INCOMPLETE DRAFT FOR BETA TESTING ONLY

User Manual for Geoscience Australia's Airborne Electromagnetic Inversion Software

Version 1.0, July 2015

Dr Ross. Colin. Brodie
Geoscience Australia

Australian Government
Geoscience Australia

APPLYING GEOSCIENCE TO AUSTRALIA'S
MOST IMPORTANT CHALLENGES

Contents

1 Introduction

1.1 Background

1.2 Online code repository

1.3 Using the pre-compiled WINDOWS executables

1.3.1 Parallelisation

The source code is parallelised in via two different methods. The first is the Message Passing Interface (MPI) and the second is OpenMP (Open Multi-Processing). Unfortunately they sound similar and are often confused, but they are distinctly different beasts.

The forward modelling program (§5) is not parallelised at all, the deterministic inversion program (§6) is parallelised via MPI or OpenMP, and the stochastic inversion (§7) program is parallelised via MPI only at this stage. The parallelisation is at a coarse level in which each sample is simply inverted by a different process.

OpenMP is a programming construct that tells a compiler which part of the program should be multithreaded. Just one instance of the program is executed. OpenMP requires a shared-memory computer, like a typical multicore desktop PC, where all the processes access the same memory address.

MPI allows physically different computers to send messages to each other over a network to implement parallelism. Those computers can potentially be on opposite sides of the world. Many instances of the program are executed by a daemon (usually called mpirun or mpiexec) then the programs communicate across the network to exchange data where necessary. MPI can be used on a typical multicore desktop PC, or on a massively parallel distributed-memory cluster computer.

To use OpenMP parallelism you will not need to install any additional software or daemons. This makes it the easiest option for most users who only have a standalone multicore PC to invert their data on. However to make use of MPI you will need to install an implementation of MPI on their system which will include the mpirun/mpiexec daemons and shared libraries. While this is messier, it potentially allows the more adventurous to make use of all the computers in their office.

The code repository includes two pre-compiled 64 bit WINDOWS executable for the deterministic inversions program in the directory ***./bin/x64/Release***. To use ***galeisbstdem.exe*** you will need to install the "Microsoft HPC Pack 2012" (see §1.3.4), which was the MPI implementation that the program was built with. This version allows you to use either MPI or OpenMP parallelism. The second executable which is called, ***galeisbstdem-nompi.exe***, can be used if your system administrator really does not want to be troubled installing "Microsoft HPC Pack 2012". This executable does not require "Microsoft HPC Pack 2012" to be installed. However you can still use

OpenMP parallelism. The forward modelling program does not require MPI but the stochastic inversion program does at this stage.

1.3.2 Microsoft Visual Studio 2013 DLLs

To run the pre-compiled WINDOWS executables you will most likely need to download and install Microsoft's free "Visual C++ Redistributable Packages for Visual Studio 2013" which contains the dynamic link library (.dll) files required by the programs. The package is available from http://www.microsoft.com/en-au/download/details.aspx?id=40784. The required libraries (***msvcp120.dll***, ***msvcr120.dll***, ***vcomp120.dll***) may already have been installed in your ***c:\windows\system32*** (or equivalent) system directory by a different software package, in which case this step will not be necessary. The package does not install the compiler itself, but just the necessary libraries. You do not need to install a compiler unless you prefer to recompile the source code.

1.3.3 FFTW

All the programs require the use of the open source FFTW (http://www.fftw.org) package for computation of the Fast Fourier Transforms (FFTs). The WINDOWS libraries have been redistributed in the source code repository. The required dynamic link library (DLL) ***libfftw3-3.dll*** can be found in the directory ***.\third_party\fftw3.2.2.dlls***. That directory needs to be in your WINDOWS search path when you execute programs.

1.3.4 MPI

To run the pre-compiled MPI enabled executables you will most likely need to download and install the "Microsoft HPC Pack 2012" implementation of MPI which the programs have been built with. The package is freely available from http://www.microsoft.com/en-au/download/details.aspx?id=36054.

If you definitely do not want to bother installing MPI on your WINDOWS system then there is a companion version of galeisbstdem.exe, called galeisbstdem-nompi.exe in the WINDOWS executable directory (***./bin/x64/Release***) which has been compiled without the MPI library. This executable will run without MPI having been installed on your system and it can make use of OpenMP parallelism instead.

1.3.5 Program execution

The programs are executed from the command line in the forms,

```
>> program_name control_file_name
>> program_name control_file_name number_of_openmp_threads
>> mpirun –np number_of_processors program_name  control_file_name
>> mpiexec –np number_of_processors program_name control_file_nam
```

1.3.6 Search paths

Unless you include the full pathname of the executable when running a program from the command line, WINDOWS batch file or Linux shell script, you will need to make sure that the executable file and any run time dependencies (e.g., ***.dll*** dynamic link libraries on WINDOWS or ***.so*** shared libraries on Linux) are in your search path. This dependencies will generally include the FFTW and MPI libraries.

To do this on WINDOWS you can search for "Edit environment variables for your account" in Control Panel and (create and) edit the PATH variable to include the directory path to the location where you have installed the executable files and dynamic link libraries.

2 Block language files

2.1 Structure

Block language files are ASCII text files that are used as program control files for the user to specify: input, output and log files; column numbers; system specification files; reference models and weights; and various other program options. They a similar to header and job control files used in the ER Mapper™ and Intrepid™ software packages. Block language files have the following general structure.

```
BlockName Begin
  //This is a comment line
  KeyWord1 = Value1
  KeyWord2 = Value2
  BlockNameA Begin
    KeyWord1 = Value1
    KeyWord2 = Value2
    …
    BlockNameX Begin
        ArrayValue11 ArrayaValue12
        ArrayValue21 ArrayaValue22
        …
    BlockNameX End
  BlockNameA End

  BlockNameB Begin
    KeyWord1 = Value1
    …
  BlockNameB End
  …
BlockName End
```

Examples of block language files used for AEM system specifications are shown in Appendix A, Appendix B, Appendix C, and Appendix D. Examples of block language files used for program control are shown in Appendix E.

2.2 Parsing rules

There are certain rules that must be observed with block language files.

1. Every ***BlockName Begin*** token must be matched with a corresponding ***BlockName End*** token.
2. Blocks can be nested in a hierarchical structure. The correct hierarchical structure must be observed.
3. The ordering of blocks and keywords is not important, but they must maintain the correct hierarchical structure.
4. Block names and keywords on the left of the equals signs ("=") must be spelt exactly as required by the programs.

5. Comparisons between block names, keywords and values are not case insensitive. Therefore it is permissible to use ***TX_Height = Column 50*** or ***tx_Height = coLumn 50***.
6. File and directory path names are case sensitive on a Linux system. Hence, irrespective of the case insensitivity outlined in the previous rule, it must be recognised that file and directory path names are case sensitive on a Linux system and must be provided in the correct case on Linux (see §2.5).
7. Leading and trailing whitespace (tabs and spaces) are trimmed from block names, keywords and values before comparisons are made. Therefore indentation and spaces are allowed to improve readability.
8. Blank lines are ignored.
9. Comments can be inserted or valid lines may temporarily be disabled by inserting "//" in front of the block name or keyword. To be totally truthful any unrecognised block name or keyword is simply ignored.
10. Comments or other unrecognised tokens must not be placed inside blocks where a consecutive array of numbers is expected. For example the following block would cause a program crash.

```
  WaveFormCurrent Begin
    //This comment will most likely crash the program
    -0.0200000000000    0.0
    -0.0199933333333    1.0
    -0.0000066666667    1.0
     0.0000000000000    0.0
     0.0000066666667   -1.0
     0.0199933333333   -1.0
     0.0200000000000    0.0
  WaveFormCurrent End
```

11. Boolean values can be specified with either ***yes, no, true, false, 0, or 1***. For example ***SolveRX_Pitch = yes*** will produce the same result as ***SolveRX_Pitch = True.***
12. If a keyword is repeated in a block then only the first occurrence of that keyword will be recognised. For example, in the snippet below the programs will expect the transmitter roll data (***TX_Roll***) to be in column 18 rather than column 17 because ***TX_Roll = Column 18*** appears first.

```
Columns Begin
  …
  //TX_Roll  = Column 19 // This TX_Roll will be ignored
  TX_Roll    = Column 18
  TX_Roll    = Column 17 //This TX_Roll will be ignored
  TX_Pitch   = Column 20
  …
Columns End
```

2.3 Specifying input values

In many cases it is possible to invert data directly from ASCII files provided by a contractor without any reformatting. This is made possible by certain features of the program. Depending on the circumstances, input information (electromagnetic data, noise estimates, system geometry or model weights) may be input in the following three forms,

```
KeyWord = Column C
KeyWord = -Column C
KeyWord = constant_value
```

For example the programs require the transmitter pitch to be specified in degrees with an 'aircraft-nose-down is positive' sign convention. If a transmitter pitch, in degrees is located in column 18 of the input data file and it conforms to the same coordinate system sign polarity as the modelling and inversion programs (§3.1), then the first form ***TX_Pitch = Column 18*** may be used.

However, if the transmitter pitch was in the opposite sign polarity to the inversion program coordinate system, the second form ***TX_Pitch = -Column 18*** is a convenient shortcut. This form passes the negative of column 18 to the program so that the input data file does not have to be reformatted. Do not try to use ***TX_Pitch = Column -18.*** The third form is often used to specify the transmitter pitch in inversion of TEMPEST data because the data are standardly provided with a 'aircraft-nose-up is positive' sign convention.

If the transmitter pitch is assumed to be a constant, say zero, for every sounding in the dataset (perhaps the pitch is not measured or the input data has been processed to a standard geometry), the third form, ***TX_Pitch = 0*** is appropriate and convenient.

Note that the first column in an input file is "Column 1" (not Column 0).

2.4 Specifying multiple input values

When it is necessary to specify ***N*** related inputs, such as layer conductivities, thicknesses, or electromagnetic window data, it is possible to use one of the four forms below.

```
KeyWord = Column C
KeyWord = -Column C
KeyWord = constant_1 constant_2 constant_3 … … constant_N
KeyWord = constant_for_whole_array
```

For example, if there are 45 Z-component windows located in columns 177 through to column 221 of the input file. The first form could be used to specify the location of Z-component data as shown below.

```
ZComponentSecondary = Column 177
```

Alternatively if the window data in the file where in the opposite sign polarity to that expected by the programs, this second form would be appropriate.

```
ZComponentSecondary = -Column 177
```

This second form is often used to specify Z-component data because window data are typically supplied as positive values, but the programs produce negative Z-component values when forward modelling the effect of a +Z-axis directed primary field being switched off. Note that the programs only support ordered columns, therefore in the first and second forms the values must be located in consecutive columns.

The third form is useful when it is necessary to specify an array of values that are not in the input file and are spatially constant, that is they do not need to change from sounding to sounding. For example one way to specify a spatially constant conductivity and thickness reference (and starting) model for a three layer deterministic inversion would be as follows:

```
ReferenceModel Begin
   Conductivity = 0.01 0.1 0.001
   Thickness    =   20   30
ReferenceModel End
```

The fourth form is a shorthand method of specifying a spatially constant array of values where all values in the array are equal. For example, in a 10 layer fixed-thickness inversion, we could specify the reference model standard deviation (or uncertainty) to be 3.0 log-decades by using this third form,

```
StdDevReferenceModel Begin
   Conductivity = 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
StdDevReferenceModel End
```

or more conveniently by using this fourth form.

```
StdDevReferenceModel Begin
   Conductivity = 3
StdDevReferenceModel End
```

If the program is expecting an array of ***N*** values it will first looks for the first or second forms. If these are not present it will check if ***N*** values have been supplied (third form) and use those as the array. If the correct number of values is not specified then the program will crash. Otherwise, if a single value is supplied (forth form), it uses that value in all elements of the array.

2.5 Specifying file and directory path names

As discussed in parsing rule 5 of Section 2.2, keyword and value comparisons are not case sensitive in the block language files. Irrespective of that case insensitivity, it must be recognised that, when specifying file and directory path names on a Linux system, the paths must be given in the correct case.. For example if you are working on a Linux system, and your AEM system file is named ***TempestFrome.stm***, do not use

`SystemFile = tempestfrome.stm`. However this would work on a Windows system.

In the block language files, path name separators (i.e., **"\\"** on Windows and **"/"** on Linux) can be used interchangeably because the code automatically translates the separators to the correct form before trying to create or open files. This allows a job setup on a local desktop WINDOWS PC to be copied to a Linux based cluster and run without any need to change path names in the control files.

It is possible, and often advisable, to use relative rather than absolute paths when specifying file or directory names in control files. Relative paths are relative to the directory in which the program was executed. For example, if the program was launched from ***D:\frome\inversion\run.01***, then is permissible to specify an AEM system file using either ***`SystemFile = D:\frome\stmfiles\Tempest.stm`*** or alternatively ***`SystemFile = ../../stmfiles/Tempest.stm`***.

3 AEM system specification (STM) files

3.1 Introduction

The AEM system specification files are used by all the forward modelling and inversion programs as a means of specifying the details of the AEM data that is being modelled. It does not particularly describe the physical AEM system itself, but rather the system that would produce the equivalent data to be inverted after having undergone the various processing and normalisations.

Conventionally, and throughout this document, we will call the AEM system specification files "*STM*" files. We also usually give them the filename extension ***.stm***, although that extension is not mandatory, or an extension even required. STM files are in the block language structure as discussed in Section 2. Each of the blocks and their keyword meanings are described in the remainder of the section.

3.2 Structure

The overall structure of a STM file is shown below.

```
System Begin
  Name = <Nickname for the system> [unused]
  Type = Time Domain [unused]

  Transmitter Begin
    NumberOfTurns = <number of turns on the  TX loop>
    PeakCurrent   = <peak current of the TX waveform>
    LoopArea      = <TX loop area in (m^2)>
    BaseFrequency = <waveform base frequency (Hz)>
    WaveformDigitisingFrequency = <waveform sampling freq (Hz)>
    WaveForm<Current|Received> Begin
         time[1] value[1]
         time[2] value[2]
         …
         …
         time[np] value[np]
    [Or alternatively]
         File = <external disk file with two columns as above>
    WaveForm<Current|Received> End
  Transmitter End

  Receiver Begin
    NumberOfWindows = <number of windows = nw>
    WindowWeightingScheme = BoxCar | AreaUnderCurve | LinearTaper

    WindowTimes Begin
         start-time[1] end_time[1]
         start-time[2] end_time[2]
         …
         …
         start-time[nw] end_time[nw]
    WindowTimes End
```

```
     LowPassFilter Begin
          CutOffFrequency = cutoff[1] cutoff[2] … cutoff[nf]
          Order           = order[1]  order[2] … order[nf]
     LowPassFilter End

  Receiver End

  ForwardModelling Begin
     OutputType = B | dB/dt
     XOutputScaling = <scaling factor of X-component data>
     YOutputScaling = <scaling factor of Y-component data>
     ZOutputScaling = <scaling factor of Z-component data>
     SecondaryFieldNormalisation  = none | PPM | PP2M
     FrequenciesPerDecade = <integer ≥ 5>
     NumberOfAbsiccaInHankelTransformEvaluation = <integer ≥ 17>
  ForwardModelling End
System End
```

The ***Name*** keyword is a nickname for the system which is unused by the programs.

The ***Type*** keyword is also currently unused but should be set to ***Type = Time Domain***.

3.3 Transmitter block

NumberOfTurns: sets the number of turns on the transmitter loop. If the data being inverted are normalised for the number of transmitter turn, this should be set to unity.

PeakCurrent: (A) is used to scale the normalised waveform specified in the ***WaveFormCurrent*** block. If the data being inverted are normalised for peak current, this should be set to unity.

LoopArea: (m^2) defines the transmitter loop area in meters squared. If the data being inverted are normalised for transmitter loop area, this should be set to unity.

BaseFrequency: (Hz) sets the repetition frequency in Hertz of the periodic bipolar transmitter waveform. Consider for example a waveform that has a 5 ms positive-going pulse, followed by a 15 ms off-time, followed by 5 ms negative-going pulse, and finally another 15 ms off-time before repeating. This waveform has a 20 ms half-period, a 40 ms full period and a base frequency of 25 H***WaveformDigitisingFrequency***: (Hz) is not strictly a property of the physical transmitter. The piecewise-linear waveform specification in the ***WaveFormCurrent*** block is digitised (linearly re-sampled) into a (typically finer) equispaced time series suitable for the Fast Fourier Transforms (FFTs) to be applied. The ***WaveformDigitisingFrequency*** (*f*) should be large enough to correctly represent the frequency content in the waveform being modelled. Also since a time series with the same spacing is used in the windowing, it should be large enough to ensure that the sample interval ($\Delta t = 1/f$) is less than or equal to the width of the narrowest receiver window. This parameter should also be set high enough to minimise the impact of Gibbs phenomena (ringing) in in the FFTs as explained in §4.4.

`WaveFormCurrent`: block is used to specify a piecewise linear representation of the normalised (by the peak current) transmitter current waveform. The waveform is specified in two columns giving the time (s) / current (A) pairs of each point in the piecewise linear representation. At least a complete half-period of the waveform must be specified, the other half will be automatically infilled by the program with the opposite polarity pulse.

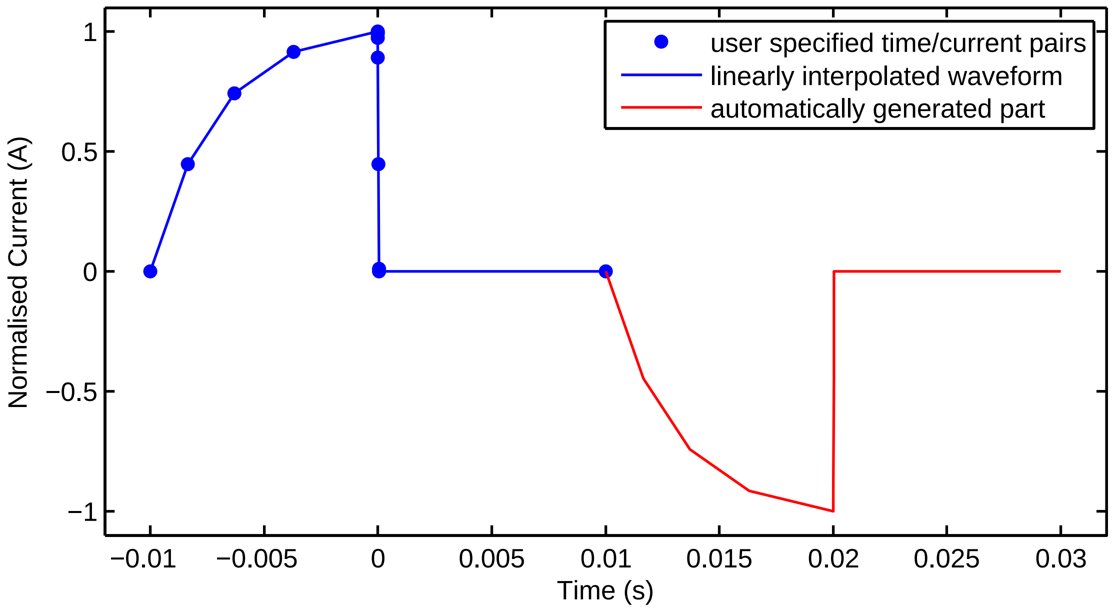

Figure 1: Specification of the waveform current for the 25 Hz base frequency high moment SKYTEM in the example below. At least half of the waveform must be specified by time/current pairs. The program linearly interpolates between the pairs and automatically fills in the remainder of the waveform.

The time values in seconds must be consistent with the times specified in the **`Receiver.WindowTimes`** block (§3.4). Accordingly, it is important that you know where time-zero is defined on the waveform and that the receiver windows time definitions are consistent with that same time-zero. For some systems time-zero is at the beginning of the current off-ramp, and others it is toward the end of the ramp, but in principle it can be defined anywhere on the waveform.

It is also important that you specify the equivalent waveform of the data being inverted rather than the waveform actually transmitted. This particularly applies to the TEMPEST and SPECTREM data which are usually processed to B-field data equivalent to that which would be resultant from square wave. This may also apply to the VTEM data that have undergone full waveform processing to an idealised waveform equivalent data.

It is usually most intuitive to define the positive going pulse and the corresponding following off-time. Typically the first time/current pair will have a negative time value approximately equal to the negative of the pulse width, and the last will have a positive time value approximately equal to the half-period minus the pulse width (unless the whole waveform is specified). The two columns of time/current pairs can be provided within the block as in the 25 Hz base frequency high moment SKYTEM example that follows:

```
WaveFormCurrent Begin
   -1.0000e-002 0.0000e+000
   -8.3400e-003 4.4600e-001
   -6.3000e-003 7.4100e-001
   -3.7000e-003 9.1400e-001
   0.0000e+000 1.0000e+000
   6.1300e-007 9.9800e-001
   1.2000e-006 9.8900e-001
   1.8700e-006 9.7400e-001
   5.3200e-006 8.9200e-001
   2.4100e-005 4.4600e-001
   4.2900e-005 9.5500e-003
   4.3500e-005 4.2200e-003
   4.4200e-005 1.3800e-003
   4.5100e-005 2.7800e-004
   4.7100e-005 0.0000e+000
   1.0000e-002 0.0000e+000
WaveFormCurrent End
```

Alternatively, when the waveform has several hundreds or thousands of points, an external ASCII disk file containing the two columns is usually more convenient. This is achieved using the ***File*** keyword as shown in the VTEM example below.

```
WaveFormCurrent Begin
   File = VTEM-plus-7.3ms-pulse-southernthomson.cfm
WaveFormCurrent End
```

In this case, although possible, it was not convenient to specify all 3,841 points defining the VTEM waveform inside the STM file itself. The external (.cfm) file contains 3,841 points in two columns as specified above, but without the ***WaveFormCurrent Begin/End*** lines. For reference this .cfm file is located in directory ***examples/thomson-vtem/stmfiles*** of the code repository. The file extension is not important but it must reside in the same directory as the parent STM file.

WaveFormRecieved: block can be used to define a receiver voltage waveform if a transmitter current waveform is not available. The waveform is defined in the same way as a current waveform except that time/voltage pairs are specified rather than time/current pairs. This method has not been used or extensively tested by GA, but it may be useful for modelling legacy systems where only receiver waveforms were available.

3.4 Receiver block

`NumberOfWindows`: the number of receiver windows or gates. This must exactly match the number of windows specified in the **`WindoeTime`** block. If you are running an inversion and for some reason do not wish to use all the windows, remove the unwanted windows from the **`WindowTimes`** block and reduce **`NumberOfWindows`** accordingly.

`WindowWeightingScheme`: allows the user to specify one of three methods of combining, in a weighted sum, the individual samples in the decay time series in order to generate the individual windows. **`BoxCar`** simply takes the mean of all values on or in between the window extents (used for TEMPEST). **`AreaUnderCurve`** performs a trapezoid-method integration of the area under the curve between the window extents, and divides by the window width (used for SKYTEM). **`LinearTaper`** uses a weighting function that also includes samples one window-width before the window start time and one window-width after the window end time. The weights are linearly-tapered to zero on either side of the window and are normalised (used for VTEM).

The shapes of the window weighting functions are schematically shown in Figure 2. When using the linear taper, the user should be careful to make sure that the right hand taper of the final window does not extend into the following half-cycle. This would have been the case for the VTEM system example in the repository. To prevent this a small symmetric adjustment was made to the boundaries of the final window (see for example the comments in the STM file ***examples/thomson-vtem/stmfiles/VTEM-plus-7.3ms-pulse-southernthomson.stm***).

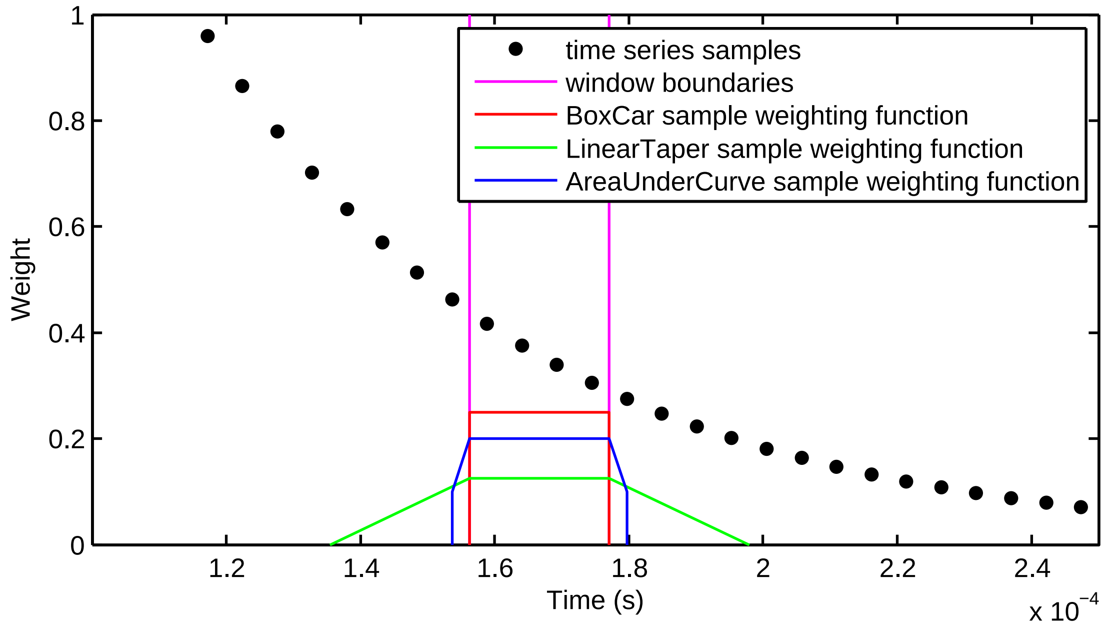

Figure 2: Shapes of the three weighting functions that may be chosen for the weighted average used in the calculation of window values from the time series.

WindowTimes: block is used to define the start and end times in seconds of the gates or windows. The start and end times are given in pairs in two columns. The number of pairs must be the same as the ***NumberOfWindows*** and the times must be consistent with the times used for the transmitter waveform in the ***Transmitter.WaveFormCurrent*** block.

LowPassFilter: block can be used to simulate the effect of low-pass filters in the receiver. The orders and cut-off frequency in Hertz of consecutive Butterworth filters may be specified. For an n order filter with cut-off frequency ω_C, the filter gain at frequency ω is given by,

$$G(\omega) = \sqrt{\frac{1}{1+\left(\frac{\omega}{\omega_C}\right)^{2n}}}. \tag{1}$$

No filters are applied if this block is not specified. The snippet below from the SKYTEM example in the repository will apply two low pass first order filters with cut-off frequencies at 350 kHz and 450 kHz.

```
LowPassFilter Begin
  CutOffFrequency = 300000 450000
  Order           = 1      1
LowPassFilter End
```

3.5 Forward modelling block

OutputType: specifies if the output of the forward models should be a B-field response ***B*** or its time-derivative ***dB/dt***.

XOutputScaling, ***YOutputScaling, ZOutputScaling***: specify the scaling factor applied to the outputs of the X, Y, and Z-component data respectively. For example this can be used to convert the forward model results from the default units of Tesla (or Tesla/second) to femtoTesla (or femtoTesla/second) by setting ***XOutputScaling = 1e15***.

SecondaryFieldNormalisation: allows different types of normalisation of the secondary fields with respect to the primary field (for B-field outputs only). Legal values are ***None***, ***PPM***, and ***PP2M***. No normalisation is applied if ***None*** is selected. If ***PPM*** is selected the secondary fields are expressed as parts-per-million of the component-wise primary field at the peak current. ***PP2M*** applies the normalisation based on the peak-to-peak primary field variation rather than the peak-to-zero variation. We understand that ***PP2M*** is the normalisation appropriate for modelling of SPECTREM data.

FrequenciesPerDecade: sets the number of discrete frequencies per logarithmic decade at which the impulse response is calculated. The impulse response of the conductivity model is calculated at logarithmically spaced frequencies from just below the base frequency to just above the Nyquist frequency, at the specified density. These are then splined to attain values of the impulse response at the same linearly spaced frequencies that occur in the waveform's frequency domain FFT spectrum. It is recommended that ***FrequenciesPerDecade*** is set no lower than ***5*** or ***6***. Using a higher value will provide a little better accuracy but will come at the expense of extra computation time.

NumberOfAbsiccaInHankelTransformEvaluation: sets the number of abscissa (numerical integration nodes) at which the complex reflection coefficients are calculated in the numerical evaluation of the Hankel transform integrals. Evaluation of the Hankel transforms is performed by direct quadrature rather than by the digital filtering techniques used in most other codes. It is recommended that ***NumberOfAbsiccaInHankelTransformEvaluation*** be set to at least ***21*** and possibly up to ***41***. Larger values will give better accuracy up to a point but will also come at the expense of extra computation time.

4 Notes on forward modelling routine

4.1 Coordinate system

Since each AEM sounding (or sample) is inverted separately the coordinate system is different for the inversion of each sounding. A right handed XYZ-Cartesian coordinate system is used. The origin of the coordinate system is on the Earth's surface directly below the centre of the transmitter loop. The positive x-axis is in the direction of flight of the aircraft at that sounding location, the positive y-axis is in the direction of the left wing and the positive z-axis is directed vertically upwards.

4.2 Dipole source

The forward modelling routine is based on a magnetic dipole source. Therefore there is no facility in the STM file to define the parameters of a large loop transmitter source. However, the orientation of the transmitter loop can be defined in the forward modelling and inversion programs by defining a roll, pitch and yaw for each sounding.

4.3 Quasi static approximation

The forward modelling routine relies on the quasistatic assumption and does not consider the effect of displacement currents.

4.4 Fourier transforms

One of the known problems with the code is error that can be introduced in the forward modelling results due to Gibbs phenomena (ringing) in the FFTs. This particularly occurs when the conductivity model is very resistive (~0.001 S/m) and the resultant secondary fields are very small. In these cases the numerical ringing error near abrupt waveform ramps may swamp the secondary field in the first or last few windows (Figure 3). In most cases the numerical ringing error in the forward model is significantly less than typical AEM system observational noise levels and poses no serious problem. However this should be monitored when modelling particularly resistive environments. The forward modelling program (§5) can be used to test this.

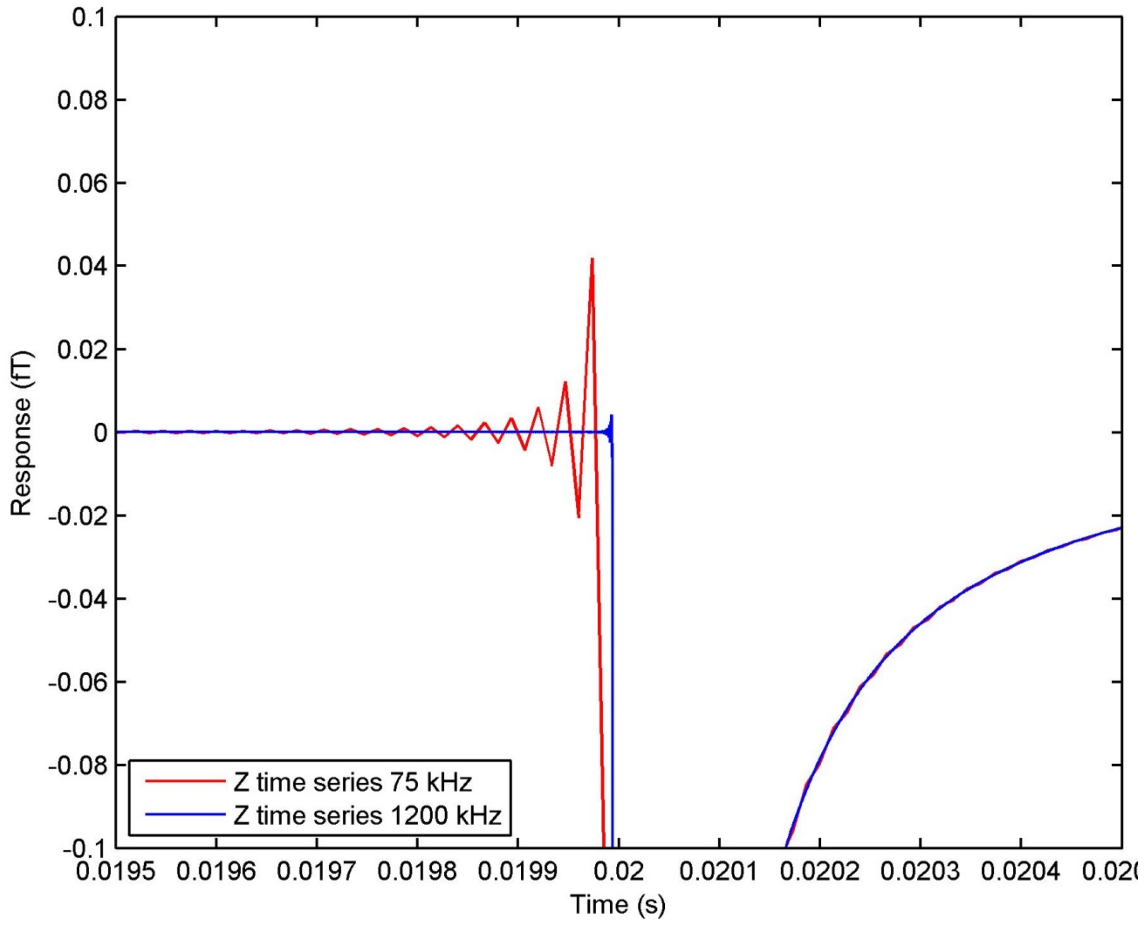

Figure 3: Details near the end of one half-cycle and the start of the next half-cycle of the Z-component secondary field time series, of a relatively resistive 0.001 S/m half-space for the TEMPEST system when the waveform is digitised at 75 kHz and 1200 kHz. Note the significant ringing in the 75 kHz result compared to the 1200 kHz result.

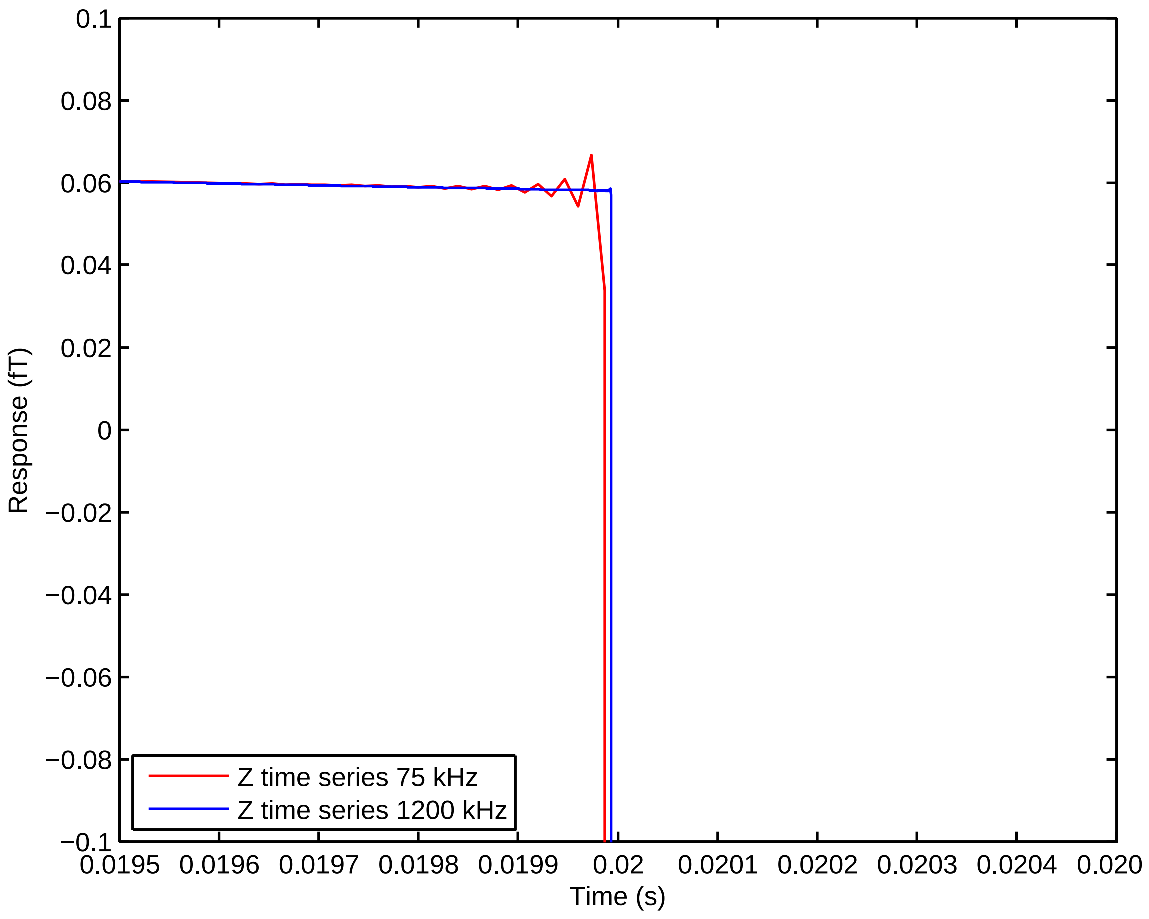

Figure 4: Details near the end of one half-cycle and the start of the next half-cycle of the Z-component secondary field time series, for a relatively conductive 0.100 S/m half-space for the TEMPEST system when the waveform is digitised at 75 kHz and 1200 kHz. Note that while the amount of ringing in the 75 kHz result is greater than in the 1200 kHz result, it is less than in the resistive model shown in Figure 3. Also note that it oscillates about a somewhat greater true background value than for the resistive model where it oscillated about a true value that is below noise levels.

The only way that we know to reduce this effect is to use a higher ***WaveformDigitisingFrequency*** than would otherwise be necessary. This allows the abrupt changes in the waveform to be more accurately represented. Example TEMPEST forward models using 75 kHz and 1200 kHz waveform digitising frequencies are compared in Figure 5 and Figure 6 for relatively conductive and resistive half-spaces respectively. In the conductive case there is no discernible difference between the 75 kHz and 1200 kHz digitizing frequency models, and any differences are well below typical system noise levels. However for the resistive case the differences are significant compared to noise levels in the first three windows. Also the last window value does not plot on the logarithmic plot because it is slightly negative.

Typically we would use 75 kHz digitising frequency for TEMPEST inversion because that mimics the digital receiver sampling interval of the physical TEMPEST system. However if in doubt one should increase the waveform digitising frequency to 1200 kHz or thereabouts to minimise this problem. The higher digitising frequency will come at extra computational cost.

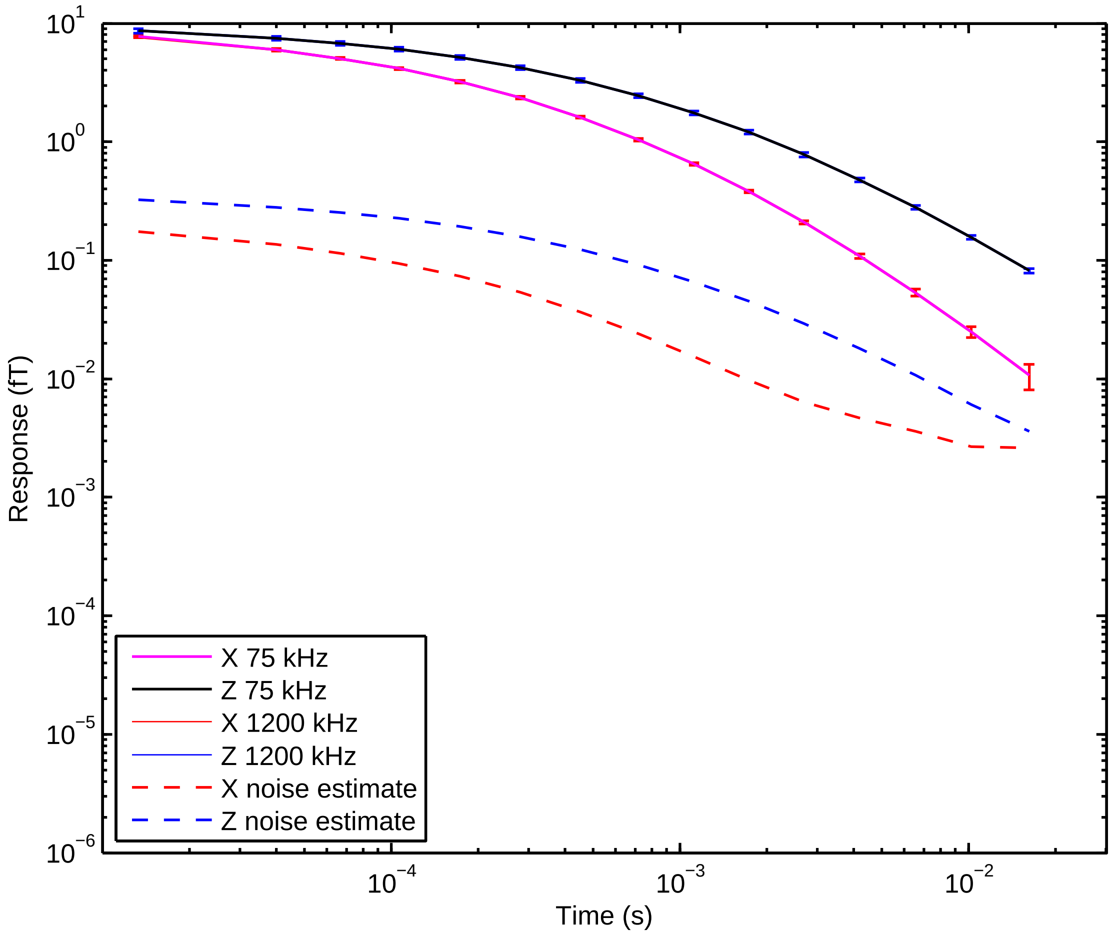

Figure 5: Example forward model of a 0.1 S/m half-space for the TEMPEST system when the waveform is digitised at 75 kHz and 1200 kHz. Typical system noise estimates of TEMPEST data (based on high altitude derived additive noise floor plus 2.3% and 3.7% X- and Z-component relative error respectively) are shown via the error bars and the dashed lines. For this relatively conductive scenario there is no discernible difference between the 75 kHz and 1200 kHz forward models.

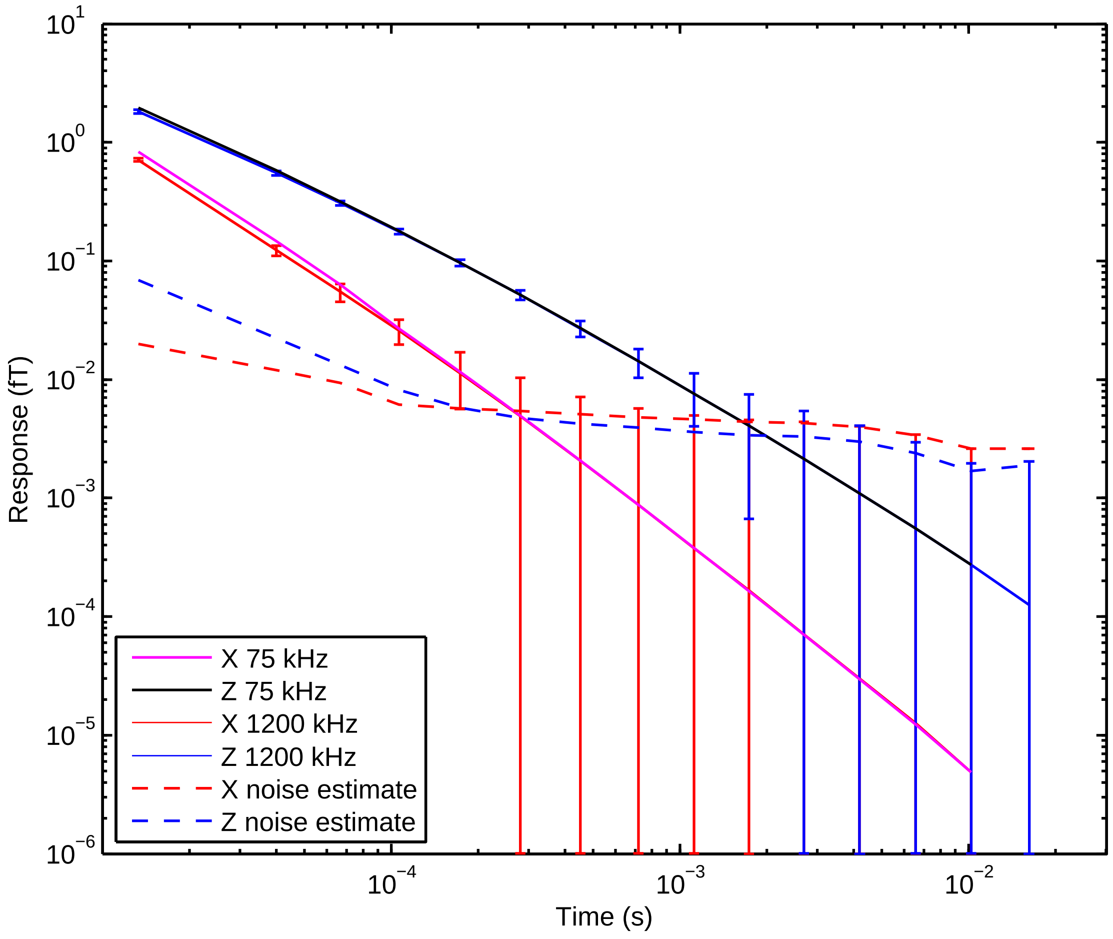

Figure 6: Example forward model of a 0.001 S/m half-space for the TEMPEST system when the waveform is digitised at 75 kHz and 1200 kHz. Typical system noise estimates of TEMPEST data (based on high altitude derived additive noise floor plus 2.3% and 3.7% X- and Z-component relative error respectively) are shown via the error bars and the dashed lines. For this relatively resistive scenario there are differences greater than the observational noise levels between the 75 kHz and 1200 kHz forward models in the first three windows and the last window.

5 Forward modelling program

5.1 Introduction

The forward modelling program is called gaforwardmodeltdem.exe. It is a simple utility designed for running a few forward models for testing purposes. It is useful for making sure that STM files are valid and for checking the results of modelling against other software. It could also be used to generate a suite of forward models for resolution analysis. It is not designed for generating a vast number of forward responses and is thus not parallelised or optimised.

5.2 Execution

The programs is executed from the command line as follows:

```
>> gaforwardmodeltdem.exe control_file_name
```

Make sure that the executable and any dependencies are in your search path (see §1.3.6).

5.3 Control file

The program control file specifies the STM file of the AEM system to be modelled, the input model file and the output data and header files as follows:

```
Control Begin
   SystemFile          = <aem system specification file>
   InputModelFile      = <input geometry and conductivity model file>
   OutputDataFile      = <output forward model data file>
   OutputDataHeader    = <output forward model header file>
Control End
```

The ***SystemFile*** is an AEM system specification file as described in Section 3 and the formats of the input model file and output files are explained in Section 5.4 and 5.5. An example control file is shown in the snippet below.

```
Control Begin
   SystemFile          = ..\stmfiles\Skytem-LM-BHMAR-RP.stm
   InputModelFile      = input_model_skytem.txt
   OutputDataFile      = output_model_Skytem-LM-BHMAR-RP.asc
   OutputDataHeader    = output_model_Skytem-LM-BHMAR-RP.hdr
Control End
```

5.4 Input model file

The ***InputModelFile*** contains the AEM system geometry plus conductivities and thicknesses of each earth model for which a forward model is to be computed. The input model file is a space or comma delimited ASCII text file with each record or line specifying a different model to be computed. The column format of the input model file is as follows:

```
Column    Label          Description and Units
```

```
1     tx_height      TX height above ground m
2     tx_roll        TX roll degrees, left wing up is +ve
3     tx_pitch       TX pitch degrees, nose down is +ve
4     tx_yaw         TX yaw degrees degrees, turn left is +ve
5     txrx_dx        TX-RX inline separation m, RX in front of TX +ve
6     txrx_dy        TX-RX transverse separation m, RX left of TX +ve
7     txrx_dz        TX-RX vertical separation m, RX above TX is +ve
8     rx_roll        RX roll degrees degrees, left wing up is +ve
9     rx_pitch       RX pitch degrees degrees, nose down is +ve
10    rx_yaw         RX yaw degrees degrees, turn left is +ve
11    N              number of layers
12    cond:1         layer 1 conductivity S/m
13    cond:2         layer 2 conductivity S/m
...
12+N-1    cond:N         layer N conductivity S/m
12+N      thickness:1    layer 1 thickness m
12+N+1    thickness:2    layer 2 thickness m
...
12+2N-1   thickness:N-1  layer N-1 thickness m
```

Note that a different number of layers (N) can be specified in each record of the file and that the thickness of the bottom half-space layer is infinite and is thus not specified. Note also that there is no header line in the input file.

The following example input model file is for forward modelling one-, two- and three-layer conductivity models for a SKYTEM system where the transmitter loop is 30 m above ground and is level (zero roll, pitch and yaw) and the receiver is 12.7 m behind and 2.09 m above the centre of the transmitter loop.

```
30 0 0 0 -12.7 0 +2.09 0 0 0 1 0.100
30 0 0 0 -12.7 0 +2.09 0 0 0 2 0.200 0.001 20
30 0 0 0 -12.7 0 +2.09 0 0 0 3 0.010 0.200 0.001 20 20
```

5.5 Output files

In the ASCII text file specified by the ***OutputDataFile*** keyword the program outputs the forward models, primary field and *W* secondary field windows, in the following column format;

```
Columns        Label          Description
1         X_Primary     X-component primary field (B-field only)
2         Y_Primary     Y-component primary field (B-field only)
3         Z_Primary     Z-component primary field (B-field only)
4-4+W-1        X_Secondary   X-component secondary field windows (1-W)
4+W-4+2W-1     Y_Secondary   Y-component secondary field windows (1-W)
4+2W-4+3W-1    Z_Secondary   Z-component secondary field windows (1-W)
```

The primary field values are only output if ***OutputType=B*** in the STM file. Hence the first three columns are not included for ***dB/dt*** output systems. A simple ASCII header file giving the column numbers and labels, as shown in the snippet above, is also output in the file specified by the ***OutputDataHeader*** keyword.

5.6 Examples

Examples of the use of the forward modelling program are provided in the following repository directories;

1. ***examples/bhmar-skytem/gaforwardmodeltdem***,
2. ***examples/frome-tempest/gaforwardmodeltdem***,
3. ***examples/thomson-vtem/gaforwardmodeltdem***.

6 Deterministic inversion program

6.1 Introduction

The deterministic time domain AEM inversion program is called galeisbstdem.exe. The *"sbs"* in the name stands for sample-by-sample, which indicates that each AEM sample or sounding is inverted totally independently of every other sample. It is only after the inversion that the resultant models are stitched together into sections, grids, or voxets in some other program.

The *"lei"* in the name stands for layered-earth-inversion, which indicates that the underlying parameterisation of the inversion model is a 1D or layered earth in which the conductivity and thickness of every individual layer of the conductivity model is assumed to be laterally (i.e., in a horizontal sense) constant for each sounding that is inverted. It is only after stitching them together in another program that the appearance of laterally variable conductivity model is built up.

The meaning of deterministic is that, in each run, the programs finds just one solution (conductivity model) that fits the data for each sounding. This is quite different from the stochastic inversion program described in Section 7, which finds a large ensemble of models that fit the data.

There are various options in the programs, one of which is to solve for the Tx-Rx horizontal and vertical offsets and/or Rx pitch to help deal with the practical problems associated with imprecise knowledge of the Rx position and primary field in fixed-wing AEM. Other significant options include; (1) smooth multi-layer fixed-thickness, or blocky few-layer variable-thickness style inversions, (2) which Rx component(s) to invert, (3) which geometry variables to solve for.

6.1.1 Outline of the algorithm

The underlying algorithm, to a large extent, follows the theoretical developments of (Constable *et al.*, 1987), however with additional functionality to deal with common practical AEM problems such as solving for system geometry variables.

The algorithm minimizes an overall objective function,

$$\Phi = \Phi_{\mathrm{d}} + \lambda\Phi_{\mathrm{m}}, \tag{2}$$

subject to the constraint,

$$\Phi_{\mathrm{d}} \cong 1. \tag{3}$$

The objective function is comprised of a data misfit term Φ_{d}, and model norm term Φ_{m}, whose influences are relatively weighted by the regularization parameter λ,

The data misfit Φ_{d} is an *L2* measure of the discrepancy between the observed data and the forward model, normalized by the estimated errors in observed data and the number of data.

The model norm Φ_m term is given by,

$$\Phi_m = \alpha_c\Phi_c + \alpha_t\Phi_t + \alpha_g\Phi_g + \alpha_s\Phi_s, \qquad (4)$$

is composed of the an overall regularization parameter λ that scales a weighted sum of four separate *L2* norms. The terms Φ_c, Φ_t, Φ_g, quantify the difference between the inversion model parameters (unknowns) and their corresponding conductivity, thickness and system geometry reference model parameters respectively. The term Φ_s quantifies the roughness of the conductivity model's vertical structure. The weights on the individual model norm terms (α_c, α_t, α_g, and α_s) are chosen by the user and remain fixed remain fixed throughout the iterative inversion procedure.

The overall regularization parameter λ is automatically chosen by the algorithm. Initially λ is set to a high value and in each non-linear iteration a line search is performed on λ to find its value so that Φ_d is reduced to a target misfit $\Phi_d \cong 0.7$ of its previous value. This iterative process continues until the constraint $\Phi_d \cong 1$ is satisfied, or the improvement between successive iterations is too small or the maximum number of iterations is reached.

6.1.2 System geometry inversion

The program has the functionality to invert for parameters of the system geometry that may not be very easily measured or estimated. This is most likely to be the case for fixed-wing systems where the towed-receiver bird is distant to the aircraft and it is logistically difficult to accurately measure its attitude and position relative to the transmitter (Smith, 2001b). The functionality can however be used for helicopter systems if the system geometry is not fully measured.

It is never particularly desirable to invert for system geometry parameters because AEM inversion is ambiguous (non-unique) enough without adding extra unknowns into the mix. However, we have found that when inverting data from fixed-wing platforms it is often impossible to fit data, to the expected noise levels, when inverting both X- and Z-component data simultaneously. It is usually possible to fit the X- or Z-component data in two independent inversions, but only to find that the two conductivity models are systematically different, which then raises the question of which of the two models should be used. For fixed-wing systems that employ full-waveform recording and processing, a particular problem arises because of the intertwined problem of estimating the primary field and system geometry. That problem can be aided by inversion of total (primary plus secondary) field data and is discussed in detail in Section 6.1.3,

Even for off-time helicopter systems, there may be circumstances where it is necessary to invert for unknown elements of the system geometry. Examples would be where vegetation causes large errors in altimeter data or where altimeter and orientation instruments (e.g., gyroscopes or tilt meters) are not mounted on the transmitter/receiver assembly. Though not optimal, the system geometry inversion option can be used on the secondary field data, without the need to invert total field data. This should only be used if there genuine reasons and evidence to believe that the system geometry data are incorrect and are not allowing the AEM data to fitted with reasonable conductivity models.

For an off-time system, in which all the receiver windows are recorded when the transmitter is turned off, and has an accurately known system geometry there is no problem— we may simply use the measured system geometry to forward model the secondary field and invert the solely secondary field response data and do not need to concern ourselves with the primary or total fields.

6.1.3 Reconciling primary field and system geometry

Uncertain knowledge of the system geometry is intertwined with uncertain knowledge of the primary field, because if the system geometry was known accurately the primary field at the receiver bird could be calculated accurately. We are not particularly interested in the primary field or the system geometry since they tell us nothing about subsurface conductivity. Nevertheless, the system geometry is required for forward modelling and hence inversion of the secondary field response, which does inform us about the subsurface.

For systems where the geometry is not explicitly measured, on-time primary field responses may be used to estimate the receiver position under certain assumptions about subsurface conductivity and the orientation of the receiver. However at survey altitude the system actually measures a combined total (primary plus secondary) field response and the primary field must first be isolated before it can be used to estimate a corresponding system geometry. But there is a conundrum because the former is not possible without the obviously unknown subsurface conductivity. Therefore service providers have to estimate the system geometry and primary field using a certain set of assumptions. These assumptions are different for each system. See Smith (2001a); Lane *et al.* (2000), and (Leggatt *et al.*, 2000) for informative discussions on this issue.

An alternative way to deal with this conundrum is to simultaneously invert total field data for a conductivity model as well as the unknown parameters of the system geometry. This in turn implies a different inversion derived) estimate of the primary field. This functionality is built into the algorithm and it has been widely used for the inversion of TEMPEST data (e.g., Roach, 2012) and to some extent SPECTREM data (e.g., Ley-Cooper and Brodie, 2013). However, please note that the functionality for inverting total field data in this way can only be applied to AEM system data where the data are equivalent square-wave B-field data.

6.1.4 Total field reconstruction

Since we routinely use the functionality to use total field data when solving for system geometry when inverting data from the TEMPEST system, a brief explanation is provided here. Firstly, the height, pitch, roll and geometry corrected (HPRG) TEMPEST data should not be used for total field reconstruction and system geometry inversion, because it has already undergone processing modifications that are not consistent with and cannot be unwound by the program. It is the non-HPRG data that must be used instead.

Part of the TEMPEST data processing sequence involves partitioning the total (primary plus secondary) field response that is actually measured over the full waveform into estimates of the unknown primary and secondary field components (Lane *et al.*, 2000; Bergeron and Lawrence, 2010). Then, using the partitioned primary field estimate, a corresponding estimate of the transmitter to receiver horizontal in-line ($\mathrm{D_x}$) and vertical ($\mathrm{D_z}$) separations are made analytically. The procedure assumes that the transmitter and receiver are flying straight and level and that the receiver coils are directly behind the aircraft (i.e., $\mathrm{T_r} = \mathrm{T_y} = \mathrm{D_y} = \mathrm{R_r} = \mathrm{R_p} = \mathrm{R_y} = 0$) except that the transmitter is pitched at $\mathrm{T_p} = -0.45°$ (+0.45 in the TEMPEST sign convention) for the Skyvan platform and $\mathrm{T_p} = -0.90°$ (+0.90 in the TEMPEST sign convention) for the Casa platform.

The estimated secondary field data, and the measured elements of the system geometry ($\mathrm{T_h}$, $\mathrm{T_p}$, $\mathrm{T_r}$) and the corresponding estimates of the unmeasured elements of the system geometry ($\mathrm{D_x}$ and $\mathrm{D_z}$) are usually delivered to clients as the non-height, pitch, roll and geometry corrected (non-HPRG) data. The assumed elements of the system geometry $\mathrm{T_y} = \mathrm{D_y} = \mathrm{R_r} = \mathrm{R_p} = \mathrm{R_y} = 0$) are not usually delivered as columns in the non-HPRG dataset, but they are nevertheless implicit in that dataset. However the estimated primary field data are not delivered to clients at this point in time.

To make its own estimate of primary field and system geometry as part of the inversion procedure, the program needs to work with total field data. The user may calculate the primary field outside the program and directly input it via the data file or it may be reconstructed within the program, To perform the later, the program must be told the estimated $\mathrm{D_x}$ and $\mathrm{D_z}$ and the assumptions used in their generation (e.g., $\mathrm{T_p} = -0.45°$, for the Skyvan platform and $\mathrm{T_r} = \mathrm{T_y} = \mathrm{D_y} = \mathrm{R_r} = \mathrm{R_p} = \mathrm{R_y} = 0$). With this information, it is a simple matter of recomputing the primary field then adding them to the delivered secondary field data to generate the total field data that are subsequently inverted.

6.2 Execution

Since the program is parallelized via MPI or OpenMP it can be executed in three different ways. To execute the program as a stand-alone (single CPU process) use the following method.

```
>> galeisbstdem.exe control_file_name
```

To execute the program on *N* threads using OpenMP parallelism use the following invocation.

```
>> galeisbstdem.exe control_file_name N
```

To execute the program using N processors using MPI parallelism you need to use your system's MPI installation's start up daemon (usually called *'mpirun'* or *'mpiexec'*) to invoke the program as follows:

```
>> mpirun -np N galeisbstdem.exe control_file_name
>> mpiexec -np N galeisbstdem.exe control_file_nam
```

If you are using the pre-compiled WINDOWS executables, but do not have the Microsoft HPC Pack 2012 (see §1.3.4) implementation of MPI installed on your PC, then you may use the separate program that was compiled without MPI. This program will still use OpenMP on N threads if you use the invocation.

```
>> galeisbstdem-nompi.exe control_file_name N
```

Make sure that the executable and any dependencies are in your search path (see §1.3.6).

6.3 Control file

6.3.1 Structure

Overall the control file is structured as shown in the snippet below. Each of the nested blocks will be explained in the remaining subsections of this section. To simplify the discussion the shorthand where ***{X/Y/Z}Blah*** expands to ***XBlah, YBlah*** or ***ZBlah*** is used.

```
Control Begin
  NumberOfSystems = Ns number of EMSystem blocks to follow
  EMSystem1 Block
  …
  EMSystemNs Block
  Earth Begin
    NumberOfLayers = <Nl number of layers in conductivity model>
  Earth End
  Options Block
  Input Block
  Output Block
Control End
```

The ***NumberOfSystems*** sets the number of AEM systems N_S that are to be considered in the inversion. This will in most cases be set to one, except where multi-moment AEM system data are to be inverted. This option has been successfully used with the dual-moment SKYTEM system, and although it has so far never been tested, it could in principle be used to invert multi-pulse HELIGEOTEM data,

To invert multi-moment data each of the data must be included in the same input data file, which also means the sounding for each moment must be located at coincident fiducials or locations. For example with interleaved high and low moment SYKTEM data, which are ultimately acquired at different locations along the flight line, need to somehow be combined into data at coincident fiducial position externally from the inversion program. The user needs to decide the most appropriate way to achieve this, whether it be by interpolation, nearest neighbour resampling or asking the service provider to deliver data where the data for each moment are drawn from the stacking filter at coincident locations. In the past for SKYTEM data we have simply linearly interpolated the most coarsely sampled moment data to the same fiducial location as the most finely sampled moment data.

The ***NumberOfLayers*** keyword in the ***Earth*** block sets the fixed number of layers N_l in the conductivity model.

6.3.2 AEM system blocks

Each of the N_s AEM system to be considered must be specified in blocks named ***EMSystem1*** through to ***EMSystemN$_s$***. Their structure is as shown below.

```
EMSystem{1/2/…Ns} Begin
   SystemFile = path to AEM system STM file
   Use{X/Y/Z}Component = <yes | no>
   Use{X/Y/Z}Component = <yes | no>
   Use{X/Y/Z}Component = <yes | no>

   InvertTotalField = <yes | no>
   ReconstructPrimaryFieldFromInputGeometry = <yes | no>

EstimateNoiseFromModel      = <yes | no>
   {X/Y/Z}MultiplicativeNoise = <Val mj percent noise for component>
   {X/Y/Z}AdditiveNoise      = <Arr[1:Nw]} ajk additive noise windows>

   {X/Y/Z}ComponentSecondary = <Column C | -Column C>
   {X/Y/Z}ComponentPrimary   = <Column C | -Column C>
{X/Y/Z}ComponentNoise      = <Column C | -Column C>
EMSystem{1/2/…Ns} End
```

SystemFile sets the path to the STM file in the format specified in Section 3.

Use{X/Y/Z}Component tells the program which components of the data are to be used in the inversion. We have never tested Y-component data inversion.

InvertTotalField = yes should only be used when dealing with equivalent square-wave B-field data such as TEMPEST and SPECTREM. It allows the primary plus secondary field to be added together and inverted as total field data. This may be used when inverting for the TX-RX offsets and RX pitch in fixed-wing systems where the system geometry may not be accurately measured or estimated, and hence the primary/secondary field separations may not have been accurate either as explained earlier in Section 6.1.2.

Set ***ReconstructPrimaryFieldFromInputGeometry = yes*** if you are inverting total field data and the primary field values for each component are not available in the input data file. In this case the primary field data will be calculated from the input system geometry values specified in the ***TotalFieldReconstruction*** block. To use this option it is important to understand how the primary field was removed in the first place. If ***ReconstructPrimaryFieldFromInputGeometry = no*** then the total field is reconstructed using the primary field value specified by the ***{X/Y/Z}ComponentPrimary*** keywords.

EstimateNoiseFromModel=yes tells the program to calculate noise for each sample, component, window combination using an additive/multiplicative noodel. In this model the standard deviation noise estimate for the *i^th^* sample, *j^th^* component and *k^th^* window is,

$$e_{ijk} = \sqrt{(0.01 \times m_j \times s_{ijk})^2 + a_{jk}^2} \qquad (5)$$

where m_j is the percentage error assigned to all windows in the j^{th} component, s_{ijk} is the secondary field data for the i^{th} sample, j^{th} component and k^{th} window, a_{jk} is the additive noise floor assigned to the j^{th} component and k^{th} window. If you prefer not to use this noise model set ***EstimateNoiseFromModel=no*** then the noise estimates must be read directly from the input data file using the ***{X/Y/Z}ComponentNoise*** keyword below. Either way, make certain that no noise estimate is ever zero otherwise you will get divide by zero errors and crash the program,

{X/Y/Z}MultiplicativeNoise specifies the survey wide percentage noise estimate (i.e., m_j) for the component if the noise model is being used.

{X/Y/Z}AdditiveNoise specifies the survey wide additive noise estimate (i.e., a_{jk}) for every window of the component if the noise model is being used.

{X/Y/Z}ComponentSecondary specifies the start column number of the N_w secondary field windows in the ***DataFile***. Note that the units and sign convention of the input data must be the same as those produced by the forward modelling routine according to the scaling and normalisation settings in the STM file (see §3.5). Recall that as explained in Section 2.4you can conveniently flip the sign of the input data by using the "-Column" shortcut" (e.g., ***ZComponentSecondary = -Column 52)***

You only need to provide this for the components that are being inverted as per the ***Use{X/Y/Z}Component*** keyword.

{X/Y/Z}ComponentPrimary specifies the one column number of the primary field data in the ***DataFile***. This is not required if you are inverting total field data and have set ***InvertTotalField = yes***. You only need to provide this for the components that are being inverted as per the ***Use{X/Y/Z}Component*** keyword.

{X/Y/Z}ComponentNoise specifies the start column number of the N_w columns containing standard deviation noise estimate columns in the ***DataFile***. This must be used if you prefer not to use the noise model discussed above (Equation 5) and have set

EstimateNoiseFromModel=no. You only need to provide this for the components that are being inverted as per the ***Use{X/Y/Z}Component*** keyword.

6.3.3 Options block

The ***Options*** block is sued to set various program switches. The basic structure is shown in the snippet below.

```
Options Begin
  SolveConductivity = <yes | no>
  SolveThickness    = <yes | no>

  SolveTX_Height = <yes | no>
  SolveTX_Roll   = <yes | no>
  SolveTX_Pitch  = <yes | no>
  SolveTX_Yaw    = <yes | no>
  SolveTXRX_DX   = <yes | no>
  SolveTXRX_DY   = <yes | no>
  SolveTXRX_DZ   = <yes | no>
  SolveRX_Roll   = <yes | no>
  SolveRX_Pitch  = <yes | no>
  SolveRX_Yaw    = <yes | no>

  AlphaConductivity = <αc conductivity reference model weight>
  AlphaThickness    = <αt thickness reference model weight>
  AlphaGeometry     = <αg system geometry reference model weight>
  AlphaSmoothness   = <αs vertical smoothness weight>
  SmoothnessMethod  = <Minimise1stDerivatives|Minimise2ndDerivatives>

  MinimumPhiD = stop when Φd goes below this value
  MinimumPercentageImprovement = stop if Φd improvs less than this %
  MaximumIterations = stop after this number of iterations
Options End
```

Always use ***SolveConductivity = yes*** to solve for the layer conductivities. The program will probably crash if ***SolveConductivity = no*** is an untested option at this stage.

For a multi-layer vertically smoothed Occam style model set ***SolveThickness=no*** and the layer thicknesses will remained fixed at the reference/starting model's layer thicknesses. To run a few-layer blocky style inversion set ***SolveThickness=yes*** and the routine will solve for the layer thicknesses as well as their conductivities.

The ten keywords ***SolveTX_Height*** through to ***SolveRX_Yaw*** shown in the snippit above allow the user to optionally solve for the ten AEM system geometry parameters. Although all ten are available only ***SolveTX_Height***, ***SolveTXRX_DX,*** ***SolveTXRX_DZ*** and ***SolveRX_Pitch*** have been tested or likely to be useful. If you do not solve for geometry parameters they are left the same as the ***Input.Columns*** block.

The keywords ***AlphaConductivity***, ***AlphaThickness***, ***AlphaGeometry***, and ***AlphaSmoothness*** respectively set the weights α_c, α_t, α_g, α_s of the individual model norm terms in the objective function (Equation 4). The larger the weights the greater the penalty for the inversion model deviating away from the reference model. These are in effect relative weights because the overall regularization parameter λ scales them automatically.

It is recommended that you always keep ***AlphaConductivity=1.0*** and vary the other weights. If the uncertainties on the reference model parameters (see the ***StdDevReferenceModel*** block) are set to realistic values then ***AlphaThickness*** and ***Alphageometry*** should also be set around 1.0. ***AlphaThickness*** and ***Alphageometry*** have no effect if you are not solving for thickness or geometry respectively.

If ***AlphaSmoothness*** is too small the inversion may get unwieldy oscillations and not be able to fit the data. If it is too large it may produce overly smooth models or prevent the data from fitting. In general it is recommended that you start by setting ***AlphaSmoothness*** to a large value (e.g., 1.0e5) and then vary it down by factors of ten in additional program runs until you achieve your preferred level of conductivity model vertical smoothness. Ultimately, provided the data is being adequately fitted, it is a subjective choice unless the model is being compared to some type of ground truth.

The user can choose to minimise either the first or second derivatives of the vertical conductivity structure using ***SmoothnessMethod=Minimise1stDerivatives*** or ***SmoothnessMethod=Minimise2ndDerivatives***. If neither is specified the default is to use the second derivatives.

The keyword ***MinimumPhiD*** sets the minimum value to which the normalised data misfit Φ_d term (Equation 2 and 3) will be reduced before the algorithm stops iterating and it moves on to the next sounding. Usually this should be set to a value of 1.0 which, if the noise estimates are good, implies that data is fitted adequately but also not over fitted (i.e., not fitting the noise).

If for some reason the algorithm cannot improve the data misfit Φ_d by more than ***MinimumPercentageImprovement*** the algorithm stops iterating and it moves onto the next sounding.

MaximumIterations sets the maximum number of iterations allowed before the algorithm stops iterating and it moves onto the next sounding.

6.3.4 Input block

The ***Input*** block is where the input data file and most of the column numbers are specified. However, recall that the column numbers for the AEM data (and possibly noise estimates) are specified in the relevant AEM system block (§6.3.2). The general structure of the input block is shown in the snippet below.

The ***DataFile*** keyword give the path name to the single data file which holds all the AEM data and possibly noise estimates, the reference model and its uncertainties. The data file must be in a flat ASCII space-delimited column format. If it has lines of header at the beginning, these can be skipped using the ***HeaderLines*** keyword. You may choose to invert only every *n*th sample, in which case you would set the keyword ***Subsample=n***. The input data file may not have the interspersed line numbers that is sometime the case with GEOSOFT .xyz style data files.

```
Input Begin
  DataFile    = <path to input data file>
  HeaderLines = <number of header lines to be skipped>
  Subsample   = <subsampling interval of input data file>

  Columns Begin
    SurveyNumber    = <Val | Column C | -Column C>
    DateNumber      = <Val | Column C | -Column C>
    FlightNumber    = <Val | Column C | -Column C>
    LineNumber      = <Val | Column C | -Column C>
    FidNumber       = <Val | Column C | -Column C>
    Easting         = <Val | Column C | -Column C>
    Northing        = <Val | Column C | -Column C>
    GroundElevation = <Val | Column C | -Column C>
    Altimeter       = <Val | Column C | -Column C>

    TX_Height       = <Val | Column C | -Column C>
    TX_Roll         = <Val | Column C | -Column C>
    TX_Pitch        = <Val | Column C | -Column C>
    TX_Yaw          = <Val | Column C | -Column C>
    TXRX_DX         = <Val | Column C | -Column C>
    TXRX_DY         = <Val | Column C | -Column C>
    TXRX_DZ         = <Val | Column C | -Column C>
    RX_Roll         = <Val | Column C | -Column C>
    RX_Pitch        = <Val | Column C | -Column C>
    RX_Yaw          = <Val | Column C | -Column C>

    TotalFieldReconstruction Begin
        //Only required sometimes
        TX_Roll         = <Val | Column C | -Column C>
        TX_Pitch        = <Val | Column C | -Column C>
        TX_Yaw          = <Val | Column C | -Column C>
        TXRX_DX         = <Val | Column C | -Column C>
        TXRX_DY         = <Val | Column C | -Column C>
        TXRX_DZ         = <Val | Column C | -Column C>
        RX_Roll         = <Val | Column C | -Column C>
        RX_Pitch        = <Val | Column C | -Column C>
        RX_Yaw          = <Val | Column C | -Column C>
    TotalFieldReconstruction End

    ReferenceModel Begin
        Conductivity = <Arr[NL] | Val | Column C | -Column C>
        Thickness    = <Arr[NL] | Val | Column C | -Column C>
        //These below required if inverting for geometry
        TXRX_DX      = <Val | Column C | -Column C>
        TXRX_DZ      = <Val | Column C | -Column C>
```

```
          RX_Pitch      = <Val | Column C | -Column C>
     ReferenceModel End

     StdDevReferenceModel Begin
          //IMPORTANT conductivity and thickness stddevs
          //must be in units of log10(S/m) and log10(m)
          Conductivity = <Arr[N_L] | Val | Column C | -Column C>
          Thickness     = <Arr[N_L-1] | Val | Column C | -Column C>
          //These below required if inverting for geometry
          TXRX_DX       = <Val | Column C | -Column C>
          TXRX_DZ       = <Val | Column C | -Column C>
          RX_Pitch      = <Val | Column C | -Column C>
     StdDevReferenceModel End
  Columns End
Input End
```

The **Columns** block is where the program is told where individual data fields are located within the data file. In some cases a field may not actually be in the data file because it is constant for the whole dataset. In these cases you can set the field's value without giving a column number. This is the meaning of the notation **<Val>** in the snippet above. For example the transmitter yaw is usually not supplied in a data file because it may be zero by definition. In this case it could be set by using ***TX_Yaw=0***. On the other hand if the field was in the data file it can be set by using the **Column** notation, or alternatively the **-Column** notation if you want to have the sign flipped as was explained in Section 2.3.

The fields ***SurveyNumber***, ***DateNumber***, ***FlightNumber***, ***LineNumber***, ***FidNumber***, ***Easting***, ***Northing***, ***GroundElevation*** and ***Altimeter*** are hopefully self-explanatory. These are not used by the inversion for anything other than to be written back out to the output file for the user's convenience and book keeping. It may not be obvious that this includes the ***Altimeter*** field since the ***TX_Height*** field is used for the transmitter height in the modelling. If for example, the altimeter and the transmitter height were equivalent, there is no problem with setting both these to the same column number.

The system geometry input variables ***TX_Height***, ***TX_Roll***, ***TX_Pitch***, ***TX_Yaw***, ***TXRX_DX***, ***TXRX_DY***, ***TXRX_DZ***, ***RX_Roll***, ***RX_Pitch*** and ***RX_Yaw*** are used to specify the input system geometry that is needed for the modelling. Be certain to specify these with the coordinate system and sign convention used by the inversion program (see §4.1) rather than the AEM system itself. It is often convienent to flip signs of variable using, for example ***TX_Pitch = -Column 19***, which is often necessary for TEMPEST data. Also more often than not you will probably set many of these value to zero, for example ***RX_Pitch = 0***, because they are not measured and are assumed to be zero.

The ***TotalFieldReconstruction*** sub-block is only required and used if you are inverting B-field total field data and need to reconstruct the primary field and have set ***ReconstructPrimaryFieldFromInputGeometry = yes*** because the primary field data are no available in the data file. Inside the block specify the column numbers or values of the system geometry variables that were used or assumed in removing the primary field. In other words a primary field forward model of the system geometry variable specified in this block should be equal to the primary field that was removed from the data. This system geometry is first used to run a primary field forward model which is added to the secondary field data to produce the total field data used for inversion. Note that the transmitter height is not required since it does no enter into the primary field calculation. Note also that you only need to specify the geometry variables that are different from the input system geometry variables specified earlier.

The ***ReferenceModel*** block is where you specify the inversion reference model, which also doubles as the starting model. For the layer conductivity and thickness fields an array of *n* values is required as is indicated by the ***Arr[n]*** notation above. If the values are constant for the whole survey dataset, they can be specified in the control file by just typing in the array of space delimited values after the equals sign. For example to specify five layer conductivity reference model you could set ***Conductivity = 0.01 0.07 0.2 0.07 0.001***. However you may prefer to have a reference model that changes with survey location, in which case you would use ***Conductivity = Column 56*** if your five reference model conductivity values were in columns 56 to 60. For especially lazy users, a single value rather than an array of values may be supplied if all the values in the array are to be the same. This notation is useful if, for example, you are inverting for a thirty layer conductivity model but want to have the sane reference value for every layer (i.e., a homogeneous halfspace reference model). In this case you can just be set to ***Conductivity = 0.001*** and the program will expand the single value out to an array of thirty values.

6.3.5 Output block

In the ***Output*** block you specify what information to output and the pathname of the output file(s). The general structure of the block is shown in the snippet below.

```
Output Begin
  DataFile = <path to output data file>
  LogFile  = <path to output log file>

  PositiveLayerBottomDepths = <yes | no>
  NegativeLayerBottomDepths = <yes | no>
  InterfaceElevations       = <yes | no>
  PredictedData             = <yes | no>
  ParameterSensitivity      = <yes | no>
  ParameterUncertainty      = <yes | no>
Output End
```

The ***DataFile*** keyword specifies the pathname of the output data file. It can be an absolute path or a relative path from where the program was launched. The program actually outputs a different file from every MPI or OpenMP process and will automatically append the process number to the output file name. For example if you have ***DataFile=fred.asc*** the outputs will be called ***fred.0000.asc***, ***fred.0001.asc*** ***fred.0002.asc***, **…** and so on. The individual files are simply concatenated and sorted together after the inversion is finished using either a WINDOWS batch (***.bat***) or LINUX shell (***.sh***) script as shown in Section 6.4.3.

The ***LogFile*** keyword is where you specify the output log file which will be populated with a record of the control file and the STM files used in the inversion plus a summary of the inversion of each sample and some error messages. Similarly to the output data file each process outputs its own log file.

The program always outputs a unique identifier which is the record number from the input data file to the output data file. This is later used in combining and sorting the results (see §6.4.3). It also always outputs all of the sounding identification and location fields, that is the survey number, flight number through to the altimeter field specified in the input columns block (see §6.3.4).

The program always outputs the inverted system geometry, the number of layers in the model and their conductivities (in S/m) and thicknesses (in m). Given that the bottom layer is actually infinitely thick, in the inversion modelling, you may be surprised to see that it outputs N_l thickness rather than N_l-1 thicknesses. If you were not solving for thickness, the bottom layer is assigned a thickness equal to the thickness of the second-to-bottom layer for output. If you were not solving for thickness, the bottom layer is assigned a value of 100 m for output. This is simply a convenience when using various visualisation programs that do not handle an infinite thickness elegantly.

If you set ***PositiveLayerBottomDepths=yes*** the program will output the depths to the bottom of each layer (in m) as a positive value. Similarly you can output these as negative depth values using ***NegativeLayerBottomDepths=yes***, which we find is useful for generating conductivity sections in some programs (e.g., ProfileAnalyst™). Also by setting ***InterfaceElevations=yes***, the elevations of the N_l+1 layer interfaces including, above the top layer and below the bottom layer, can be output. The interface elevations are calculated using the ***GroundElevation*** field specified in the ***Input.Columns*** block (§6.3.4), and hence the interface elevations are relative to the same height datum as that input field. For all these options the (infinite) bottom layer thickness is assigned as explained in the preceding paragraph.

The predicted data, that is the forward response of the final inversion model, can also be optionally output by setting ***PredictedData=yes***.

Two different measures of sensitivity may also be output as well by setting ***ParameterSensitivity=yes*** and/or ***ParameterUncertainty=yes***. The sensitivities must not be interpreted in any absolute definitive sense because they are based on linearized estimates of the sensitivities calculated at the final inversion model's location in parameter space. They do not adequately account the influence of the regularization on the final inversion model's position in parameter space, or the trade-offs between parameters.

The ***ParameterSensitivity*** measure for the j^{th} parameter is given by,

$$S_j \;= \Sigma_i^{N_d} \left|\frac{J_{ij}}{e_i}\right|, \tag{6}$$

where the values,

$$J_{ij} = \frac{\partial d_i}{\partial p_j}, \tag{7}$$

are the elements of the Jacobian matrix for the ***i*th** datum ***d_i*** and ***j*th** parameter ***p_j***, ***e_i*** is the noise estimate on the ***i*th** datum, and ***N_d*** is the total number of data. For parameter to which the data are particularly sensitive, the ***ParameterSensitivity*** will be larger than for other parameters.

The ***ParameterUncertainty*** measure for the ***j*th** parameter is the square root of the diagonal of the posterior model covariance matrix, and is given by,

$$U_j \;= \sqrt{diag([J^T W_d J + \; C_m^{-1} + Ws]^{-1}\,)}. \tag{8}$$

Where ***W_d*** is the inverse data covariance matrix and C_m is the prior model covariance matrix which includes all the model regularization terms. The ***ParameterUncertainty*** for well resolved layer conductivities will be less than for poorly resolved layer conductivities.

6.4 Outputs data files and headers

The inversion results are output to a separate space delimited flat ASCII column data file for each process. These are easily sorted and combined using the procedure shown in Section 6.4.3. The data files are accompanied by both a simple header and a more comprehensive ASEG-GDF2 header file.

6.4.1 Simple header

The simple header file is output in a file with extension ***.hdr***. An example header is shown in abbreviated form below. It simply lists a column number, or column number range for multi-band fields, and an associated field name identifier. This format does not show any units or formatting information but is useful for easy dechipering in simple scripting and MATLAB code.

```
1     uniqueid
2     survey
…
…
19    rx_pitch
20    rx_yaw
21    nlayers
22-51      conductivity
52-81      thickness
…
112        AlphaC
…
116        PhiD
```

```
…
122        Lambda
123        Iterations
```

6.4.2 ASEG-GDF2 header

The ASEG-GDF2 format is a standard point located data file format of the Australian Society of Exploration Geophysicists. Documentation describing the format in detail can be found online at https://aseg.org.au/Standards/ASEG-GDF2-REV4.pdf. The ASEG-GDF2 header file is output in a file with extension ***.dfn***. An example is shown in abbreviated form below. Compared to the simple header, the ASEG-GDF2 header provides more comprehensive information including the units of the fields, their formatting and other comments. Using the header file the inversion results can be directly imported into third party software including, ProfileAnalyst™, Intrepid™ and Geosoft™.

```
DEFN   ST=RECD,RT=COMM;RT:A4;COMMENTS:A76
DEFN 1 ST=RECD,RT=; uniqueid : I12 : Inversion sequence number
DEFN 2 ST=RECD,RT=; survey : I12 : Survey number

DEFN 19 ST=RECD,RT=; rx_pitch : F9.2 : UNITS = degrees , Rx pitch -
nose down +ve
…
DEFN 22 ST=RECD,RT=; conductivity : 30E15.6 : UNITS = S/m , Layer
conductivity
DEFN 23 ST=RECD,RT=; thickness : 30F9.2 : UNITS = m , Layer thickness
…
DEFN 25 ST=RECD,RT=; conductivity_uncertainty : 30E15.6 : UNITS =
log10(S/m)
DEFN 26 ST=RECD,RT=; EMSystem_1_ZS : 19E15.6 : EMSystem_1_Z component
secondary field windows
…
DEFN 28 ST=RECD,RT=; AlphaC : E15.6 : AlphaC inversion parameter
…
DEFN 38 ST=RECD,RT=; Lambda : E15.6 : Lambda regularization parameter
DEFN 39 ST=RECD,RT=; Iterations : I4 : Number of iterations
DEFN 40 ST=RECD,RT=;END DEFN
```

6.4.3 Sorting and combining results from all processors

The results from each computational process are output to separate data and header files. It is a simple process to sort and combine all outputs to one file using a batch file or shell script. The following two snippets, a WINDOWS batch file and a LINUX shell script, show how the output data files (***inversion.output.*.asc***) are easily sorted and combined into a single new file called ***.\lines\inversion.output.dat*** in a subdirectory. The sorting back into the original input file order is enabled because the first column of the output file is the record number of the input file. In these examples the header and log files from the first processor are copied and kept associated with the sorted and combined output file as headres and a record log of the inversion procedure. Examples of these batch and script files are provided in the relevant examples directories of the code repository (§6.5) in files called ***sort_sbs.bat*** and ***sort_sbs.sh***.

```
REM Example WINDOWS/DOS batch file for sorting and combining output

mkdir lines

copy inversion.output.*.asc lines\temp.txt
copy inversion.output.0000.log lines\inversion.output.log
copy inversion.output.0000.hdr lines\inversion.output.hdr
copy inversion.output.0000.dfn lines\inversion.output.dfn

cd lines
sort /REC 65535 temp.txt /O inversion.output.dat
del temp.txt
pause
```

```
#!/bin/tcsh
# Example LINUX shell script for sorting and combining output

if( !(-d lines) ) then
   mkdir lines
endif


sort -n inversion.output.*.asc > lines/inversion.output.dat
cp inversion.output.0000.hdr     lines/inversion.output.hdr
cp inversion.output.0000.dfn     lines/inversion.output.dfn
cp inversion.output.0000.log     lines/inversion.output.log
```

6.4.4 Cleaning up

Obviously one needs to be certain that their sorting and combining step has worked correctly before cleaning up the files from each individual processor or all their hard work will be lost. After successfully sorting and combining the output files the output files from each processor can easily be cleaned up using the batch files or shell scripts as shown in the following two snippets. Examples of these batch and script files are provided in the relevant examples directories of the code repository (§6.5) in files called ***cleanup.bat*** and ***cleanup.sh***.

```
REM Example WINDOWS/DOS batch file for deleting the outputs from
individual processors after sorting and combining outputs

del inversion.output.*.asc
del inversion.output.*.hdr
del inversion.output.*.dfn
del inversion.output.*.log
```

```
#!/bin/tcsh
# Example LINUX shell script for deleting the outputs from individual
processors after sorting and combining outputs

rm inversion.output.*.asc
rm inversion.output.*.hdr
rm inversion.output.*.dfn
rm inversion.output.*.log
```

6.5 Examples

Examples of the use of the deterministic inversion program are provided in the following repository directories;

1. ***examples/bhmar-skytem/galeisbstdem***,
2. ***examples/frome-tempest/galeisbstdem***,
3. ***examples/thomson-vtem/galeisbstdem***.

7 Stochastic inversion program

8 References

Bergeron, M. C. and Lawrence, M., 2010. Frome Airborne Electromagnetic (AEM) Mapping Survey Acquisition and Processing Report for Geoscience Australia. Fugro Airborne Surveys, Perth. **Unpublished**.

Constable, S. C., Parker, R. L. and Constable, C. G., 1987. Occam's inversion; a practical algorithm for generating smooth models from electromagnetic sounding data. *Geophysics* **52(3)**, 289-300.

Lane, R., Green, A., Golding, C., Owers, M., Pik, P., Plunkett, C., Sattel, D. and Thorn, B., 2000. An example of 3D conductivity mapping using the TEMPEST airborne electromagnetic system. *Exploration Geophysics* **31**, 162-172.

Leggatt, P. B., Klinkert, P. S. and Hage, T. B., 2000. The Spectrem airborne electromagnetic system—further developments. *Geophysics* **65(6)**, 1976-1982.

Ley-Cooper, A. Y. and Brodie, R. C., 2013. Inversion of Spectrem AEM data for conductivity and system geometry. *In: 23rd International Geophysical Conference and Exhibition*, Melbourne, Victoria, Australia, Australian Society of Exploration Geophysicists. Online: http://www.publish.csiro.au/paper/ASEG2013ab145.

Roach, I. C. (ed), 2012. The Frome airborne electromagnetic (AEM) survey, South Australia: implications for energy, minerals and regional geology. Geoscience Australia. **Record 2012/40**, 296 pp.

Smith, R. S., 2001a. On removing the primary field from fixed–wing time–domain airborne electromagnetic data: some consequences for quantitative modelling, estimating bird position and detecting perfect conductors. *Geophysical Prospecting* **49**, 405–416.

Smith, R. S., 2001b. Tracking the transmitting–receiving offset in fixed–wing transient EM systems: methodology and application. *Exploration Geophysics* **32**, 14–19.

Appendix A Example SKYTEM low moment system specification file

```
System Begin
  Name = SkyTem-Low-Moment
  Type = Time Domain

  Transmitter Begin
    NumberOfTurns = 1
    PeakCurrent   = 1
    LoopArea      = 1
    BaseFrequency = 222.22222222222222
    WaveformDigitisingFrequency = 3640888.888888889
    WaveFormCurrent Begin
        -1.000E-03 0.000E+00
        -9.146E-04 6.264E-01
        -7.879E-04 9.132E-01
        -5.964E-04 9.905E-01
        0.000E+00 1.000E+00
        4.629E-07 9.891E-01
        8.751E-07 9.426E-01
        1.354E-06 8.545E-01
        2.540E-06 6.053E-01
        3.972E-06 3.030E-01
        5.404E-06 4.077E-02
        5.721E-06 1.632E-02
        6.113E-06 4.419E-03
        6.663E-06 6.323E-04
        8.068E-06 0.000E+00
        1.250E-03 0.000E+00
    WaveFormCurrent End

  Transmitter End

  Receiver Begin
    NumberOfWindows = 18
    WindowWeightingScheme = AreaUnderCurve
    WindowTimes Begin
        0.00001539 0.00001900
        0.00001939 0.00002400
        0.00002439 0.00003100
        0.00003139 0.00003900
        0.00003939 0.00004900
        0.00004939 0.00006200
        0.00006239 0.00007800
        0.00007839 0.00009900
        0.00009939 0.00012500
        0.00012539 0.00015700
        0.00015739 0.00019900
        0.00019939 0.00025000
        0.00025039 0.00031500
        0.00031539 0.00039700
        0.00039739 0.00050000
```

```
          0.00050039 0.00063000
          0.00063039 0.00079300
          0.00079339 0.00099900
     WindowTimes End
     LowPassFilter Begin
          CutOffFrequency = 300000 450000
          Order           = 1      1
     LowPassFilter End
   Receiver End

   ForwardModelling Begin
     OutputType = dB/dt
     SaveDiagnosticFiles = no
     XOutputScaling = 1
     YOutputScaling = 1
     ZOutputScaling = 1
     SecondaryFieldNormalisation  =  none
     FrequenciesPerDecade = 5
     NumberOfAbsiccaInHankelTransformEvaluation = 21
   ForwardModelling End
System End
```

Appendix B Example SKYTEM high moment system specification file

```
System Begin
  Name = SkyTem-HighMoment
  Type = Time Domain

  Transmitter Begin
    NumberOfTurns = 1
    PeakCurrent   = 1
    LoopArea      = 1
    BaseFrequency = 25
    WaveformDigitisingFrequency = 819200
    WaveFormCurrent Begin
        -1.000E-02 0.000E+00
        -8.386E-03 4.568E-01
        -6.380E-03 7.526E-01
        -3.783E-03 9.204E-01
        0.000E+00 1.000E+00
        3.960E-07 9.984E-01
        7.782E-07 9.914E-01
        1.212E-06 9.799E-01
        3.440E-06 9.175E-01
        1.981E-05 4.587E-01
        3.619E-05 7.675E-03
        3.664E-05 3.072E-03
        3.719E-05 8.319E-04
        3.798E-05 1.190E-04
        3.997E-05 0.000E+00
        1.000E-02 0.000E+00
    WaveFormCurrent End
  Transmitter End

  Receiver Begin
    NumberOfWindows = 21
    WindowWeightingScheme = AreaUnderCurve
    WindowTimes Begin
        7.53900E-05 9.60000E-05
        9.63900E-05 1.22000E-04
        1.22390E-04 1.54000E-04
        1.54390E-04 1.96000E-04
        1.96390E-04 2.47000E-04
        2.47390E-04 3.12000E-04
        3.12390E-04 3.94000E-04
        3.94390E-04 4.97000E-04
        4.97390E-04 6.27000E-04
        6.27390E-04 7.90000E-04
        7.90390E-04 9.96000E-04
        9.96390E-04 1.25500E-03
        1.25539E-03 1.58100E-03
        1.58139E-03 1.99100E-03
        1.99139E-03 2.50800E-03
        2.50839E-03 3.15800E-03
        3.15839E-03 3.97700E-03
        3.97739E-03 5.00800E-03
        5.00839E-03 6.30600E-03
        6.30639E-03 7.93900E-03
        7.93939E-03 9.73900E-03
    WindowTimes End
    LowPassFilter Begin
        CutOffFrequency = 300000 450000
        Order           = 1      1
    LowPassFilter End
  Receiver End

  ForwardModelling Begin
    OutputType = dB/dt
    SaveDiagnosticFiles = no
    XOutputScaling = 1
    YOutputScaling = 1
    ZOutputScaling = 1
    SecondaryFieldNormalisation  =  none
    FrequenciesPerDecade = 5
    NumberOfAbsiccaInHankelTransformEvaluation = 21
  ForwardModelling End
System End
```

Appendix C Example VTEM system specification file

```
System Begin
  Name = VTEM-plus-7.3ms-pulse-southernthomson
  Type = Time Domain

  Transmitter Begin
    NumberOfTurns = 1
    PeakCurrent   = 1
    LoopArea      = 1
    BaseFrequency = 25
    WaveformDigitisingFrequency =  192000
    WaveFormCurrent Begin
        File = VTEM-plus-7.3ms-pulse-southernthomson.cfm
    WaveFormCurrent End
  Transmitter End

  Receiver Begin
    NumberOfWindows = 45
    WindowWeightingScheme = LinearTaper
    WindowTimes Begin
        0.0000180 0.0000230
        0.0000230 0.0000290
        0.0000290 0.0000340
        0.0000340 0.0000390
        0.0000390 0.0000450
        0.0000450 0.0000510
        0.0000510 0.0000590
        0.0000590 0.0000680
        0.0000680 0.0000780
        0.0000780 0.0000900
        0.0000900 0.0001030
        0.0001030 0.0001180
        0.0001180 0.0001360
        0.0001360 0.0001560
        0.0001560 0.0001790
        0.0001790 0.0002060
        0.0002060 0.0002360
        0.0002360 0.0002710
        0.0002710 0.0003120
        0.0003120 0.0003580
        0.0003580 0.0004110
        0.0004110 0.0004720
        0.0004720 0.0005430
        0.0005430 0.0006230
        0.0006230 0.0007160
        0.0007160 0.0008230
        0.0008230 0.0009450
        0.0009450 0.0010860
        0.0010860 0.0012470
        0.0012470 0.0014320
        0.0014320 0.0016460
        0.0016460 0.0018910
        0.0018910 0.0021720
```

```
          0.0021720 0.0024950
          0.0024950 0.0028650
          0.0028650 0.0032920
          0.0032920 0.0037810
          0.0037810 0.0043410
          0.0043410 0.0049870
          0.0049870 0.0057290
          0.0057290 0.0065810
          0.0065810 0.0075600
          0.0075600 0.0086850
          0.0086850 0.0099770
          0.0100851 0.0113498
     WindowTimes End

          //Notes
          //0.0099770   0.0114580 - real Gate 48 as per VTEM specs
          //0.0100851   0.0113498 - symetric altered window to prevent
linear taper extending into following half cycle
          //0.0099770   0.0112957 = non-symetric altered window to
prevent linear taper extending into following half cycle

   Receiver End

   ForwardModelling Begin

     OutputType = dB/dt

     XOutputScaling =  1e12
     YOutputScaling =  1e12
     ZOutputScaling =  1e12
     SecondaryFieldNormalisation  =  none

     FrequenciesPerDecade = 6
     NumberOfAbsiccaInHankelTransformEvaluation = 21

     SaveDiagnosticFiles = no

   ForwardModelling End

System End
```

Appendix D Example TEMPEST system specification file

```
System Begin
  Name = Tempest
  Type = Time Domain

  Transmitter Begin
    NumberOfTurns = 1
    PeakCurrent   = 0.5
    LoopArea      = 1
    BaseFrequency = 25
    WaveFormCurrent Begin
        -0.0200000000000    0.0
        -0.0199933333333    1.0
        -0.0000066666667    1.0
         0.0000000000000    0.0
         0.0000066666667   -1.0
         0.0199933333333   -1.0
         0.0200000000000    0.0
    WaveFormCurrent End
    WaveformDigitisingFrequency = 75000
  Transmitter End

  Receiver Begin

    NumberOfWindows = 15
    WindowWeightingScheme = Boxcar

    WindowTimes Begin
        0.0000066667  0.0000200000
        0.0000333333  0.0000466667
        0.0000600000  0.0000733333
        0.0000866667  0.0001266667
        0.0001400000  0.0002066667
        0.0002200000  0.0003400000
        0.0003533333  0.0005533333
        0.0005666667  0.0008733333
        0.0008866667  0.0013533333
        0.0013666667  0.0021000000
        0.0021133333  0.0032733333
        0.0032866667  0.0051133333
        0.0051266667  0.0079933333
        0.0080066667  0.0123933333
        0.0124066667  0.0199933333
    WindowTimes End

  Receiver End

  ForwardModelling Begin

    OutputType = B

    XOutputScaling = 1e15
    YOutputScaling = 1e15
```

```
     ZOutputScaling = 1e15
     SecondaryFieldNormalisation  =  none

     FrequenciesPerDecade = 6
     NumberOfAbsiccaInHankelTransformEvaluation = 21

   ForwardModelling End

System End
```

Appendix E Example GALEISBSTDEM control file for inversion of SKYTEM data

Appendix F Example GALEISBSTDEM control file for inversion of VTEM data

Appendix G Example GALEISBSTDEM control file for inversion of TEMPEST data