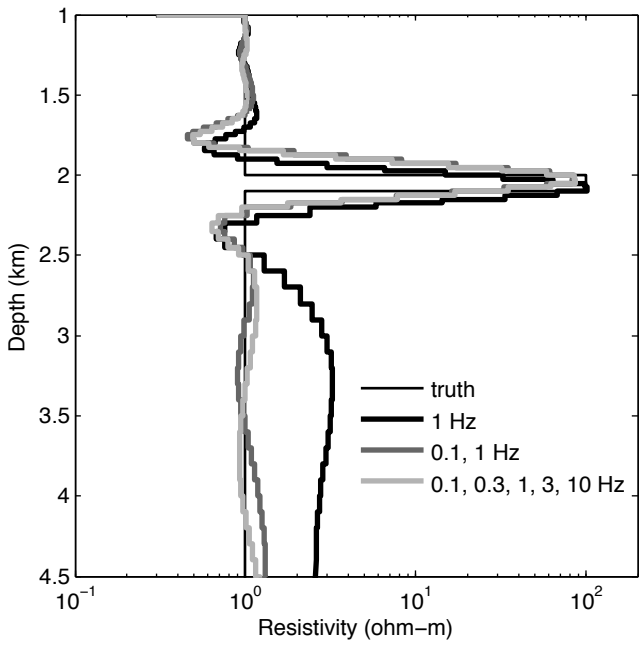
OCCAM1DCSEM:

An Open-Source Inversion Program for Generating Smooth 1D Models from Controlled-Source Electromagnetic and Magnetotelluric Data.



Documentation by:
Kerry Key
Scripps Institution of Oceanography
University of California, San Diego

A product of the Seafloor Electromagnetic Methods Consortium
http://marineemlab.ucsd.edu/semc.html

Revision 1.4, March 26, 2010.

Contents

List of Figures

List of Tables

1 About

This document provides instructions for using the OCCAM1DCSEM inversion code and the forward modeling code DIPOLE1D. OCCAM1DCSEM is a Fortran package for generating smooth one-dimensional models from controlled-source electromagnetic and magnetotelluric data. The OCCAM1DCSEM package is built around a new 1D CSEM forward code named DIPOLE1D, which has been designed to handle a diverse number of 1D CSEM modeling scenarios. This package also comes with Matlab front- and rear-end routines for editing survey data, creating the files required by OCCAM1DCSEM and for plotting the resulting inversion models and CSEM responses.

Here's a list of features currently supported in OCCAM1DCSEM and DIPOLE1D:

- Uses a Cartesian coordinate system, so you don't need to worry about converting to cylindrical coordinates any more
- Models all 3 field components (x,y,z) of both the electric and magnetic fields
- Supports inversion of real and imaginary, amplitude and phase, and polarization ellipse data.
- Receivers can be located anywhere in the model
- Receiver sensor axes can be given a fixed rotation (supports full 3D rotation)
- Option to solve for the receiver sensor orientations (supports solving for the full 3D rotation if z component data available)
- Transmitters can be located anywhere in the model
- Transmitter antenna can be rotated both horizontally and vertically
- Model layers are parameterized in absolute depths, not thicknesses
- There are no assumptions of fixed layers (you have to input the air and sea layers)
- Handles stratified seawater conductivity layering
- Can invert for layer conductivities both above and below the receivers and transmitters.
- Includes 1D MT inversion (alone or jointly with CSEM inversion).
- Supports point dipoles and finite length dipoles. *new February 2010, thanks to David Myer's for his efforts.*
- Supports minimum gradient support roughness penalty (also known as weighted L2 regularization). *new March 2010.*

And here's a list of current limitations:

- ~~Uses a point dipole approximation~~ , see above
- Only handles isotropic conductivity

1.1 References

The forward solver Dipole1D.f90 and its application with Occam are described in:

> Key, K., 2009, 1D inversion of multicomponent, multifrequency marine CSEM data: Methodology and synthetic studies for resolving thin resistive layers: Geophysics, 74, F9-F20, DOI: 10.1190/1.3058434.

The Occam's inversion algorithm is originally described in:

> Constable, S.C., R.L. Parker, and C.G. Constable, 1987, Occam's inversion - A practical algorithm for generating smooth models from electromagnetic sounding data: Geophysics, 52, 289-300.

The Orthogonal Procrustes Rotation Analysis (OPRA) method for determining the receiver orientation is described in a manuscript accepted to Geophysics (January 2010):

> Key, K. and A. Lockwood, Determining the orientation of marine CSEM receivers using orthogonal Procrustes rotation analysis, accepted to Geophysics.

Please reference these papers when publishing studies obtained with this code.

1.2 License

1.2.1 Dipole1D.f90 and CSEM1D.f90

Kerry Key
Scripps Institution of Oceanography
kkey@ucsd.edu

1.2.2 Occam.f90

1.3 Acknowledgments

- Steven Constable provided the original f77 Occam's inversion engine and gave helpful advice on incorporating the new CSEM forward solver Dipole1D.f90.
- David Myer ported the f77 Occam code into Fortran90/95 as part of an upgrade of the Occam2DMT inversion program. David and I have since been updating the code to Fortran2003.
- James Gunning and John Hedditch provided helpful coding suggestions.
- David Myer initiated the modifications to handle a finite length dipole in February 2010.
- This work was funded by the Seafloor Electromagnetic Methods Consortium at Scripps Institution of Oceanography. See http://marineemlab.ucsd.edu/semc.html for more info.

  The following sponsors of the Seafloor Electromagnetic Methods Consortium are thanked for their generosity:

  - Aker Exploration
  - Anadarko
  - AOA Geophysics Inc
  - BG International
  - BGP International
  - BHP Billiton
  - Black Gold Energy
  - BP
  - CGG-Veritas (through 2009)
  - Chevron
  - ConocoPhillips
  - CSIRO
  - Det Norske Oljeselskap
  - Discover Petroleum (through 2009)
  - EMGS
  - ExxonMobil
  - Fugro

- Geophysical Resources and Services
- GERD
- Idemitsu Petroleum Norge AS
- JOGMEC
- Hess Corporation
- KMS Technologies
- OHM Surveys
- Petrobas
- PGS
- Phoenix Geophysics
- QUASAR Federal Systems
- Reliance Industries
- Repsol YPF
- Rocksource ASA
- Santos (through 2009)
- Shell
- Statoil
- Total
- WesternGeco Electromagnetics
- Woodside
- Zonge Engineering and Research Organization

1.4 Bugs, Fixes, Questions and Comments

Send bug reports, fixes, questions and comments to my email: kkey@ucsd.edu. I'll try my best to answer your questions, but don't expect an immediate reply.

2 Compiling the Source Code

For instructions on installing OCCAM1DCSEM on Unix and Windows machines, see the file "INSTALL" located in the /Source folder.

3 Occam's Inversion Described in a Single Page

This section is a brief overview of Occam's inversion. It may be helpful for understanding the various parameters in the input files. If you already have a good understanding of EM inversion, you can safely skip over to the next section. More detailed descriptions of Occam can be found in Constable et al. (1987) and deGroot-Hedlin and Constable (1990). The specific application to marine CSEM is described in Key (2009), which is included in the /Documentation folder.

Occam's inversion seeks to minimize the following unconstrained regularized functional:

$$U = \|\boldsymbol{\partial}\mathbf{m}\|^2 + \|\mathbf{P}(\mathbf{m} - \mathbf{m}_*)\|^2 + \mu^{-1}\left[\|\mathbf{W}(\mathbf{d} - F(\mathbf{m}))\|^2 - \chi_*^2\right]. \tag{1}$$

The first term is a norm of the model roughness and is computed by applying a differencing operator $\boldsymbol{\partial}$ to the elements of the model vector $\mathbf{m}$. For the one dimensional models considered here, $\mathbf{m}$ is a vector of $\log_{10}\rho$ for each layer and $\boldsymbol{\partial}$ is chosen to be a matrix of first-differencing operators so that $\boldsymbol{\partial}\mathbf{m}$ approximates the vertical derivative of $\log_{10}\rho$. The parameterization with respect to $\log_{10}\rho$ ensures that resistivity remains positive during the inversion. If jumps in resistivity are desired at certain depths, the corresponding elements of $\boldsymbol{\partial}$ can be set to zero. The second term is a measure of the difference of $\mathbf{m}$ from an a priori preference model $\mathbf{m}_*$. The diagonal matrix $\mathbf{P}$ contains scaling parameters that determine the relative weighting between the preference and the model roughness. Preference model values, if used at all, are typically desired for a only few model layers and the corresponding diagonal elements of $\mathbf{P}$ will be nonzero. The roughness and preference terms in the above equation are regularizers that serve to stabilize the inversion and keep it from producing wildly oscillating resistivity structure (see the regularized/unregularized comparisons in Constable et al. (1987)).

Finally, the third term is a measure of the misfit of the model's forward response $F(\mathbf{m})$ (i.e., the electric and magnetic fields for model $\mathbf{m}$) to the data $\mathbf{d}$. There is no restriction on the data vector $\mathbf{d}$. For example, it can contain data from multiple transmitters, receivers and frequencies, and can include magnetotelluric (MT) data. $\mathbf{W}$ is a data covariance weighting function and is here selected to be a diagonal matrix with elements corresponding to inverse data standard errors. In other words, $\mathbf{W}$ weights the relative contribution of each datum to the misfit based on its uncertainty. Thus, data with large errors are scaled to limit their influence, while data with small errors will have a bigger impact on the misfit budget.

χ_*^2 is the target misfit and its inclusion illustrates that minimizing U does not necessarily find the best fitting model, but rather a smooth model that is within the specified target misfit. The target misfit χ_*^2 is usually chosen so that the root mean square (RMS) misfit x_{rms} is equal to unity:

$$x_{rms} = \sqrt{\frac{\chi^2}{n}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\frac{d_i - F_i(\mathbf{m})}{s_i}\right]^2}, \tag{2}$$

where n is the number of data and s_i is the standard error of the ith datum.

The Lagrange multiplier μ serves to balance the trade-off between the data fit and the model roughness and model preference. The nonlinear minimization of equation 1 is described in Constable et al. (1987) and one of the main innovations of the Occam method is the automatic selection of μ. So that's Occam described in a single page of text. The next few sections describe the input files required to run the OCCAM1DCSEM program.

4 Model, Transmitter and Receiver Geometry

The 1D model uses a right handed Cartesian geometry with the z axis pointing down, as shown in Figure 1. All positions are in units of meters and angles are in degrees. There are N-layers with resistivity ρ_i in units of ohm-m and each layer position is defined in terms of the top depth of the layer. There are no fixed assumptions of any layer depths, including the air-sea interface. So you will need to describe the entire model in the setup files. Receivers and transmitters can be located anywhere in the stack of layers. When modeling real survey data, you will need to decide what the x, y axes correspond to. For example, you could use UTM coordinates and have x = North, y = East. Or you could have survey local coordinates where x = inline, y = crossline, or y = inline, x = crossline. Whatever you choose, it must be consistent between the transmitters and receivers and the angles defined below. Check and then double check that you've done the correct transformations.

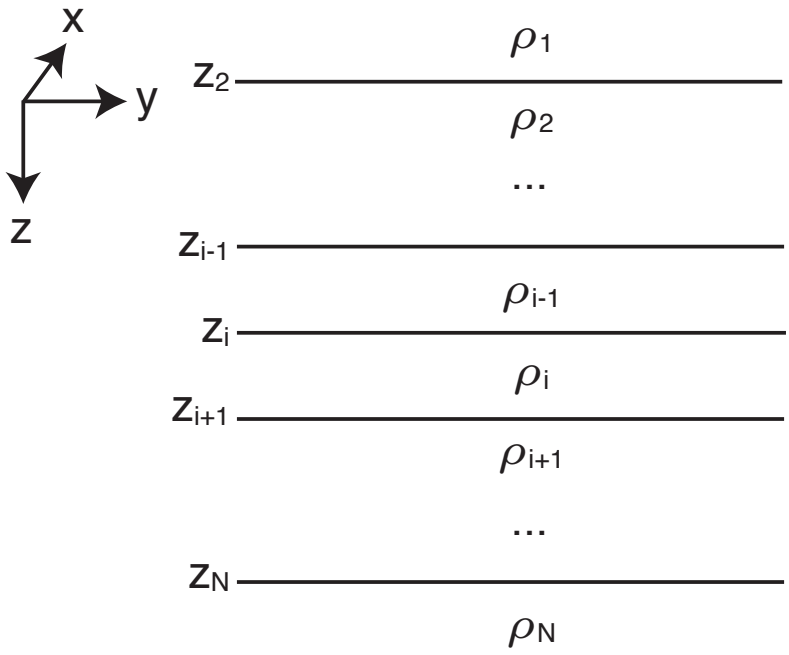


Figure 1: 1D model geometry. There are N-layers with resistivity ρ_i in units of ohm-m. Each layer is defined in terms of the absolute depth of the top of the layer in units of meters. Receivers and transmitters can be located anywhere in the stack of layers.

4.1 Transmitter Orientation

The transmitter orientation parameters are shown in Figure 2. The transmitter azimuth is defined to be the horizontal rotation of the transmitter antenna from the x axis, positive towards y. So an angle of 0 means the antenna points along x, while an angle of 90 degrees means the antenna points along y. If x is also the towline direction, then the azimuth is equivalent to the antenna yaw. The transmitter dip angle is positive down from the azimuth angle. A dip of 0 degrees means the antenna is horizontal. A dip of 90 degrees means the antenna is pointing downward. If the tail of the antenna is sagging, the dip angle will be negative. If the tail is higher than the head of the antenna, the dip angle will be positive. Dipole1D supports modeling using either a point dipole approximation (faster) or a finite length dipole (slower but more accurate for short range data). For a finite length antenna, the (x, y, z) position of the antenna should be given as the antenna midpoint. Lastly, note that the term azimuth here refers to the orientation of the antenna, not the source-receiver azimuth used in earlier cylindrical coordinate modeling parameterizations.

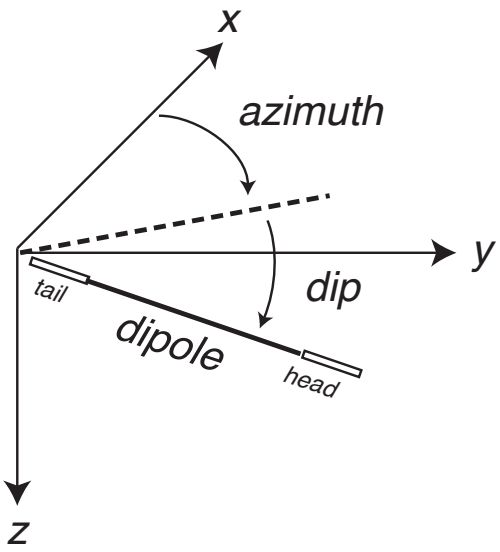


Figure 2: Transmitter orientation parameters. The transmitter azimuth is defined to be the horizontal rotation of the transmitter antenna from the x axis, positive towards y. So an azimuth of 0 means the antenna points along x, while an angle of 90 degrees means the antenna points along y. The transmitter dip angle is positive down from the azimuth angle.

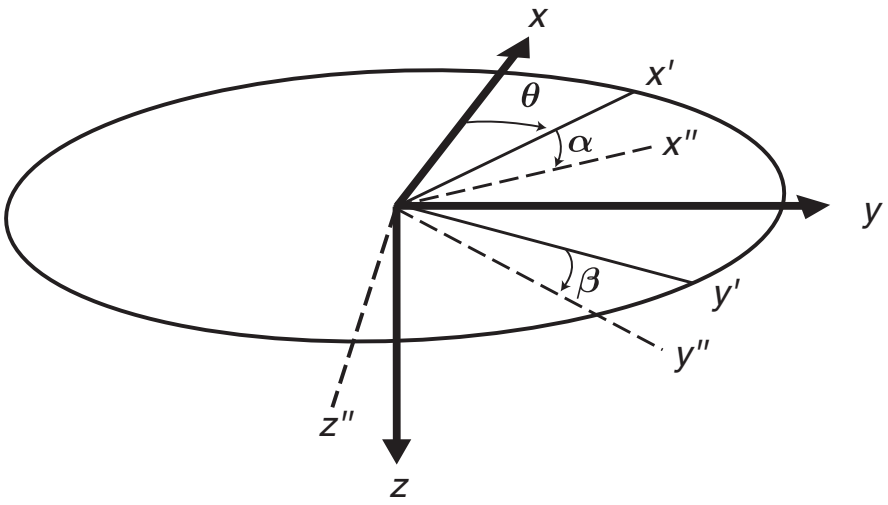


Figure 3: Rotation parameters for the absolute orientation of receiver sensors along axes (x'', y'', z''). Rotation in the horizontal plane can be described by the angle θ, which is the rotation about the z axis from the survey axes (x, y) into (x', y'). The tilt of the receiver can be specified with the angles α and β, which describe rotations about the y' and x'' axes, respectively. Note that α describes tilt in a vertically oriented plane along θ, while the angle β describes tilt along a dipping plane when $\alpha \neq 0$. The circle denotes the horizontal plane.

4.2 Receiver Orientation

OCCAM1DCSEM can model receiver data pre-rotated to the survey axes (x, y, z), or at whatever arbitrarily rotated axes the data are collected along (x'', y'', z''). To do this, the full 3D rotation of the receiver needs to be specified using three angles. The orientation of the sensor axes (x'', y'', z'') from the survey axes (x, y, z) can be described by three angles: θ, α, and β, as shown in Figure 3. Below I give a mathematical description of these rotation parameters.

The complex valued vector electric and magnetic field data $\mathbf{F}''$ recorded in the receiver's rotated coordinate system are related to the fields $\mathbf{F}$ in the survey coordinate system by a 3×3 orthonormal rotation matrix $\mathbf{R}$ as

$$\mathbf{F}'' = \mathbf{F}\mathbf{R}. \tag{3}$$

For n discrete source-receiver offsets, $\mathbf{F}$ and $\mathbf{F}''$ can be written as $2n \times 3$ arrays:

$$\mathbf{F} = \begin{bmatrix} \mathrm{Re}(F_{x,1}) & \mathrm{Re}(F_{y,1}) & \mathrm{Re}(F_{z,1}) \\ \mathrm{Im}(F_{x,1}) & \mathrm{Im}(F_{y,1}) & \mathrm{Im}(F_{z,1}) \\ \vdots & \vdots & \vdots \\ \mathrm{Re}(F_{x,n}) & \mathrm{Re}(F_{y,n}) & \mathrm{Re}(F_{z,n}) \\ \mathrm{Im}(F_{x,n}) & \mathrm{Im}(F_{y,n}) & \mathrm{Im}(F_{z,n}) \end{bmatrix}, \tag{4}$$

where $F_{x,i}$ specifies the x component of the i-th datum and the real and imaginary parts of the complex data have been separated into alternating rows. The rotation matrix $\mathbf{R}$ can be written as

a product of rotations about individual axes:

$$\mathbf{R} = \mathbf{R}_z \mathbf{R}_{y'} \mathbf{R}_{x''}, \tag{5}$$

where

$$\mathbf{R}_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6}$$

$$\mathbf{R}_{y'} = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}, \tag{7}$$

and

$$\mathbf{R}_{x''} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix}. \tag{8}$$

$\mathbf{R}_z$ is a rotation by θ about the z axis, giving new coordinate axes (x', y', z). $\mathbf{R}_{y'}$ is a rotation by α about the y' axis, giving new coordinate axes (x'', y', z'). $\mathbf{R}_{x''}$ is a rotation by β about the x'' axis, giving the final rotated axes (x'', y'', z'').

If you specify non-zero rotation parameters for a given receiver, then OCCAM1DCSEM will rotate the model response to the data orientation using equation 3.

5 Input Files Required by OCCAM1DCSEM

OCCAM1DCSEM requires three input files: IterationFile, ModelFile, and DataFile. These files are simple text files and use a flexible format that allows for blank lines and comments (text after ! and % signs is ignored). The basic structure for many of the lines in these files is *keyword : value*, where *keyword* specifies a parameter to set and *value* is the value to use. They are separated by a colon. All text fields are case insensitive, except for the strings listing file names. Certain *keyword : value* pairs are followed with arrays of numbers (receiver or transmitter locations, etc). Also, don't be a moron and use file names that have spaces in them :)

The following subsections describe the details of each file, and you can also study the examples provided with the OCCAM1DCSEM code distribution in the folder /Examples. Additionally, there is a Matlab routine called createEMDataFile.m which you can use to convert data to the correct format and to create the IterationFile, ModelFile, and DataFile. So if you tire easily when reading verbose file format descriptions, skip on ahead to the Matlab code section and remember to refer back to this section if you have any problems.

5.1 Iteration File Format

The IterationFile is the first file read by OCCAM1DCSEM and specifies the parameters that Occam's inversion will use to begin its search for the smoothest model that fits the data. By default, this file is named "startup" on the first iteration of Occam, but a different file name can be specified using a command line argument (described in Section 6). The code is backwards compatible with the older format OCCAMITER_1.0, but I strongly recommend using David Myer's new flexible format OCCAMITER_FLEX. Here is an example IterationFile, with brief descriptions after the ! symbol:

```
Format:             OCCAMITER_FLEX      ! Flexible format
Description:        test                ! Not used by Occam, but you can use this for your own notes.
Model File:         model               ! Name of the Model File. Case sensitive.
Data File:          data                ! Name of the Data File. Case sensitive.
Date/Time:          11/25/2008 16:46:39 ! On output, date and time stamp placed here.
Max Iter:           100                 ! Maximum number of Occam iterations to compute.
Target Misfit:      1.0                 ! Target RMS misfit, see equation 2.
Roughness Type:     1                   ! See section below for Roughness Penalty options
!Model Limits:      min,max             ! Optional, places hard limits on log10(rho) values.
!Model Value Steps: stepsize            ! Optional, forces model into discrete steps of stepsize.
Debug Level:        1                   ! Console output. 0: minimal, 1: default, 2: detailed
Iteration:          0                   ! Iteration number, use 0 for starting from scratch.
Lagrange Value:     5.000000            ! log10(largrance multiplier), starting value.
Roughness Value:    0.1000000E+08       ! Roughness of last model, ignored on startup
Misfit Value:       100.0000            ! Misfit of model listed below. Ignored on startup
Misfit Reached:     0                   ! 0: not reached, 1: reached.  Useful when restarting.
Param Count:        75                  ! Number of free inversion parameters.
     0             ! A listing of the free inversion parameter values in log10(resistivity).
     0             ! Here the 0's correspond to log10(1 ohm-m).
     0             ! Since this is iteration 0, the inversion will thus be started with a
     0             ! 1 ohm-m halfspace.
     0
     0
     0
     0
     0
     0
     0
     0
...
```

Note that the Model Limits and Model Value Steps fields are optional. If you want to use these, then remove the ! comments at the beginning of the lines.

5.1.1 Roughness Type Options

Occam1DCSEM supports a few options for the roughness penalty used to regularize (i.e., stabilize) the non-linear inversion. The roughness type parameter can either be entered as an integer code or as a string value, as listed in Table 1.

Table 1: Roughness Type Options

Integer Code	String Code	Description
1	'FirstDiff'	First differencing penalty, *default value*
4	'DepthWeighted'	log10(depth) weighted penalty
	'mgs, δ'	Minimum gradient support penalty

The roughness penalty corresponds to the term $\|\boldsymbol{\partial}\mathbf{m}\|^2$ in equation 1, but here we generalize this into the roughness operator $\mathcal{R}$:

$$\|\mathcal{R}(\mathbf{m})\|^2 = \|\mathbf{W}_{\mathcal{R}}\boldsymbol{\partial}\mathbf{m}\|^2, \tag{9}$$

where $\mathbf{W}_{\mathcal{R}}$ is a diagonal weighting matrix that varies according to the roughness penalty type. For standard first differencing, $\mathbf{W}_{\mathcal{R}} = \mathbf{I}$, meaning the weights are all unity. This is the default regularization used by Occam1DCSEM.

For Depth Weighted roughness, the diagonal elements of $\mathbf{W}_{\mathcal{R}}$ are the logarithm of the depth of each layer (relative to the top layer). In other words, the roughness penalty slowly increases with depth in the model.

For minimum gradient support, the weight values have a more complicated form that depends on the gradient of the previous model's conductivity parameters. If the current model parameter vector is $\mathbf{m}_{j+1}$, then the i-th diagonal element of $\mathbf{W}_{\mathcal{R}}$ has the form

$$\mathbf{W}_{\mathcal{R}}(i) = \frac{1}{\sqrt{(\mathbf{m}_j(i) - \mathbf{m}_j(i-1))^2 + \delta^2}} \tag{10}$$

This form of the weighting allows for a relaxation of the roughness penalty in regions where the model begins to develop *large* gradients in the model parameters. The definition of *large* is controlled through the δ parameter, which also keeps the denominator from going to zero. This type of roughness penalty is also known as weighted L2 regularization. For further discussion, see Portniaguine and Zhdanov (1999) and van den Berg and Abubakar (2001). To use minimum gradient support, set the roughness type using:

```
Roughness Type:     mgs,0.1
```

where 0.1 or another desired value specifies δ.

Examples of the effect of the various roughness operators are shown in Figure 21 at the end of these instructions. In our experience, the standard first differencing roughness operator is the most reliable for keeping the inversion process stable so that it can find a model at the target RMS. The minimum gradient support roughness penalty allows for much sharper features to develop, often producing a more geologically satisfying model. However, this regularization is much less stable; it is not uncommon for the inversion can get stuck in local minima before it obtains the target RMS (this of course also depends on the assigned value of δ). For example, standard first differencing might be able to fit a data set to RMS 1.0, whereas the minimum gradient support roughness might result in the inversion getting stuck at RMS 2.1 or something else much larger than 1.0. If this

happens, trying inverting with different values for δ until the inversion is able to fit the data as well as the standard first difference roughness penalty (if at all).

5.2 Model File Format

The model file describes the 1D model layering and denotes whether a given layer has a fixed resistivity or is a free inversion parameter. It also specifies the roughness penalty weight applied across each layer boundary, and whether any free layers have preferred resistivity values. Note that while the Iteration file uses log10(resistivity) for the inversion parameters, the model file uses linear resistivity for any fixed layers or any preference layers (see below). Here is an example model file using the only currently supported format, Resistivity1DMod_1.0:

```
Format: Resistivity1DMod_1.0
#Layers:    77
! Layer block listing is:
! [top_depth  resistivity    penalty  preference   pref_penalty]
      -100000       1d12           0           0           0  ! Air, fixed layer of 10^12 ohm-m
           0        0.3           0           0           0  ! Sea, fixed layer of 0.3 ohm-m
        1000         -1           1           0           0  ! First free layer, the seafloor @ 1000 m
        1025         -1           1           0           0  ! use -1 or ? to denote free layer resistivity
        1050         -1           1           0           0
        1075         -1           1           0           0
        1100         -1           1           0           0
        1125         -1           1           0           0
        1150         -1           1           0           0
        1175         -1           1           0           0
        1200         -1           1           0           0
        1225         -1           1           0           0
        1250         -1           1           0           0
        1275         -1           1           0           0
        ...
```

The layer top depth is given in absolute units as meters and the top depth of the first layer is ignored by Dipole1D. Fixed resistivities are given in units of **linear** resistivity (ohm-m). For example, the above example has two fixed layers, the air (1d12 ohm-m, or 10^{12} ohm-m) and a single sea layer (0.3 ohm-m). However, you could also include a stratified seawater resistivity profile, as described in Key (2009).

For free inversion layers, the resistivity value is flagged as a free parameter using either of a -1 or a ?. The penalty value is an user specified weight applied the roughness operator, as applied to the resistivity jump across each layer boundary where both layers resistivities are free parameters. Usually this will just be a 1. If you want to allow a jump in conductivity at a certain depth, you can relax the roughness penalty at that depth by setting the penalty weight to 0 (or something smaller than 1) at that layer boundary depth. Here's an example matrix representation of how the penalty weights (w_i) are applied to the default first difference roughness operator (matrix of 1's and -1's) and the model vector (m_i) in equation 1, for a simplistic 5 layer model:

$$\boldsymbol{\partial}\mathbf{m} = \begin{bmatrix} w_1 & 0 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 & 0 \\ 0 & 0 & w_3 & 0 & 0 \\ 0 & 0 & 0 & w_4 & 0 \\ 0 & 0 & 0 & 0 & w_5 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix}$$

The form of the weights is flexible. Usually just a 1 is sufficient. However, you could make the penalty weights (w_i) increase with depth if you thought there was a good geological reason for the model to be smoother with depth. Or you could scale them by the layer thicknesses if you want the roughness penalty to approximate the vertical spatial derivative.

The preference value is the **linear** resistivity (ohm-m) that you might prefer for certain layers, although this feature is not commonly used. For example, you might have a priori geologic information that a certain layer is probably resistive, and you can impart this preference on Occam by specifying the preferred value. A good case for using a preference value is when you want to test a hypothesis of whether a certain layer is resistive (or conversely conductive). You can invert the data with the preference value given and see how well Occam is able to fit the data. If the inversion isn't able to fit the data well, or the inverted resistivity is far from the preference, you might be able to rule out the presence of certain structures. Also, in some inversions i've performed on real survey data, i've found that adding penalty cuts to the roughness operator can sometimes destabilize the inversion to the point of it wandering off to infinity. I was able to re-stablize the inversion by setting some preference resistivity values for the thin layers between the penalty jumps. However, beware that adding too many cuts and preference resistivity values will ultimately turn a fairly objective inversion into a "desired output" filter.

If you want a preference value, then you need to also set the pref_penalty (its weight) to a non-zero value (1 is the nominal value), otherwise it will be ignored when pref_penalty=0. Here's a matrix representation of the preference terms in equation 1 for a simplistic 5 layer model:

$$\mathbf{P}(\mathbf{m}-\mathbf{m}_*) = \begin{bmatrix} p_1 & 0 & 0 & 0 & 0 \\ 0 & p_2 & 0 & 0 & 0 \\ 0 & 0 & p_3 & 0 & 0 \\ 0 & 0 & 0 & p_4 & 0 \\ 0 & 0 & 0 & 0 & p_5 \end{bmatrix} \begin{bmatrix} m_1 - m_{*1} \\ m_2 - m_{*2} \\ m_3 - m_{*3} \\ m_4 - m_{*4} \\ m_5 - m_{*5} \end{bmatrix}$$

So setting the preference penalty weight $p_i > 0$ and preference resistivity m_{*i} means that the difference in $m_i - m_{*i}$ will be minimized by the inversion, subject to the constraint that the data still needs to be fit. Note that the preference value is given as linear resistivity, but internally this is converted to log10(resistivity). Here's a toy example ModelFile that includes two cuts in the roughness penalty, a preferred resistivity layer and uses ? instead of -1 to denote the free parameter layers:

```
Format: Resistivity1DMod_1.0
#Layers:    77
! Layer block listing is:
! [top_depth  resistivity    penalty  preference   pref_penalty]
      -100000       1d12           0           0           0  ! Air, fixed layer of 10^12 ohm-m
           0        0.3           0           0           0  ! Sea, fixed layer of 0.3 ohm-m
        1000          ?           1           0           0  ! First free layer, the seafloor @ 1000 m
        1025          ?           1           0           0  ! use -1 or ? to denote free layer resistivity
        1050          ?           1           0           0
        1075          ?           1           0           0
        1100          ?           0           0           0  ! Allows a resistivity jump at 1100 m
        1125          ?           1           0           0
        1150          ?           0           0           0  ! Allows a resistivity jump at 1150 m
        1175          ?           1           0           0
```

```
1200          ?          1          55          1  ! A 55 ohm-m preferred layer, spanning
1225          ?          1          55          1  ! from  1200-1250 m depth
1250          ?          1           0          0
1275          ?          1           0          0
...
```

5.3 Data File Format

The data file lists the transmitters, receivers and frequencies, and has a table of data parameters, values and standard errors. The currently supported data formats are named EMData_1.1 and EMData_1.2 (technically EMData_1.0 is still supported, but it was lacking so I won't describe it here). This format is designed to be compatible with future 2D and 3D inversion codes and hence is very general. Like the Iteration and Model files, the Data file format can include arbitrary comment lines (using ! or %) or blank lines. Here's an example DataFile:

```
Format:  EMData_1.2
Dipole Length: 250        ! [m] Optional, the default is 0 m which is a point dipole
# integ pts:   10         ! Optional and only used if Dipole Length > 0.
# Transmitters:   41
!          X            Y            Z      AZIMUTH          DIP
           0            0          950           90            0
           0          500          950           90            0
           0         1000          950           90            0
          ...
# Frequencies:    2
         0.1
           1
# Receivers:      20
!           X           Y            Z         Theta        Alpha         Beta
            0           0         1000             0            0            0
            0        1000         1000             0            0            0
            0        2000         1000             0            0            0
           ...
# Data:       714
!         TYPE        FREQ#        TX#            RX#        DATA    SD_ERROR
        RealEy            1           1              1 -2.94217e-07  5.86785e-09
        ImagEy            1           1              1  4.71955e-10  5.86785e-09
        RealEz            1           1              1  6.95924e-09  5.86785e-09
        ImagEz            1           1              1 -1.35431e-09  5.86785e-09
        RealBx            1           1              1  2.89886e-11  5.84098e-13
        ImagBx            1           1              1 -1.82912e-13  5.84098e-13
        RealEy            1           2              1  5.35934e-10  1.08866e-11
        ImagEy            1           2              1  7.50943e-11  1.08866e-11
        RealEz            1           2              1 -4.89381e-11  1.08866e-11
          ...
```

The Transmitters block lists the transmitter dipole length, the number of transmitters and then the x, y, z (meters) locations of the transmitters, and their rotation angle (degrees clockwise from x) and dip angle (degrees positive down). Note that both the "Dipole Length" parameter is optional. Set this to a non-zero positive number to model a finite length dipole. The parameter "# integ pts" is optional and is only used if "Dipole Length" > 0. This parameter specifies the number of points to use to the Gauss quadrature integration for the finite length dipole. A default value of

10 is used if you don't specify a value in the data file. The runtime of the code increases with this parameter (each integration point is an additional transmitter to model), so you may want to optimize this parameter for your particular data set. For example, you might be able to get away with a value of 2 or 3 if you are at least a couple of dipole lengths away from the transmitter.

The Frequencies block lists the number of frequencies and then the values (Hz).

The Receivers block lists the number of receivers and then the x, y, z (meters) positions of the receivers and each receiver's rotation angles, as shown in Figure 3.

The Data block lists the number of data and then contains a table of data parameters, data and standard errors. The first column of the data table specifies the data type. The currently supported data types for CSEM and MT data are given in Table 2. The data type can be specified using either the string codes or the numeric codes given in Table 2. The string values are not case sensitive. The second column of the data table is the frequency index for each datum. The third column is the transmitter index and the fourth column gives the receiver index. The fifth and sixth columns are the data and standard errors. Electric fields should be given in units of V/Am^2 and magnetic fields in units of T/Am. For MT data, the impedances are in units of ohms, apparent resistivities in units of linear ohm-m and phases in degrees. Impedance phases (for Zyx) should be moved to the first quadrant (nominally 45 degrees, not -135 degrees).

That covers the basics of the data file. However, there are a few optional fields that may help you deal with the details of real survey data. These are explained in the following sections. These are not required in the Data file, but if present they are used.

Table 2: Data types used in format EMData_1.1

	Integer Code	String Code	Description
CSEM Data:	1	RealEx	Real Ex
	2	ImagEx	Imaginary Ex
	3	RealEy	Real Ey
	4	ImagEy	Imaginary Ey
	5	RealEz	Real Ez
	6	ImagEz	Imaginary Ez
	11	RealBx	Real Bx
	12	ImagBx	Imaginary Bx
	13	RealBy	Real By
	14	ImagBy	Imaginary By
	15	RealBz	Real Bz
	16	ImagBz	Imaginary Bz
	21	AmpEx	Amplitude Ex
	22	PhsEx	Phase Ex
	23	AmpEy	Amplitude Ey
	24	PhsEy	Phase Ey
	25	AmpEz	Amplitude Ez
	26	PhsEz	Phase Ez
	31	AmpBx	Amplitude Bx
	32	PhsBx	Phase Bx
	33	AmpBy	Amplitude By
	34	PhsBy	Phase By
	35	AmpBz	Amplitude Bz
	36	PhsBz	Phase Bz
	41	PEmax	Electric horizontal polarization ellipse maximum
	42	PEmin	Electric horizontal polarization ellipse minimum
	43	PBmax	Magnetic horizontal polarization ellipse maximum
	44	PBmin	Magnetic horizontal polarization ellipse minimum
MT Data:	101	RhoZxx	Apparent Resistivity Zxx, ignored in 1D
	102	PhsZxx	Phase Zxx, ignored in 1D
	103	RhoZxy	Apparent Resistivity Zxy
	104	PhsZxy	Phase Zxy
	105	RhoZyx	Apparent Resistivity Zyx
	106	PhsZyx	Phase Zyx
	107	RhoZyy	Apparent Resistivity Zyy, ignored in 1D
	108	PhsZyy	Phase Zyy, ignored in 1D
	111	RealZxx	Real Zxx, ignored in 1D
	112	ImagZxx	Imaginary Zxx, ignored in 1D
	113	RealZxy	Real Zxy
	114	ImagZxy	Imaginary Zxy
	115	RealZyx	Real Zyx
	116	ImagZyx	Imaginary Zyx
	117	RealZyy	Real Zyy, ignored in 1D
	118	ImagZyy	Imaginary Zyy, ignored in 1D

5.3.1 Phase Convention

The default convention used in DIPOLE1D is phase lag (i.e., phases become increasingly positive with source-receiver offset), but you can instead specify that the data use a phase lead convention (i.e., phases become increasingly negative with source-receiver offset):

```
Phase Convention: lag          ! Optional, use lag (default) or lead
```

5.3.2 Solving For Unknown Receiver Rotations

There is an option for specifying that the receiver rotations are unknown and then the inversion will solve for the best fitting rotations. Simply put, you just list ? for the receiver angles instead of numeric values. Here's how it works.

The geometry of the receiver orientation was described in section 4.2. The receiver's rotation angles shown in Figure 3 can be estimated by solving the minimization problem:

$$\min_{\mathbf{R}} \ \|\hat{\mathbf{F}}'' - \hat{\mathbf{F}}\mathbf{R}\|_F^2 \ \text{ subject to } \ \mathbf{R}^T\mathbf{R} = \mathbf{I}, \tag{11}$$

where $\mathbf{I}$ is the identity matrix and the Frobenius norm $\|\mathbf{A}\|_F^2 = \text{tr}(\mathbf{A}^T\mathbf{A})$. The data arrays are normalized using $\hat{\mathbf{F}}'' = \mathbf{W}\mathbf{F}''$ and $\hat{\mathbf{F}} = \mathbf{W}\mathbf{F}$, where the weighting function $\mathbf{W}$ is a diagonal matrix of the inverse data standard errors that serves to reduce the contribution of noisy data. Although we can never know the true unrotated field vector $\mathbf{F}$, we can approximate it using the forward response from each Occam iteration. The minimization of the above equation is then accomplished using the Orthogonal Procrustes Rotation Analysis (OPRA) method.

This is available as an option for OCCAM1DCSEM inversions. To use the OPRA method, you will need to set the receiver orientation angles to ? marks:

```
...
# Receivers:      1
!           X          Y           Z         Theta         Alpha          Beta
            0          0        1000             ?             ?             ?
           ...
```

Most of the time there will be only a single receiver (as shown above). However, you could solve for multiple receiver orientations or have some combination of known and unknown receivers all in the same data file.

There are some restrictions on the data array if you enable the OPRA method. First, the data must be given as real/imaginary components and they must be listed by site in order as RealEx, ImagEx, RealEy, ImagEy, RealEz, ImagEz, etc, as shown in this example below. Also note that the data standard errors must be isotropic among all x, y, z components. Typically I use some fraction (1-10%) of the total vector amplitude as the standard error.

```
!          TYPE         FREQ#         TX#           RX#        DATA     SD_ERROR
            1             1            1             1   2.04917e-07  5.86785e-09
            2             1            1             1   2.49658e-09  5.86785e-09
            3             1            1             1   2.20625e-07  5.86785e-09
            4             1            1             1   -3.7858e-10  5.86785e-09
            5             1            1             1   1.71885e-08  5.86785e-09
            6             1            1             1 -2.68835e-09  5.86785e-09
            1             1            2             1 -3.80006e-10  1.08866e-11
            2             1            2             1 -5.97309e-11  1.08866e-11
...
```

You can also include magnetic field data using the same ordering. If the data only contains the horizontal components, then only the angle θ will be estimated. If the data contain all x, y, z components then all three angles θ, α, β will be estimated.

OCCAM will then find the best orientation angles during each inversion iteration and these are written out to a file with the name "iter_X.rotations", where X is the iteration number. The model response files (.resp) will have the data and model responses rotated to the cartesian x, y, z axes, and this is noted in the "Receiver Orientations" block of the response file. So in other words, the data in the data file will be in the original receiver orientation while the data and responses in the .resp file are aligned along x, y, z.

Some synthetic rotated data examples are given in the /Canonical_Solve_3DRotation and /Canonical_Solve_HorizRotation folders in the /Examples folder. See the comments in the data files there for what the data orientation angles are. For both of these examples, OCCAM1DCSEM and OPRA are able to fit the data to RMS 1.0 and find the orientations angles to about 1° error. For real data with 2D and 3D structures, you might find reasonable orientation estimates by using either short offset data (dominated by diffusion through the seawater), or by using only long offset data (dominated by diffusion through air), but this depends on how non-1D the structure is and on how reliably the transmitter and receiver positions are known.

5.3.3 A Note about 1D Reciprocity

The bulk of the inversion CPU time is dominated by the 1D CSEM forward and sensitivity computations. These computations depend linearly on the number of discrete transmitter locations. Thus, the fewer the transmitters, the faster the runtime. For 1D modeling, it is often possible to use reciprocity to convert multiple transmitters into multiple receivers, and hence speed up the runtime. If you input a vector of transmitters that all have the same depth, azimuth and dip but different x,y positions, the code is smart enough to apply 1D reciprocity and convert the transmitters into many receivers and will then run much faster. This is all done automatically and as an end user you should only notice that the runtime is fast. Conversely, if the input transmitters are depth varying or have varying dips and azimuths then 1D reciprocity is not used and the runtime can be substantially longer. There are some additional ways to speed up the code for varying dips and azimuths, but I haven't code them up yet since for the most part the code should run fast enough for most users needs.

5.3.4 A Note about Standard Errors

This section contains some notes that might be useful for converting the data uncertainties between real and imaginary components and amplitude and phase. The complex CSEM data z can be represented in either form by using Euler's formula:

$$z = re^{i\phi} = u + i\,v, \tag{12}$$

where r is the amplitude, ϕ is the phase, u is the real component, v is the imaginary component and $i = \sqrt{-1}$. These are shown graphically in Figure 4. We generally assume that the data uncertainty δz is isotropic with respect to the complex data z, as shown by the gray circle in the figure. This is written as

$$\delta z = \delta u + i\,\delta v, \quad \text{where} \quad \delta u = \delta v. \tag{13}$$

Isotropic uncertainty also means the amplitude uncertainty $\delta r = \delta u = \delta v$. Lastly, the uncertainty in the phase ϕ can be found by noting that the arc length associated with isotropic uncertainty δr is

$$r\delta\phi \approx \delta r. \tag{14}$$

Therefore, the corresponding phase uncertainty $\delta\phi$ is

$$\delta\phi \approx \frac{\delta r}{r}\frac{180}{\pi}, \tag{15}$$

where the last factor converts radians to degrees. For example, a relative uncertainty $\delta r/r = 1\%$ has a corresponding phase uncertainty $\delta\phi = 0.573^\circ$, while $\delta r/r = 10\%$ gives $\delta\phi = 5.73^\circ$.

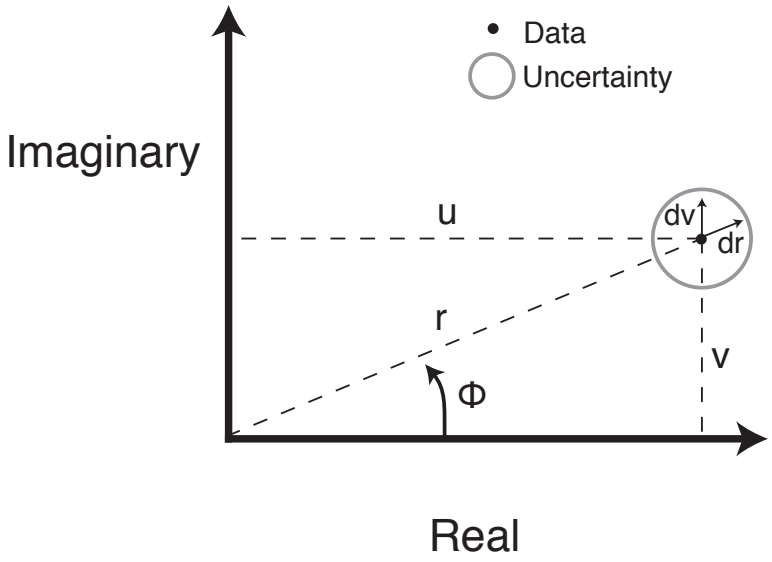


Figure 4: Complex data and uncertainty diagram showing the relationship between real and imaginary components (u and v) and amplitude (r) and phase (ϕ).

6 Running OCCAM1DCSEM

Thanks to David Myer's efforts, Occam now supports a few command line parameters for specifying run options. To run OCCAM1DCSEM, use this syntax:

```
OCCAM1DCSEM [-F] [ <IterationFileName> [<OutputFileNameRoot>] ]
```

where all the quantities inside [ ] are optional.

Options:

-F Specifies to compute only the forward response of the starting IterationFile. The response is written out to a .resp file.

IterationFileName Specifies the IterationFile to start Occam with. If this file is not given, then Occam by default will assume that the name of the starting IterationFile is "startup".

OutputFileNameRoot Specifies the root name for all files output during the Occam inversion.

Here are a few examples:

```
$ OCCAM1DCSEM
```

Here Occam will read in the file "startup" and each iteration, model response and Occam logfile will be output to the files iter_X.iter and iter_X.resp and and logfile.logfile, where X is the iteration number.

```
$ OCCAM1DCSEM Trough.iter Trough
```

Here Occam will read in the file "Trough.iter " and each iteration, model response and Occam logfile will be output to the file Trough_X.iter and Trough_X.resp and and Trough.logfile.

```
$ OCCAM1DCSEM Trough_20.iter Trough
```

Here Occam will begin with the iteration file "Trough_20.iter " which is presumeably the 20th iteration of a previous Occam run.

```
$ OCCAM1DCSEM -F  MyHypothesisModel
```

Here Occam reads in the startup file "MyHypothesisModel" and will compute the forward response and misfit of MyHypothesisModel, and will then output the response to the file "MyHypothesis-Model.resp".

When OCCAM1DCSEM begins, it will print out the information contained in the Data and Model files and will then proceed with the Occam iterations. During each iteration, the misfit as a function of Lagrange multiplier and model roughness will be output to the terminal console. At the end of each iteration, the best model misfit, roughness and Lagrange multiplier values are output to the console. Much of this information is also written to the .logfile.

6.1 Output Files

At the end of each iteration Occam will output an IterationFile with the best model parameters found during that iteration. This file has the same format as the input IterationFile described in Section 5.1. Additionally, the response of those model parameters is written out to a .resp file. The .resp file is nearly the same as the data file, but the data table portion is a #data $\times$ 8 array expanded to include the model response. The first six columns are just a copy of the data parameters, data and standard errors given in the data table, the seventh and eighth columns are the model response and the normalized residual (i.e., the number of standard deviations for the fit of that datum):

$$\text{residual} = \frac{d_i - F_i(\mathbf{m})}{s_i} \tag{16}$$

Here's an example of the data/response section of the response file:

```
! Type      Freq#          Tx#          Rx#          Data      StdError      Response      Residual
   3            1            1            1  -0.29857E-06  0.58679E-08  -0.29039E-06         -1.39
   4            1            1            1  -0.10099E-08  0.58679E-08   0.49898E-09         -0.26
   5            1            1            1  -0.23965E-08  0.58679E-08   0.54621E-08         -1.34
   6            1            1            1  -0.50163E-08  0.58679E-08   0.44280E-11         -0.86
  11            1            1            1   0.29732E-10  0.58410E-12   0.29506E-10          0.39
  12            1            1            1   0.54974E-12  0.58410E-12   0.10734E-12          0.76
   3            1            2            1   0.52793E-09  0.10887E-10   0.53702E-09         -0.84
   4            1            2            1   0.82776E-10  0.10887E-10   0.80643E-10          0.20
   5            1            2            1  -0.32813E-10  0.10887E-10  -0.29366E-10         -0.32
   6            1            2            1   0.17868E-10  0.10887E-10   0.86575E-11          0.85
         ...
```

7 DIPOLE1D - 1D Forward Modeling

The Fortran program DIPOLE1D can be used for rapid forward model studies. DIPOLE1D requires a single input file name RUNFILE to compute CSEM responses. See the folder Dipole1D located provided with several of the test data sets in /Examples/Synthetic for an example RUNFILEs. The .csem response files output by DIPOLE1D can be plotted using plotCSEM.m and you can also use them to create synthetic noisy data files with the createEMDataFile.m user interface. If you're only interested in computing forward responses for various models, DIPOLE1D is much simpler to setup than using Occam with the -F command line option (which requires that you also build the data table).

7.1 Calling DIPOLE1D directly from Matlab

In addition to running DIPOLE1D in a terminal window, you can now call it directly from Matlab, thanks to David Myer's efforts at writing the requisite Mex file for DIPOLE1D. To get this working on you system, see read the instructions in the file INSTALL.txt in /Source. Here's the Matlab help text for running mexDipole1D.m:

```
! MatLab interface for Kerry Key's Dipole1D CSEM forward model code.
! Gateway function code patterned after 'yprimefg.f' MatLab example code.
!
! David Myer
! Copyright 2009
! Scripps Institution of Oceanography
! dmyer@ucsd.edu
!
! This function ('mexDipole1D' in MatLab, NOT 'MEXFUNCTION') definition:
! Params:
!       nTransmitter - array with cols: X, Y, Z, Azimuth, Dip
!       nFreq        - vector of frequencies
!       nLayers      - array with cols: TopDepth, Resistivity
!       nReceiver    - array with cols: X, Y, Z
!       nTalk        - (optional) 0=say nothing, 1=show fwd calls,
!                        2=spew lots of text (default)
!       bPhaseConv   - (optional) 0=phase lag (dflt), 1=phase lead
!       nDipoleLen   - (optional; dflt=0) length of dipole in meters centered
!                        at the transmitter position(s).  0 = point dipole.
!       numIntegPts - (optional; dflt=10) number of integration points
!                        for Gauss quadrature integrations for finite dipole computations.
!
! Returns:
!       nFields     - array with cols:
!                        Xmtr #, Freq #, Rcvr #, Ex, Ey, Ez, Bx, By, Bz
!                     Sorted by columns 1,2,3
```

8 Matlab Editing and Plotting Tools

There are a few Matlab plotting routines in the /Matlab directory for editing data, and plotting up the inversion models, the data and the model responses. To use these, you will need to add this directory and its subdirectories to Matlab's search path. You can do this in Matlab by selecting "File → Set Path...", then clicking the "Add with subfolders" button and then selecting the /Matlab directory. These routines should meet most users' plotting needs. If you want to use your own custom plotting routines, you might find the file reading kernels in the directory /Matlab/a_utils useful.

Here's a summary of the routines, and the next few sections describe the details for each:

createEMDataFile.m	An interactive graphical user interface for editing data and creating the data, model and startup files used by OCCAM1DCSEM.
plotCSEM.m	An interactive GUI for plotting data and model responses.
plotOccam1DCSEM.m	Plots the inversion models.
plotOccamIterMisfit.m	Plots the search performance of the Occam inversion iterations.

The data plotting routines currently support these data file formats:

EMData_1.1, EMData_1.2	Occam1DCSEM data file format
EMResp_1.1, EMResp_1.2	Occam1DCSEM model response file format
Dipole1D_1.0, Dipole1D_1.1	DIPOLE1D forward model response file format
MARE2DCSEM_2.0	MARE2DCSEM finite element forward modeling code response file format.

***If you would like to have your data format supported, kindly send me a description of the format and an example file and I'd be happy to add code to support it. Or you can do this yourself by looking at the code readCSEM.m in the /Matlab/plotCSEM/a_util folder and adding a call to a function that reads your data format into the CSEM structure defined in readCSEM.m.

8.1 Editing EM Data files with createEMDataFile.m

This utility is a graphical user interface that reads in CSEM data and allows you to select which components and frequencies to output, set rotations, set noise levels, select good and bad data segments, and will then output an EMData_1.1 format data file. It can also be used to create the model and startup files required by OCCAM1DCSEM. You can also use this tool to read in forward model responses compute by MARE2DCSEM or DIPOLE1D, add noise to them and then create a synthetic data file for inverting with OCCAM1DCSEM.

The createEMDataFile.m user interface is called by typing this into the Matlab command window:

```
>> createEMDataFile
```

The user interface shown in Figure 5 will then appear.

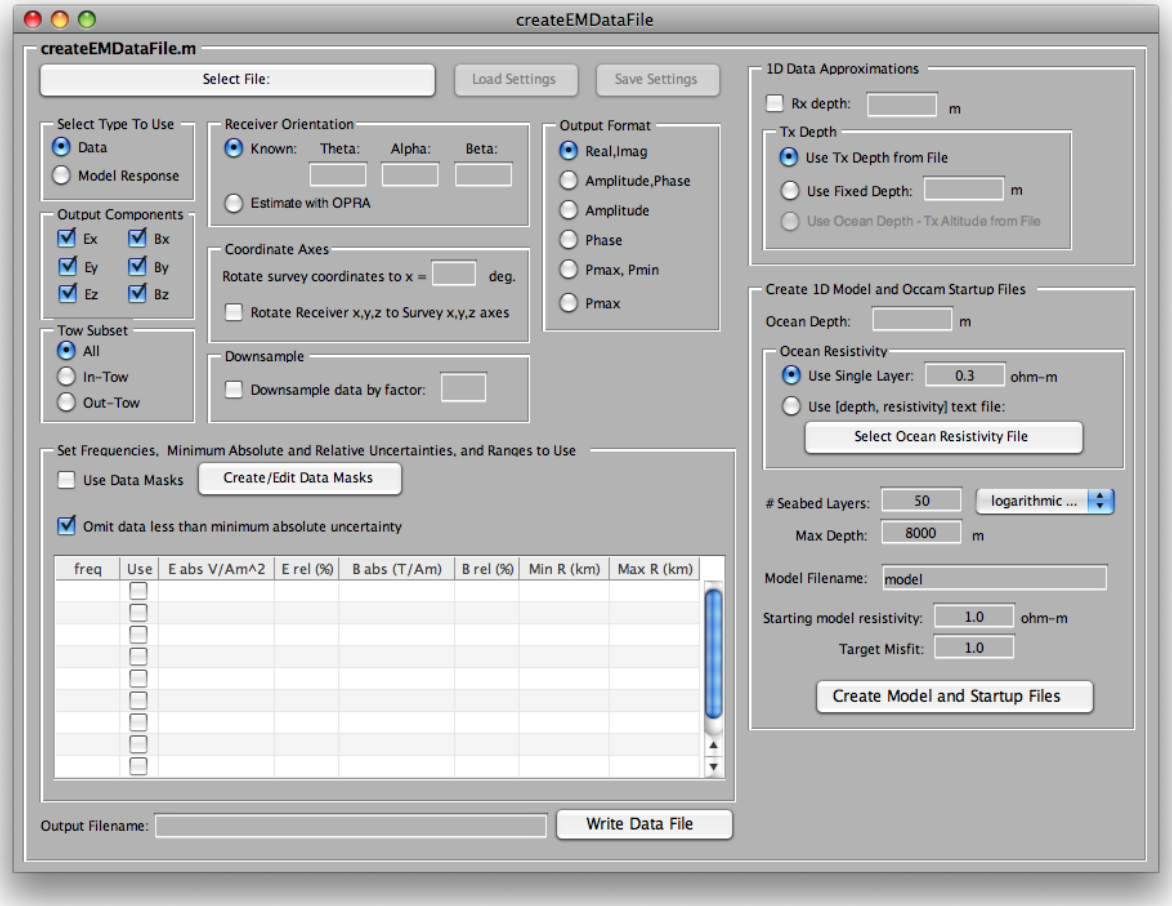


Figure 5: The Matlab interface for createEMDataFile.m.

Below are some instructions on what the various buttons do, listed by panel starting in the top left of the user interface.

Select File

Select a CSEM data file for editing. Note that createEMDataFile.m currently only supports single site data, so if you give it a file with more than one receiver, it will ask you to select which receiver to use.

Select Type To Use

Allows you to select which data type to output. Normally you don't need to do anything and will just use the default value Data. However, if you want to do synthetic inversions of forward model responses, then select the Model Response option. For this case, the model responses will have random normally distributed noise added to them (according to the relative levels specified in the Uncertainty table) before they are written to the output Data File.

Output Components

Select which components to output. Note that the x,y,z channels are relative to the instrument reference frame, which is not necessarily the same as the survey reference frame.

Tow Subset

This option allows you to output only a subset of the CSEM tow, either all data or only the in-tow or out-tow data.

Receiver Orientation

This option allows you to specify the orientation of the receiver using the three angles shown in Figure 3. Or you can select the Estimate with OPRA button if you want to use the orientation estimation method described in section 5.3.2 . Note that if you select OPRA, then all components of the data will be automatically selected for output, and the Output Format will be automatically selected as Real,Imag. If you don't want either of the E's or B's to be used with OPRA, then deselect them. Just make sure that you include all x,y (and possibly z) components of whatever fields you're using with the OPRA method.

Coordinate Axes

You can enter an angle to rotate the horizontal survey coordinate axes to (defined clockwise from x towards y). Or just leave it blank to skip any rotation of the survey axes. You can also select the Rotate Receiver checkbox if you want the receiver to be rotated so that its x,y,z channels are aligned with the survey x,y,z channels (this option is disabled if OPRA selected).

Downsample

If you have lots of data the inversion can take a long time to run. Often you may want a quick inversion run while you're investigating the noise levels and other features of the data. This option allows you to only output every nth data point. For example if you enter the number 5 in this box, only data from every 5th transmitter position will be output in the data file. Leave this blank to not downsample the data. Note that this does not stack nor average the data.

Output Format

This option allows you to select the output format for the data as one of Real and Imaginary, Amplitude and Phase, Amplitude only, Phase only, Polarization ellipse Maximum and Minimum (in the horizontal plane) or, Polarization Maximum only.

Set Frequencies, Minimum Absolute and Relative Uncertainties, and Ranges to Use

Use Data Masks

Select this option to use a mask file to denote good and bad data. Click the Create/Edit Data Masks button to launch an interactive tool for selecting the good and bad data by pointing and clicking in plot of the CSEM response for each frequency. Figure 6 shows an example of the data mask edit figure. The figure shows the electric and magnetic field responses for a given frequency. Deselected data is shown in gray and will not be included in the output data file, while selected data is shown in color. Buttons along the top of the figure allow you to select or deselect data, or to move on to the next frequency. Click on Select All and Deselect All buttons to select or deselect all data in a given plot. After hitting one of these, just click in inside either of the electric or magnetic field plots to select or deselect all data for that field (note that this work simultaneously on both the amplitude and phases, regardless of which axes you click inside). The Select+ and Deselect- buttons allow you to select discrete positions of data to omit, based on the position of the transmitter. After hitting one of these buttons click once to select the left or right side of the selected region, then click again on the right or left side of the selected region and then the selected/deselected data on the plot will be updated. Hit the Next or Previous buttons to scroll through the available frequencies in the data file. When you're done selecting data, hit the Quit and Save button. The mask file is then saved to <filename>.masks, where <filename> is replaced with the name of the data file. Note that when you click on Create/Edit data masks, the code looks for this file and will load it in and use it if it exists.

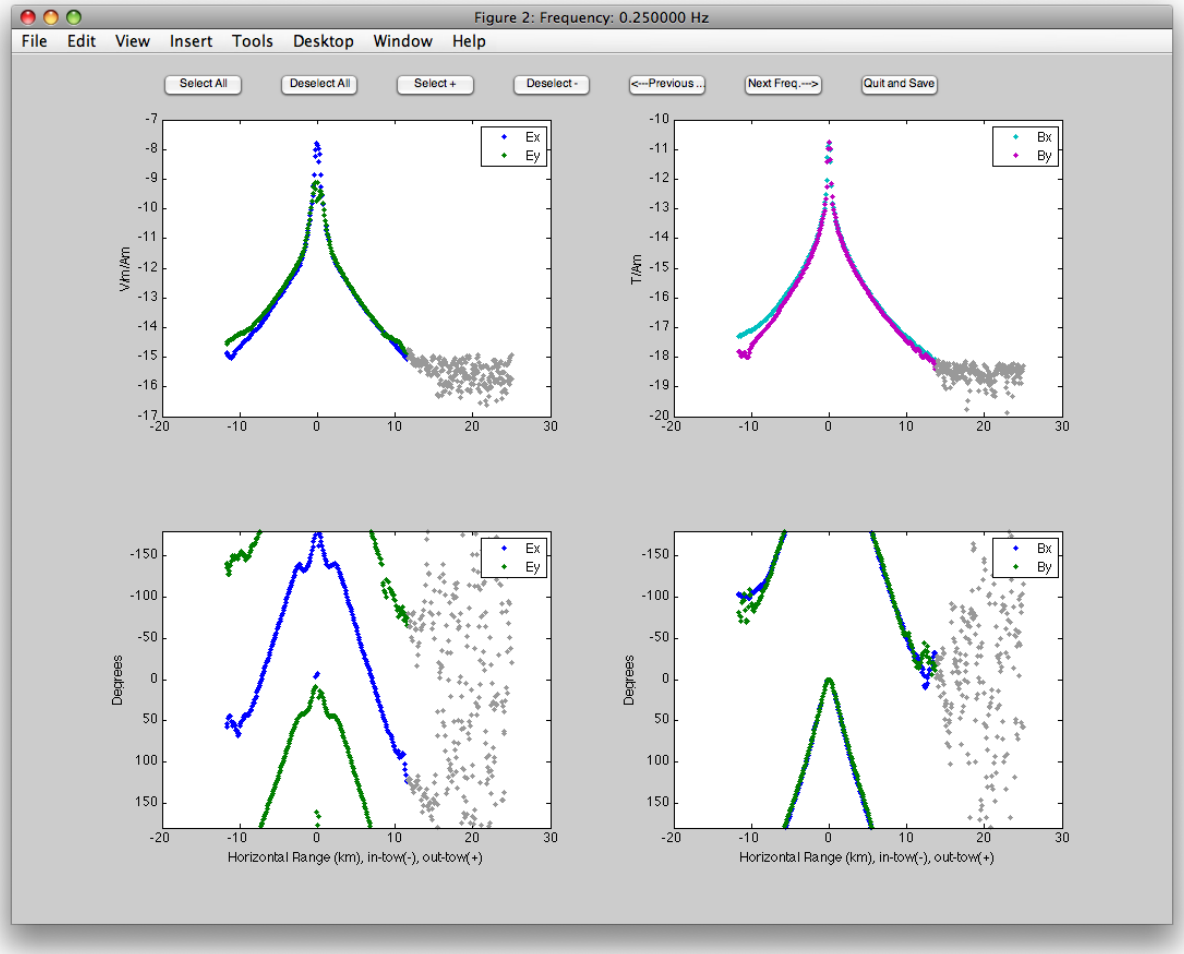


Figure 6: Create/Edit Data Mask Figure Example.

Omit data less than minimum uncertainty

This option will omit data with amplitudes smaller than those given in the Uncertainty table.

Uncertainty table

This table allows you to select the frequencies to output, to set the minimum relative and absolute noise levels for each field and to set limits on the minimum and maximum source-receiver ranges to use. You can enter in values for each frequency, or if you want the same values for each frequency, just enter some numbers in the first row (as shown in Figure 7) and the code will use this for all the frequencies.

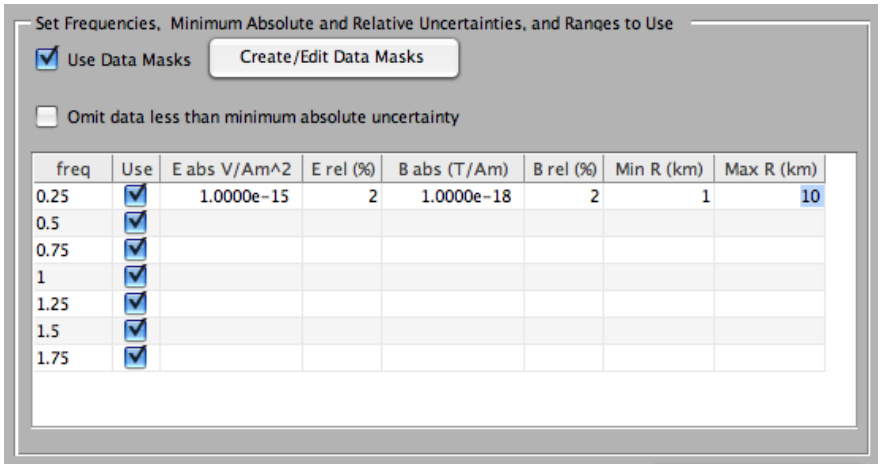


Figure 7: The Uncertainty table allows you to select frequencies, set noise levels and range bounds on the output data.

Output Filename

Allows you to specify the name of the output data file. By default this is set to <filename>.emdata, <filename> is the name of the data file you selected at the start. If you want a different name, just change it here.

Write Data File

Hit this button and the code writes out the data file using all the parameters you've specific in the user interface. A little message box will pop up letting you know its done outputting the file. If you have a big data file, this can take a few seconds.

1D Data Approximations

This panel allows you to set some parameters that are useful for 1D modeling approximations. For example, you may want to change the transmitter and receiver depths from the values given in the original data file when the survey line has lots of topography (which isn't modeled in 1D).

Receiver Depth

By default the receiver depth in the original data file is displayed, but you could change it here if you need to.

Transmitter Depth

If you're going to do 2D/3D modeling with bathymetry, then you probably want the Use Tx Depth from File button selected. For 1D modeling, you will need to select a Fixed Depth to use (so that the transmitter depth doesn't go below the 1D seafloor when the survey line has lots of regional slope!). I'm planning on having another option here that might be useful for 1D approximations (Use Ocean Depth - Tx Altitude), but the altitude measurements i've seen in the raw data files can be quite unreliable at times, and so this option is disabled for now.

Create 1D Model and Occam Startup Files

This panel has parameters that allow you to create the Model and Startup files easily.

Ocean Depth

Enter the depth of the ocean. By default this is set to be the same as the Rx depth, but you could change it if you have a towed receiver or a borehole receiver.

Ocean Resistivity

You can enter a single value for a uniform seawater layer, but you'll probably be better off using CTD or XBT data to use a more accurate stratified seawater resistivity (mainly from thermal layering). Use the Select Ocean Resistivity File button to read in a two column file of [depth (m), resistivity (ohm-m)] for the seawater.

 # Seabed Layers

Number of model layers to use for the seafloor. The popup menu allows you to select uniform or logarithmically increasing layer thicknesses. I usually use around 20-100 layers and set a maximum depth of less than 8 km.

Max Depth

Maximum depth of the seabed layering (m).

Model Filename

Name of the model file. Default is "model".

Starting model resistivity

The resistivity value for starting Occam with a uniform halfspace. Use units of linear ohm-m and the code will do the log10 conversion for you.

Target misfit

The target misfit for Occam.

Create Model and Startup Files

Hit this button to write out the Model and Startup Files. If you done this and written the data file, then you're ready to run OCCAM1DCSEM.

8.2 Plotting inversion models with plotOccam1DCSEM.m

plotOccam1DCSEM.m is a Matlab code for plotting the inversion models output by OCCAM1DCSEM. Make sure you change Matlab's Current Directory to the directory where you ran the inversion. Then use one of these commands:

```
>> plotOccam1DCSEM   % asks you to select the iteration file to plot

>> plotOccam1DCSEM('MyIterationFile.iter') % plots MyIterationFile.iter

>> plotOccam1DCSEM('lastiter') % plots the most recent iteration in the
                                 current directory

>> plotOccam1DCSEM('alliter') % plots all iterations in the current
                               directory

>> plotOccam1DCSEM({myiters}) % plots all iteration files in the cell
                               array 'myiters'
```

Figure 8 shows some example plots made with plotOccam1DCSEM.m. If you don't like repeatedly typing this rather lengthy name into the Matlab command line, I recommend that you set this command as a shortcut button on the Matlab tool bar.

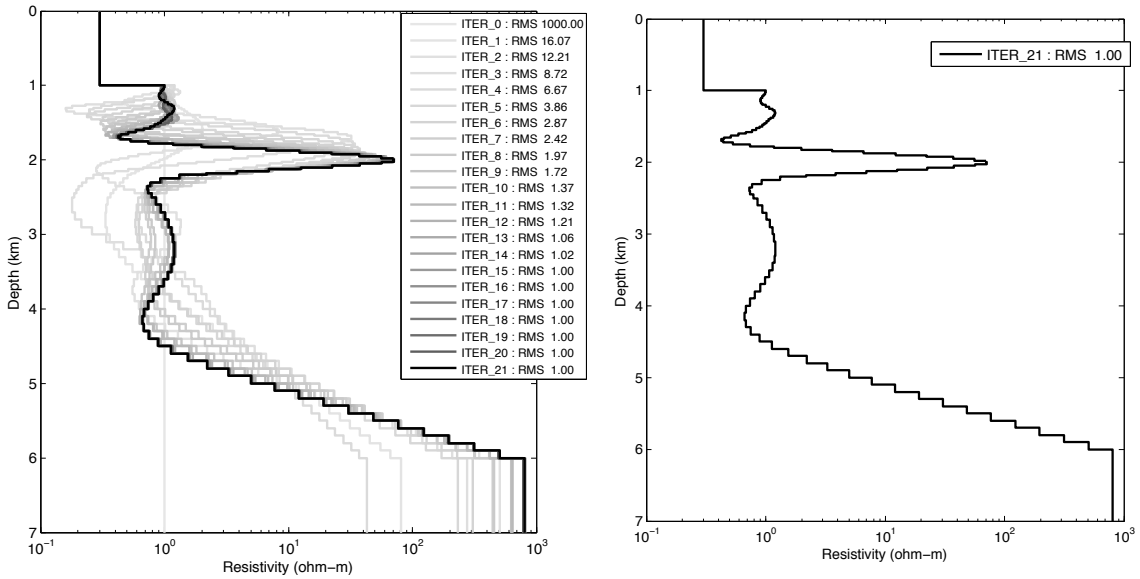


Figure 8: plotOccam1DCSEM.m examples. The figure on the left is generated using "plotOccam1DCSEM('alliter') " and shows the 21 iterations for that inversion. The iterations are colored in grayscale with the highest (and presumably best) iteration plotted in black. The figure on the right is generated using "plotOccam1DCSEM('lastiter')" or "plotOccam1DCSEM('ITER_21.iter')" and just contains the inversion model for a single iteration.

8.3 Plotting the behavior of Occam's inversion with plotOccamIterMisfit.m

This command plots up parameters that describe the search performance of the Occam inversion iterations. This routine will probably not be used much, but it is useful if the inversion fails or for studying the rare case where it seems unstable.

Usage:

```
>> plotOccamIterMisfit  % reads in all iteration  files in the current
% directory and plots misfit and roughness versus iteration, also
% plots search data recorded in the .logfile.
```

Figure 9 shows two example plots made with plotOccamIterMisfit.m.

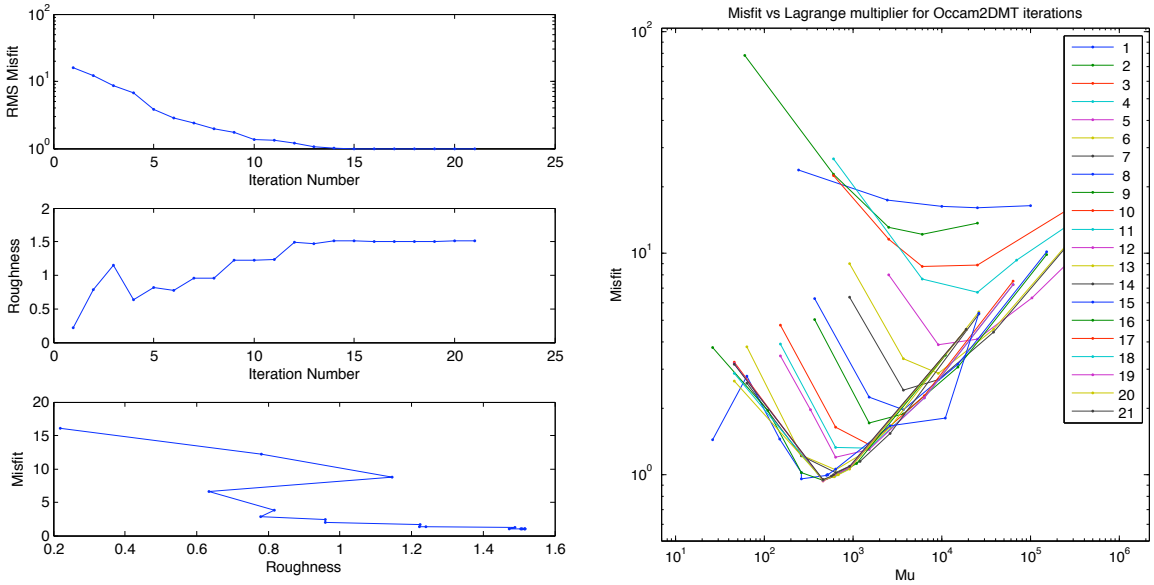


Figure 9: plotOccamIterMisfit.m examples. The left figure shows the RMS misfit and roughness versus iteration number, and versus each other. The right figure shows the misfit curve as a function of Lagrange multiplier μ for each inversion iteration (numbered in the legend).

8.4 Plotting data and model responses with plotCSEM.m

OCCAM1DCSEM comes with version 3.0 of the data plotting routine plotCSEM.m. This routine originated with the MARE2DCSEM modeling code and has been updated here to handle plotting survey data, model responses, data uncertainties and model fit residuals.

Usage:

```
>> plotCSEM       % creates an interactive menu for selecting data and
                  % plotting options. The rest should be obvious.
```

The interactive plotting menu used by plotCSEM.m is shown in Figure 10. The following subsections describe the plotting options available with the various menus and buttons.

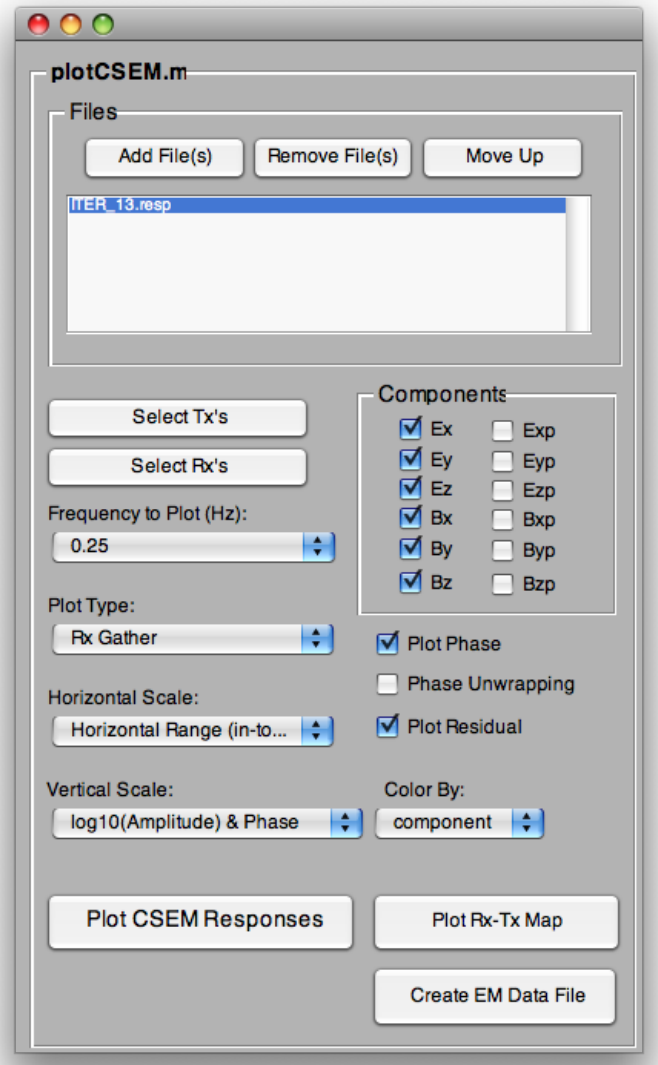


Figure 10: plotCSEM.m menu figure showing the plotting options available.

Add, Remove, Move Files

In this section you can add .csem, .resp and .data files for plotting. plotCSEM currently supports reading these data formats:

EMData_1.1,EMData_1.2	Occam1DCSEM data file format
EMResp_1.1,EMResp_1.2	Occam1DCSEM model response file format
Dipole1D_1.0,Dipole1D_1.1	DIPOLE1D forward model response file format
MARE2DCSEM_2.0	MARE2DCSEM finite element forward modeling code response file format.

Frequency to Plot

Once you have some files added, this popup menu allows you to select the frequency to plot.

Components

Check the boxes for the components to plot (if they are available in the data files). The components ending with p, such as Eyp, are the primary fields computed for MARE2DCSEM models and aren't used for OCCAM1DCSEM plotting.

Select Tx's, Select Rx's

These buttons launch listboxes that allows you to select which transmitters and receivers to plot, such as shown in Figure 11. This is mainly used for only plotting subsets of Tx's and Rx's computed using MARE2DCSEM.

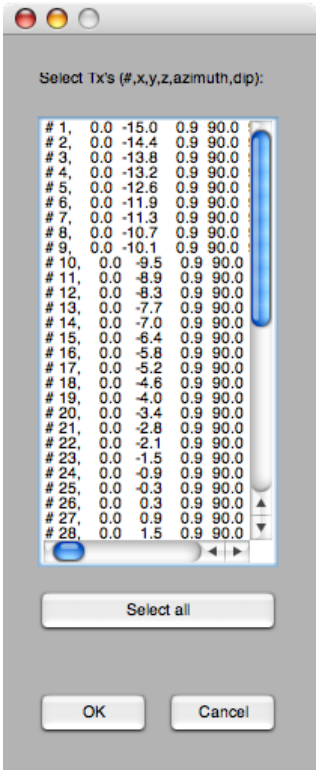


Figure 11: Transmitter selection list dialog.

Plot Type

There are three plotting types:

- Tx Gather: Data from each transmitter is plotted at the position of each receiver.
- Rx Gather: Data for each receiver is plotted at the position of each transmitter. This is what survey data recorded on a receiver are and hence is the type of plot commonly used with OCCAM1DCSEM modeling. If the data file only contains a single receiver, then the plot type "Rx Gather" is automatically selected.

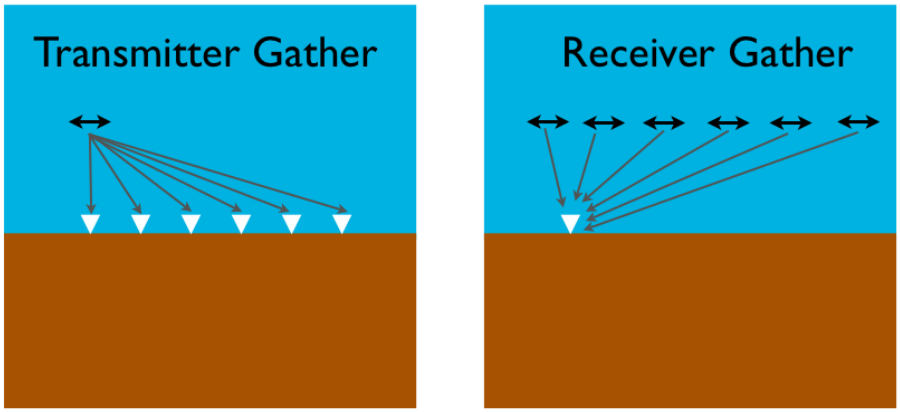


Figure 12: Transmitter and receiver gathers.

- Pseudosections: Data from many transmitters and receivers are plotted at the source-reciever mid-point for the horizontal scale, and the signed source-receiver offset for the vertical scale. This currently only works for plotting MARE2DCSEM model responses.

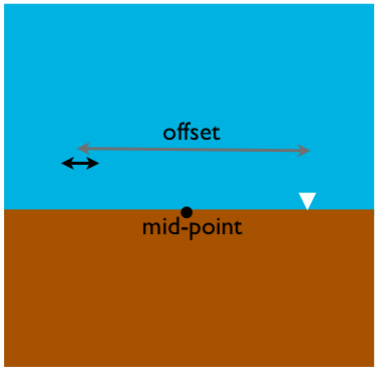


Figure 13: Mid-point and offset for pseudosection plots.

Horizontal Scale

Select the scale to use for the horizontal plotting axis. Choose from x, y, z, Range (x,y,z), Horizontal Range (x,y), Horizontal Range (signed as InTow, OutTow), Vertical Range and Rx-Tx Azimuth (useful for circular tows for anisotropy).

Vertical Scale

Set the vertical plotting scale. Choose from

- log10 Amplitude and Phase.
- Real/Imaginary.
- Polarization Min/Max (horizontal only for now...)
- Ratio: 2D/primary. Only for MARE2DCSEM: Note that the primary response corresponds to the 1D primary model given in the RUNFILE.
- Relative Difference: (2D-primary)/primary. Only for MARE2DCSEM.

If two files have been added to the file list then you can create anomaly plot comparisons using these two options:

- Ratio: File 1 / File 2. Only for MARE2DCSEM.
- Relative Difference: (File 1-File 2)/File 2. Only for MARE2DCSEM.

Plot Phase

Checkbox for plotting phase (in addition to amplitude).

Phase Unwrapping

Checkbox for performing phase unwrapping. Large phase jumps will be unwrapped so that phases can extend outside the $\pm 180^\circ$ window. This mostly works, but can sometimes produce the wrong result for sparsely sampled or noisy data.

Plot Residual

Checkbox for selecting to plot the model response residuals (see equation 16) for OCCAM1DCSEM inversions.

Color By

Checkbox for selecting whether the data and model responses are colored by field component or file number. It is useful to color by file if you want to compare the data fits for various OCCAM1DCSEM iteration response files. If you're only plotting a single response file but which has many field components (for example B_x, E_y and E_z), then it is useful to color by component.

Plot CSEM Responses

Once you've selected all the other plotting options, hit this button to create the plot.

Plot Rx-Tx Map

Hit this button to make a simple map plot showing the receiver and transmitter locations. The transmitters are plotted as short lines at the azimuth angles given in the data or response files. This can be useful for verifying that the transmitter and receiver geometry in the data file is correct.

Create EM Data File

Hit this button to launch the createEMDataFile.m user interface.

8.5 Example Data and Response Plots

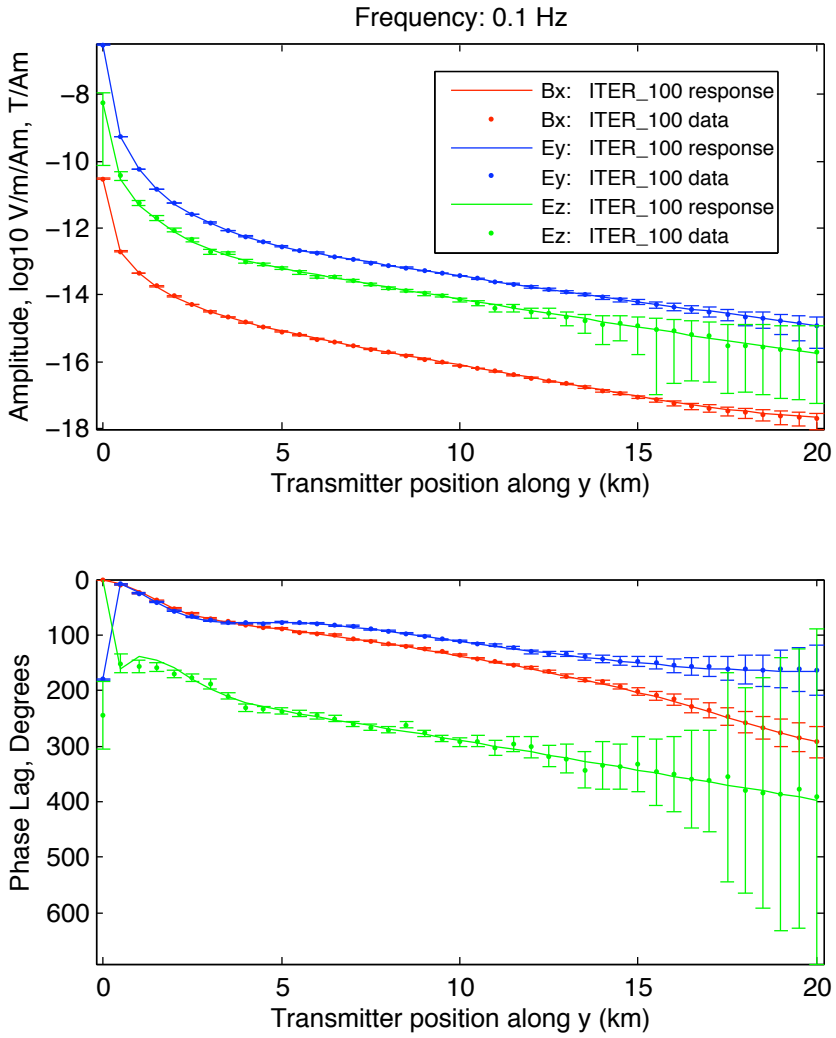


Figure 14: Example amplitude and phase plot made with plotCSEM.m. Data are plotted as dots with error bars and the model response for file ITER_100.iter is shown as lines.

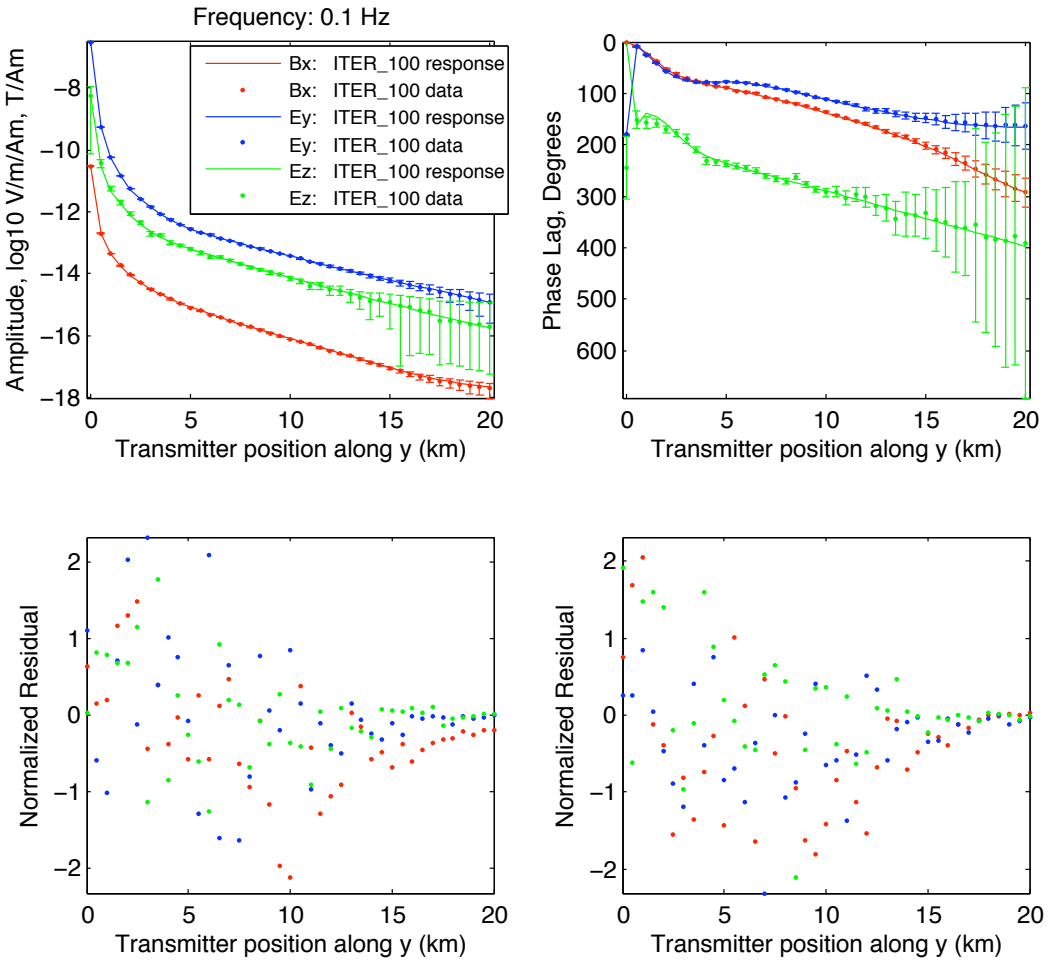


Figure 15: Example amplitude and phase plot with residuals, made with plotCSEM.m. The lower row shows the amplitude residual (left) and phase residual (right). A good fit to the data will have residuals that are largely uncorrelated.

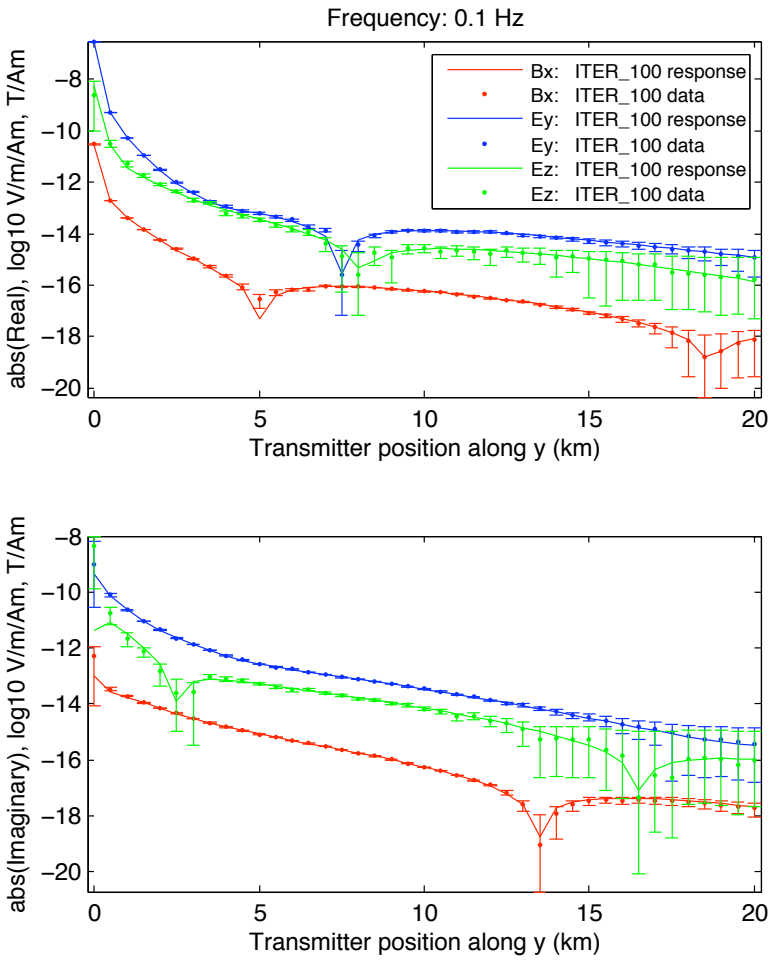


Figure 16: Example real and imaginary plot made with plotCSEM.m. The valleys in the responses are where the components have a zero crossing. Zero crossings in the real component denote where the phase crosses an odd integer multiple of 90°. Zero crossings in the imaginary component denote where the phase crosses an integer multiple of 180°.

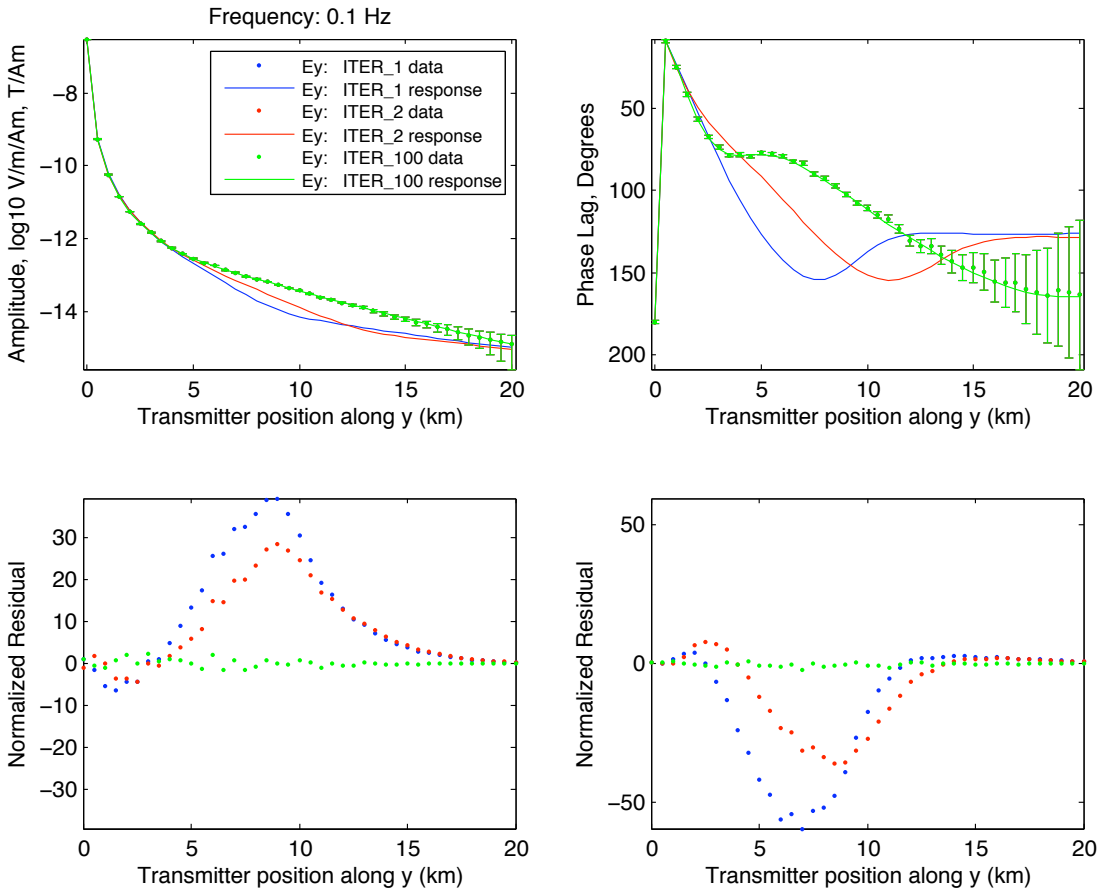


Figure 17: Example of coloring plots by file in plotCSEM.m. The three colors show the E_y responses for iterations 1, 2 and 100. Note the difference in the residual plots (iteration 100 fits to RMS 1.0).

9 OCCAM1DCSEM Examples

This section presents a few figures demonstrating results from the new OCCAM1DCSEM inversion. Some of these examples are present in the /Examples folder.

9.1 San Diego Trough CSEM Data

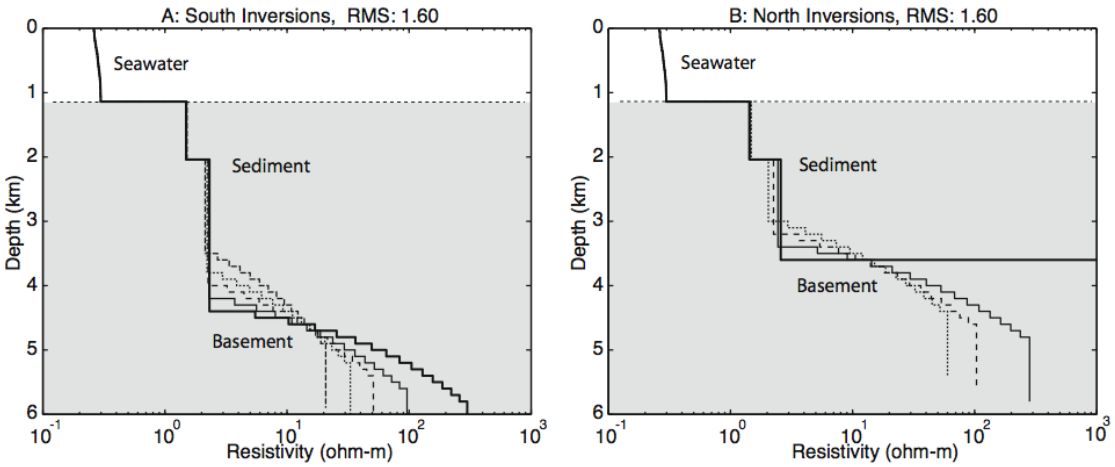


Figure 18: OCCAM1DCSEM inversion of CSEM survey data collected in the San Diego Trough for tows to the south (left) and north (right) of a long-wire EM receiver. The inversions included the stratified seawater resistivity as fixed structure and the shallow sediments where divided into two free layers based on seismic and gravity studies, and these were underlain by many thin layers for the basement. In order to study the sensitivity to the depth of the resistive basement, the thickness of the second sediment layer was varied between inversions (shown by the various lines). From Constable et al. (2009). The data used in these inversions are available in /Examples/SanDiegoTrough.

9.2 Component Resolution Tests on Synthetic Data

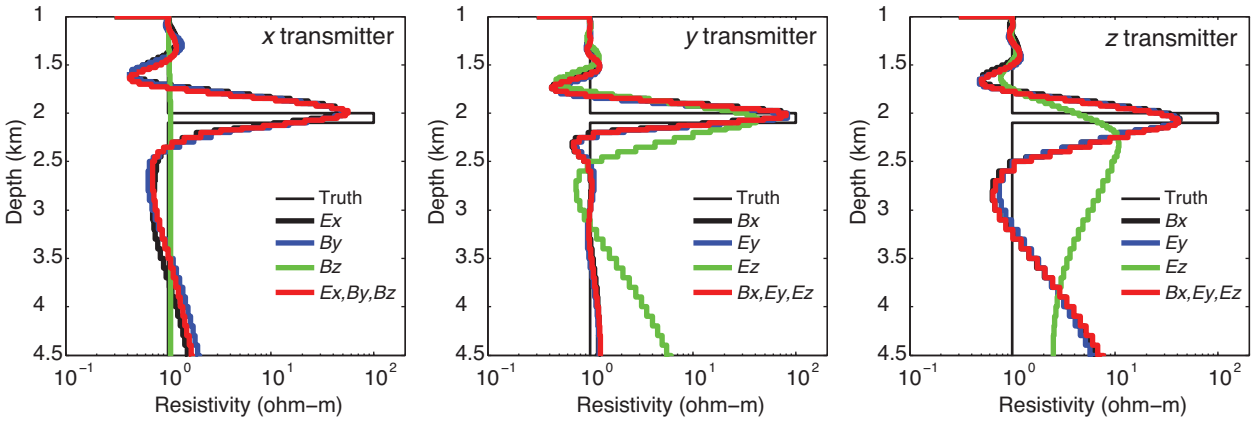


Figure 19: The resolution of the **E** and **B** field components for the three fundamental transmitter orientations (broadside x, inline y, and vertical z) for a line of receivers along the y axis. Synthetic inversion models are shown for inversion of each component separately and all three together, as indicated in the legends. The transmission frequencies were 0.1 and 1.0 Hz, the data had 1% noise added and all models fit the data to RMS 1.0. From Key (2009).

9.3 Synthetic Data from Thinly Layered Sediments: Smooth and Penalty Cut Inversions

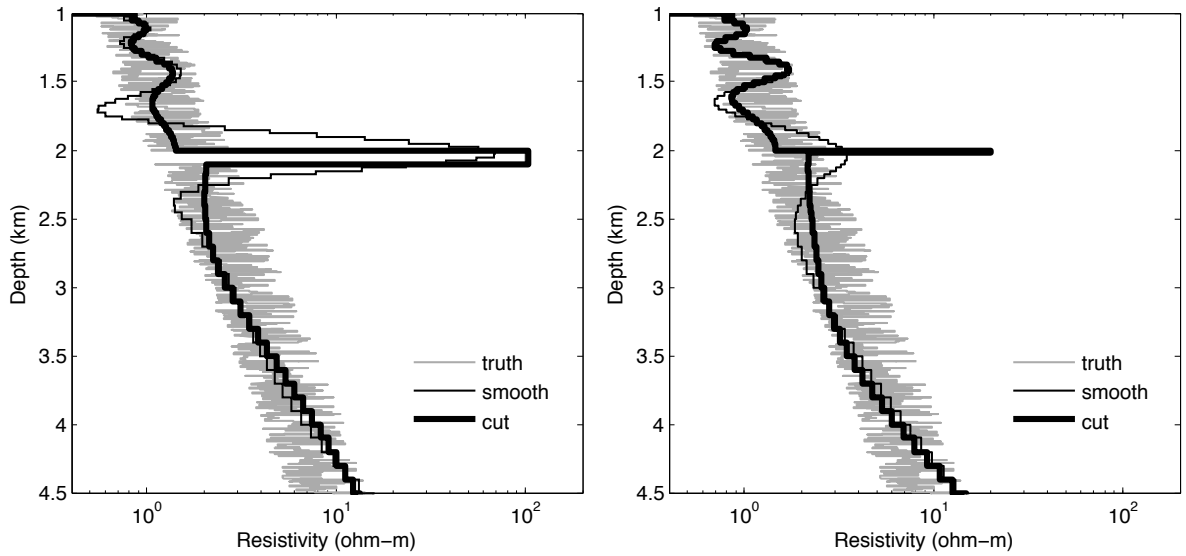


Figure 20: OCCAM1DCSEM inversion of synthetic CSEM data computed for the finely layered models shown by the gray lines. Both smooth inversion (all penalty weights set to 1) and penalty cut inversions (penalty weights set to 0 on reservoir boundaries) are shown. From Key (2009).

9.4 Minimum Gradient Support Roughness Penalty Examples

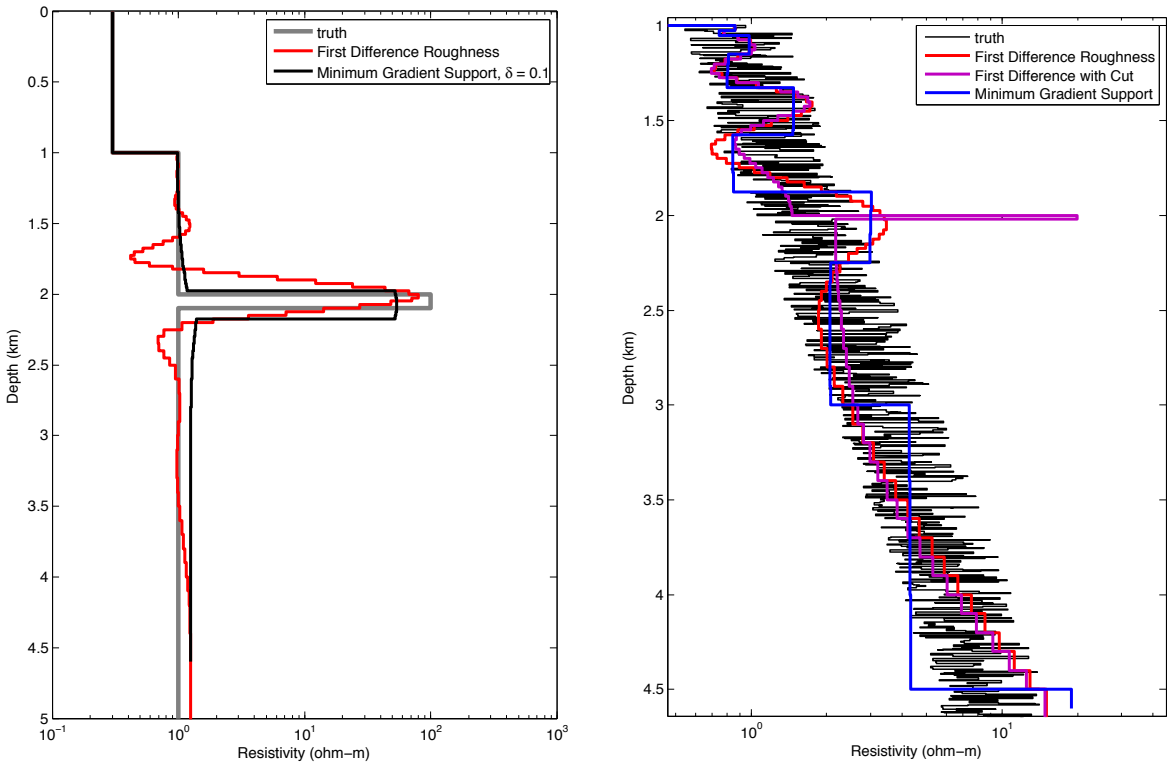


Figure 21: Examples of inversion models obtained using minimum gradient support roughness penalty compared with inversions using standard first-difference roughness. For the large resistive target in the canonical 1D model (left), both regularizations result in models that identify the presence of a large resistor, while the minimum gradient support inversion produces a more geologically pleasing discrete resistive layer. However, for much smaller targets such as the 10 m thick, 10 ohm-m reservoir in the model on the right, the minimum gradient support inversion is perhaps less pleasing than the smooth first difference model, while neither provides strong indication of the thin reservoir layer. In these hard target cases, adding structural information via penalty cuts can help isolate small resistive layers.

9.5 Joint Inversion of CSEM and MT Data

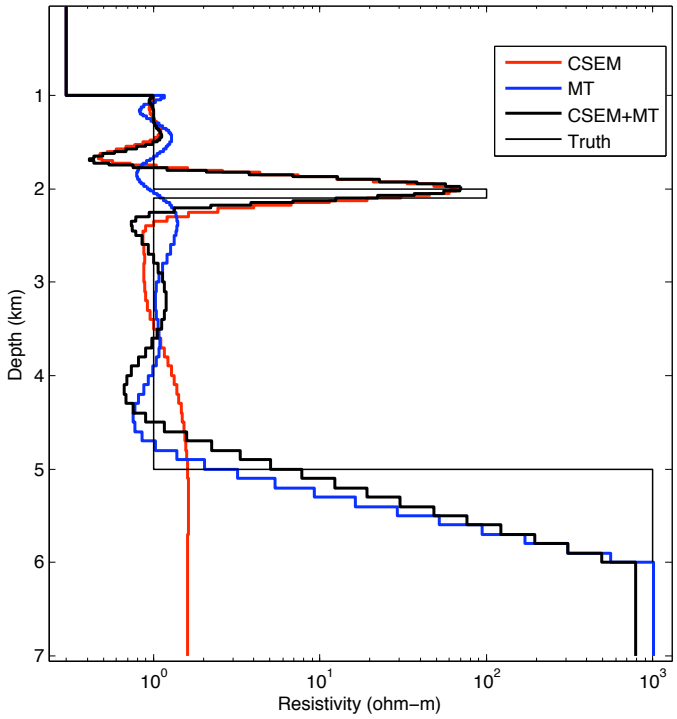


Figure 22: Synthetic inversions of CSEM and MT data inverted both independently and jointly, as indicated in the legend. The synthetic CSEM data consisted of 0.1 and 1.0 Hz with 2% noise added, while the MT data used 21 frequencies from 0.0001 to 1 Hz. The CSEM data are sensitive to the sediments and reservoir while the MT data are sensitive to the sediments and basement. The joint inversion is able to recover both the reservoir and the basement. These models and data are available in /Examples/CanonicalBasement_CSEM_MT

10 Troubleshooting

10.1 I removed the roughness penalty on some of the layers and now it won't converge to a reasonable misfit.

Removing the roughness penalty can sometimes destabilize the inversion (hence the reason the roughness penalty is also called a stabilizer and is heavily relied upon for EM inversion). If you find the inversion has trouble with this, try adding a preference resistivity value for the layers with the cut roughness penalty, and make sure you include a non-zero preference penalty value (try something in the range of 0.01 to 1). You can also stabilize the inversion by setting values for the "Model Limits" field in the Occam startup file. For example,"Model Limits: -1,4" will force the resistivity to stay between 0.1 and 10,000 ohm-m.

10.2 My inversion model looks awful!

Most likely this is a case of "garbage in = garbage out". First check that you have the correct phase convention set for your data. For example, compute the forward response for a likely model and check that phases are either increasing or decreasing in accordance with the data. If not, use the opposite phase convention by specifying the convention in the data file (see section 5.3.1). The next thing to check is that you have the right receiver and transmitter geometries and rotations. Finally, you should visually inspect the data to ensure that it is free of extended periods of noise-floor level data. Also make sure that there are no strong outliers in the data. For example, single data points ten times off the main trend can wreak havoc since it will contribute 100x to the L2 norm. Search the Occam1DCSEM residuals in the .resp files for any large outliers. If you find some, either remove the corresponding data or greatly increase its uncertainty level. You may need to iterate on this for a few inversion runs.

References

Constable, S., K. Key, and L. Lewis, 2009, Mapping offshore sedimentary structure using electromagnetic methods and terrain effects in marine magnetotelluric data: Geophysical Journal International, **176**, 431–442.

Constable, S. C., R. L. Parker, and C. G. Constable, 1987, Occam's inversion - A practical algorithm for generating smooth models from electromagnetic sounding data: Geophysics, **52**, 289–300.

deGroot-Hedlin, C. and S. Constable, 1990, Occam's inversion to generate smooth two-dimensional models from magnetotelluric data: Geophysics, **55**, 1613–1624.

Key, K., 2009, 1D inversion of multicomponent, multifrequency marine CSEM data: Methodology and synthetic studies for resolving thin resistive layers: Geophysics, **74**, F9–F20.

Portniaguine, O. and M. S. Zhdanov, 1999, Focusing geophysical inversion images: Geophysics, **64**, 874–887.

van den Berg, P. M. and A. Abubakar, 2001, Contrast source inversion method: State of art: Progress in Electromagnetic Research, **34**, 189–218.