversion will continue searching for an optimal λ. Otherwise, the inversion will accept the current best estimate of λ and begin for the next iteration.

2.2.2 Data File

The data files contain the data to be inverted. For each mode, a separate data file is required. The data is divided into three sections: a header section, a data section and a data inclusion section.

Example of Data File

```
TITLE                TEST_MODEL/TM_Mode/2_Percent_Noise
MODE_TYPE            tm
NUMBER_OF_RESPONSE   2
NUMBER_OF_PERIOD     31
     1.0000   1.2589    ...  794.3282 1000.0000
NUMBER_OF_STATION    36
      0.       3000.    ...    102000.     105000.
DATA_RESPONSE_NO_1   app
    1.0000   97.4844   98.3462   ... ...   97.2021  96.9847
       ...      ...      ...     ... ...      ...     ...
 1000.0000  100.2887  100.7545   ... ...  101.2673 101.2754
ERROR_RESPONSE_NO_1   0.02
    1.0000    0.1524    0.9892   ... ...    0.1009   0.3363
       ...       ...       ...   ... ...       ...      ...
 1000.0000    1.1713    0.3945   ... ...    0.8473   0.3754
DATA_RESPONSE_NO_2   phsdeg
    1.0000   45.8557   46.2199   ... ...   45.7885  45.6826
       ...      ...      ...     ... ...      ...     ...
 1000.0000   66.5211   67.5786   ... ...   67.8880  67.5644
ERROR_RESPONSE_NO_2   0.02
    1.0000    0.0717    0.4649   ... ...    0.0475   0.1584
       ...       ...       ...   ... ...       ...      ...
 1000.0000    0.7769    0.2646   ... ...    0.5680   0.2504
DATA_INCLUSION_NO_1 all
DATA_INCLUSION_NO_2 all
```

Descriptions of Data File

1. Header Section:

   - `TITLE` *string* [less than 70 characters]
     Specify the title of the data file.
   - `MODE_TYPE` *tm/te/tp*
     Enter the mode type of this data file. Currently, only three data types are allowed: TM, TE and TP (for Tipper). Upper or lower case are allowed.
   - `NUMBER_OF_RESPONSE` *integer* [1 <= `NUMBER_OF_RESPONSE` <= 2]
     This is the number of responses for this mode. For TM or TE, the responses should be ρ_a or ϕ while for TP the responses are the $\Re$ or $\Im$. This would be set to 1 only if one part of the response (e.g., ρ or ϕ) were available.
   - `NUMBER_OF_FREQUENCY/NUMBER_OF_PERIOD` *integer*
     This is the number of frequencies or periods, which is followed by a "free format" of frequency (in Hz) or period (in sec.) array in the next line (no blank or comment lines are allowed in between). Use `NUMBER_OF_FREQUENCY` to specify frequency, and, `NUMBER_OF_PERIOD` to specify period. `NUMBER_OF_FREQUENCY` or `NUMBER_OF_PERIOD` should not exceed `NPERiMX` in the "parameter.h" file. With the current version, two responses (in same mode) must have the same set of period or frequency. However, different sets of period or frequency can be used for different modes.
   - `NUMBER_OF_STATION` *integer*
     This is the number of stations which is followed by a set of site locations (in meters).
     `NUMBER_OF_STATION` should not exceed `NSTAiMX` in the "parameter.h" file. For the current version, two responses must have the same station locations. The program will use the left boudary offset (`LEFT_OFFSET`) specified in "startupfile" along with the first station location (readed from here)

to compute the actual location of the first station on the starting model grid. For example, if LEFT_OFFSET is set at x and first station location is at y, then the actual position of the 1st station from the left boundary of the model is at $x + y$.

2. Data Section: Before entering this section, NUMBER_OF_RESPONSE, NUMBER_OF_FREQUENCY or NUMBER_OF_PERIOD, and NUMBER_OF_STATION must be given first.

   - DATA_RESPONSE_NO_j/DATA_RESPONSE_NO_j* *app/applog/phs/phsrad/rel/img*
     Specify the type of response (by keywords) and response number (j). Following this command is the matrix of data (no blank line or comment line in between). There are two types of data matrix. For the default (DATA_RESPONSE_NO_j) the data matrix has NUMBER_OF_PERIOD rows, and NUMBER_OF_STATION + 1 columns. The first column of this data matrix must be the period or frequency followed by the data for that period from the first station to last station. This is repeated for each period or frequency. The second (DATA_RESPONSE_NO_j*) uses a matrix size NUMBER_OF_PERIOD rows, and NUMBER_OF_STATION columns, i.e. the period or frequency column is omitted. To use this format, the command DATA_RESPONSE_NO_j* is used instead of DATA_RESPONSE_NO_j, i.e. add '*' after the response number. These are the keywords for the allowed data types.
     - *app* is the apparent resistivity in $\Omega - m$ for TM and TE mode.
     - *applog* is the log10 of the apparent resistivity for TM and TE mode.
     - *phs* or *phsdeg* is the phase in degree for TM and TE mode.
     - *phsrad* is phase in radian for TM and TE mode.
     - *rel* is the real part of TP.
     - *img* is the imaginary part of TP.
   - ERROR_RESPONSE_NO_j/ERROR_RESPONSE_NO_j* *real*
     Specifies the error response number (j) and the error floor (e_f) and is followed by a matrix of error bars using the same format as for th data (see DATA_RESPONSE_NO_j option). Note that error response number (j) of the same response type must match the response number specified in DATA_RESPONSE_NO_j. Error floors are the minimum error that will be used (instead of the actual statistical errors) if the actual errors are smaller. For TM and TE, error floor value entered here is the relative error floor, while for TP, the error floor value is an absolute error floor. Here is how we translate the relative error floor of TM and TE to the absolute error floor.
     - The absolute error floor of the ρ_a is calculated from $e_f * \rho_a$
     - The absolute error floor of the $\log(\rho_a)$ is taken to be log10(1.+ e_f), as this corresponds to the specified level of relative error in ρ_a.
     - The absolute error floor of the phase ϕ is taken to be $e_f * \frac{90}{\pi}$ degree, as this corresponds to that level of relative error in the ρ_a.

3. Data Inclusion Section: The data inclusion matrix is an index matrix. Each integer correponds to one data element (a particular period and site). The purpose of having data inclusion is to avoid using bad data in the inversion. This inclusion matrix concept is the same as that used in RRI.

   - DATA_INCLUSION_NO_j *all/index*
     The error response number (j) is used to match the inclusion matrix with that of the data

response. The command is followed by two keywords: *all* used when no data are to be excluded, and *index* used when there is some change to the error level or omission of data. If *index* is used, it must be followed by an integer matrix of `NUMBER_OF_PERIOD` rows and `NUMBER_OF_STATION` columns . In the inclusion matrix, no spaces are allowed between integers, no blank lines, and no lines can be longer than 256 characters. These are the integer codes to be used in the inclusion matrix:

- 0 : exclude this data element in the inversion,
- 1 : include this data element in the inversion,
- 2-9 : include this data element in the inversion but increase the error by a factor of $2^{n-1}, n = 2, \cdots, 9$.

2.2.3 Sensitivity Inclusion File

The success of the data subspace approach depends on the selection of data points to determine the basis function. Optimal choice of data subsets is an issue that needs further study. Here we offer some simple suggestions based on our experience.

1. select the basis to uniformly cover the full data set, so that the simple linear interpolation scheme used here is effective.

2. Two examples of possible data subsets which generally seem to work well are shown.

   - "p*th*-stripe" where every p*th* period is chosen for all sites. This pattern is safe to apply in almost all cases because for physically consistency the data must be a smooth function of period. Selecting at least two periods per decade is our recommendation for this stripe pattern. A keyword *stripe:p*=`k` can be used (in the startup file) to represent the stripe pattern where `k` is the integer. The program will automatically generate the sensitivity file and output it with ".six_" as a suffix.
   - "p*th*:s*th*-checker" where every p*th* period and every s*th* station are selected to build the basis. This pattern is different from the stripe pattern in that the gaps in sites and periods are interlaced in a checkerboard fashion. This pattern is probably reasonable only in the case where the site spacing is small. Because of the way we compute sensitivities (by factorization of the matrix), it is generally most efficient to use as many sites as possible (i.e. small s*th*). This pattern is most useful to reduce storage requirements for very large data sets. A keyword *checker:p*=`k`*:s*=`k` can be used (in the startup file) to represent the checker pattern without forming a file.

Example of Sensitivity Inclusion Matrix Files

```
# example of 3rd-stripe
SENS_INCLUSION_NO_1 index
11111111111111111111111111111111
00000000000000000000000000000000
00000000000000000000000000000000
11111111111111111111111111111111
...
11111111111111111111111111111111
00000000000000000000000000000000
00000000000000000000000000000000
11111111111111111111111111111111
SENS_INCLUSION_NO_2 index
11111111111111111111111111111111
00000000000000000000000000000000
00000000000000000000000000000000
11111111111111111111111111111111
...
11111111111111111111111111111111
00000000000000000000000000000000
00000000000000000000000000000000
11111111111111111111111111111111
```

```
# example of 3rd:2nd-checker
SENS_INCLUSION_NO_1 index
10101010101010101010101010101
00000000000000000000000000000
00000000000000000000000000000
01010101010101010101010101010
...
01010101010101010101010101010
00000000000000000000000000000
00000000000000000000000000000
10101010101010101010101010101
SENS_INCLUSION_NO_2 index
10101010101010101010101010101
00000000000000000000000000000
00000000000000000000000000000
01010101010101010101010101010
...
01010101010101010101010101010
00000000000000000000000000000
00000000000000000000000000000
10101010101010101010101010101
```

Descriptions of Sensitivity Inclusion Matrix File

If using the keywords (see above) in the startup file, there is no need to provide the sensitivity inclusion file.

The format of the sensitivity inclusion matrix file is the same as the format of the data inclusion matrix in section 2.2.2. However, replacing command `DATA_INCLUSION_NO_j` with `SENS_INCLUSION_NO_j`, and only integer 0 and 1 are allowed.

- *1* : including this data element to form the representer.
- *0* : not using this data element to form the representer.

2.2.4 Starting Model File

Example of Starting Model

```
TITLE  100_Ohm-m_half_space
NY  100
80000. 70000.  ...  1500.  1500.  1500.  ...  70000. 80000.
NZB 31
10.    30.    60.   ...       80000.
nza 10
10.    30.   100.   ...       300000.
RESISTIVITY_MODEL  HalfSpace
100.
```

Description of Starting Model File

- `TITLE` *string* [less than 70 characters]
  Specify the title of the model.

- `NY` *integer*
  Specifies the number of horizontal blocks, and is followed by an array of horizontal grid spacings (in meter). `NY` should not exceed `NYOMX`, set in "parameter.h" file.

- `NZB` *integer*
  Specifies the number of (body) vertical blocks (excluding air layers), and is followed by an array of vertical grid spacings (in meter) from the surface down.

- `nza` *integer*
  Specifies the number of air layers, and is followed by an array of vertical grid spacings (in meter) from the surface up. `NZB+nza` should not exceed `NZOMX`, set in "parameter.h" file. Note that `nza` is an option. If omitting by user, a program will generate 10 air layers (grid spacing of 10., 30., 100., 300., 1000., 3000., 10000., 30000., 100000., 300000.) when need (TE and TP cases).

- `RESISTIVITY_MODEL` *HalfSpace/Layer/Index*
  - *HalfSpace* keyword indicates that this is a half space model, and is followed by the half space resistivity value on the next line.
  - *Layer* keyword indicates that this is a 1-D layered model, and is followed by an array of layered resistivity values from top to bottom (total of `NZB` layers).
  - *Index* keyword describes a 2-D resistivity model by an integer code. The maximum number of resistivities is currently restricted to 9 (corresponding to integer 1 to 9). If you need to use more than 9 resistivity values, consider the next option. The model is given by an integer matrix of

size NZB rows and NY columns, and is followed by an array of resistivity values specified by the integer code. There should be no space between integers and each line should not exceed 256 integers.

– If no keywords following the command, it requires an arbitrary 2-D model (NZB rows by NY columns) of real resistivity values.

```
#examples of models
RESISTIVITY_MODEL HalfSpace      RESISTIVITY_MODEL Layer      RESISTIVITY_MODEL Index        RESISTIVITY_MODEL
100.                             100. 100. ... 10. ... 1.     11111111111111111111           100. 100. 100. 100. 100. 100. 100. 100. 100.
                                                              11111111111111111111           100. 100. 100. 100. 100. 100. 100. 100. 100.
                                                                             ...                                    ...
                                                              22222244444444222222            10.  10.  10.  30.  30.  30.  10.  10.  10.
                                                              33333333333333333333            10.  10.  10.  10.  10.  10.  10.  10.  10.
                                                              100. 10. 1. 1000.
```

2.2.5 Prior Model File

The format of the prior model is the same as the starting model. For the prior model, we require NY, NZB and also the grid spacings to be the same as those of the starting model. Therefore, those commands become options for the prior model. The only required command is the RESISTIVITY_MODEL with the same keywords as the starting model.

2.2.6 Static Distortion File

The static distortion file is optional. This file is only required if static distortion needs to be corrected. One file is required for each data mode.

Example of Static Distortion File

```
DISTORTION_INDEX
11111111111111111111111111111111111111
DISTORTION_PARAMETER
0.1 0.2 0.3 1. 1. -0.5 ...  0.5 -0.4 -0.8 0.0 0.3
DISTORTION_INCLUSION all
```

Descriptions of Static Distortion File

- DISTORTION_INDEX
  Following this command in the next line are the distortion indices (one for each station), with no spaces, should not exceed 256 integers.
  - index 1 : correct for static distortions automatically (by program).
  - index 0 : fix the static distortions with the distortion parameters provided.
- DISTORTION_PARAMETER
  Following this command in the next line is an array of distortion parameters (one for each station) on a log10 scale. Positive parameters shift up, negative parameters shift down, and zero implies no correction. If the distortion index of a particular station is turned off (= 0), the value will never change. But if it turned on (=1), the parameter will change from iteration to iteration.
- DISTORTION_INCLUSION *all/index*
  This inclusion matrix is similar to the data (section 2.2.2) and sensitivity (section 2.2.3) inclusion matrix. Only 1 and 0 are allowed, no other integers used. This inclusion matrix defines which part of the data is used for estimating the distortion factors. For example, one might want to use only high frequency data to estimate the static distortion parameters. On the other hand, one can exclude bad data when calculating the static distortion parameters.

2.2.7 Model Control File

This is another optional file used to included prior knowledge of the model such as a known fault or the ocean, and to specify the model covariance.

Example of Model Control File

```
time_step     default
hor_length    default
ver_length    default
model_control index
00000011111111111111111111111111111111111111111
00000011111111111111111111111111111111111111111
...
11111111111111111111231111111111111111111111111
11111111111111111111231111111111111111111111111
...
11111111111111111111111111111111111111111111111
```

Description of Model Control File

- `time_step` *default*[1]/*integer*
  This is the time step used for smoothing with the model covariance. The *default* value is set to one step. Small numbers decreases computational time. Note that when adding prior knowledge into the inversion, this value should be increased to a greater number (5-10 is recommended), otherwise, undesirable steps or jumps in the model will result. Therefore, the program will automatically increase the `time_step` value to 10 (if prior knowledge is included).

- `hor_length` *default*/*real*
  This is the horizontal smoothing length scale for each model block used to control the smoothness in the horizontal direction. A higher length scale leads to a very smooth model which sometimes makes it very difficult for the inversion to fit the data. With smaller smoothing length scales static shifts can be confused with deeper model structure. Therefore, the *default* assigns value of each model block based on the maximum distance between stations and the depths of that blocks. If the *default* is not used, a constant horizontal length scale will be used throughout the model.

- `ver_length` *default*/*real*
  This is the vertical length scale of each model block used to control the smoothness in the vertical direction. The *default* value is the depth of that block. If the *default* is used in both horizontal and vertical direction, this will result in a 1:1 ratio in the deeper part, and larger horizontal length scale in the near surface. If the *default* is not used, a constant vertical length scale will be used throughout the model.

- `model_control` *default*/*index* If the *default* keyword is used, there is no model control. If the *index* keyword is used, it must followed by an integer matrix (`NZB` rows by `NY` columns). Its matrix format is the same as the integer matrix of the starting model (see 2.2.4). Each line cannot exceed 256 integers, and only positive single digits are used. These are the integer used in the model control matrix and their meaning.

  - 1 : for normal condition i.e. no priori knowledge.
  - 0 : to freeze the model at that prior model (require prior model file)
  - 2 : to have a vertical discontinuity on the right.

- 3 : to have a vertical discontinuity on the left.
- 4 : to have a horizontal discontinuity above.
- 5 : to have a horizontal discontinuity below.
- 6 : to have a vertical and horizontal discontinuity on the right and above.
- 7 : to have a vertical and horizontal discontinuity on the left and above.
- 8 : to have a vertical and horizontal discontinuity on the right and below.
- 9 : to have a vertical and horizontal discontinuity on the left and below.

```
#Example of  vertical fault
1111111111111111111111111
1111111112311111111111111
1111111112311111111111111
1111111112311111111111111
1111111112311111111111111
1111111112311111111111111
1111111112311111111111111
1111111112311111111111111
1111111112311111111111111
1111111112311111111111111
```

```
#Example of horizontal fault
111111111111111111111111111111
111111111111111111111111111111
111111555555555555555555551111
111111444444444444444444441111
111111111111111111111111111111
111111111111111111111111111111
111111111111111111111111111111
111111111111111111111111111111
111111111111111111111111111111
111111111111111111111111111111
```

```
#Examples of both horizontal and vertical fault
1111111111111111111111111
1111111112311231111111111
1111111112311231111111111
1111111112311231111111111
1111111112311231111111111
1111111112311231111111111
1111111112311231111111111
1111111112311231111111111
1111111112311231111111111
1111111112311231111111111
```

```
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
1111111111111111111111111111
```

2.3 Output Files

REBOCC generates output files at the start of the inversion, and after each iteration. File names use `OUTPUT_FILE` specified in the startup file as a prefix. At the end of the inversion, a log file is generated with suffix '.log'.

Once started, the program outputs many files: the data files, the error files, the data inclusion files and the sensitivity inclusion files. All of these output files are useful in order to check whether your data inputs are correct or not.

- The data files (with suffix '.data_tm', '.data_te' or '.data_tp' depending on data mode, one file for one data type) has the same format as the data input file (see section 2.2.2). However, it only contains the `DATA_RESPONSE_NO_j` read from the data input file.
- Similarly, the error files (with suffix such as '.error_tm') has the same format as the data files but contains only the `ERROR_RESPONSE_NO_j`. This `ERROR_RESPONSE_NO_j` is not the same as that read from the input data file, but rather represents the absolute errors after applying the error floors.
- Data inclusion files (with suffix such as '.dix_tm') contain the data inclusion files read from the data input files.
- Sensitivity inclusion files (with suffix such as '.six_tm') contains the sensitivity inclusion files read from the sensitivity inclusion file. Its format is the same as of the sensitivity inclusion file (see section 2.2.3).

After each iteration, REBOCC generates four types of output with iteration as the suffix: model files, response files, the static distortion files and the RMS files.

- The model file (with suffix such as '_model.000') contains the inverted model generated after each iteration. Its format is the same as the starting model file (see section 2.2.4), so that one can use this output model file directly as a starting model or a prior model.

- The response file (with suffix such as '_resp_tm.000', '_resp_te.000' or '_resp_tp.000'; one file for one data type) contains the data responses generated from the inverted model. Its format is the same as the data output file (e.g., '.data_tm'; see above).
- The static distortion file (with suffix '_dist_tm.000' or '_dist_te.000') contains the static distortion indices, parameters and static distortion inclusion matrix. Its format is the same as the static distortion file (see section 2.2.6), so that user can use this file directly as the static distortion input file.
- The RMS file (with suffix '_rms.000') contains the RMS misfits of all responses (no separate file for each data type). It shows the overall RMS, RMS of each responses by sites and by periods.