MULTMTRN: A Program for Multiple Station Analysis of Magnetotelluric Data

Gary D. Egbert*

March 13, 1998

1 Overview

This document describes use of the robust multiple station transfer function program described in Egbert, 1997 (Robust multiple station magnetotelluric data processing; GJI). The program is an extension of the robust single station and remote reference processing programs based on Egbert and Booker, 1986. Before running **multmtrn**, time series files must be processed by program **dnff** to convert to frequency domain Fourier Coefficient (FC) files. Note that these are the same FC files used by the single station/remote reference processing program **tranmt**. These files contain a series of FCs obtained from Fourier transforming a series of short overlapping time segments for each of a series of decimation levels. By careful use of **dnff**, short time windows from different stations which overlap in time will be coincident, and tagged with consistent *set numbers*. **multmtrn** uses these set numbers to align coincident sets. For details on how to use **dnff** (and **tranmt**), see documentation for **EMTF**.

To run the program a file (called **array.cfg** by default) is required. This file is roughly analogous to the **tranmt.cfg** file used by **tranmt**. This file tells how many "stations" (i.e., groupings of channels, each of which is contained in a separate file (or set of files)), names of FC files, output file root, and some preliminary "weights", which can be used as one way to emphasize certain channels in the definition of plane wave sources. In addition, there are a large number of command line options which control functioning of the program. These are described in greater detail below. Here we summarize the key points and features.

There are three (possible) goals of multivariate analysis with **multmtrn**. The first is to understand the character of the signal and noise. Can the situation be reasonably characterized by the standard quasi–uniform (plane wave) MT source assumption with incoherent noise added to each channel? Or is there evidence for consistent coherent noise? The program produces output which make it fairly simple to verify if the former case holds, and to gain some understanding of the nature of coherent noise. The program also produces output which can be used to generate "generalized transfer functions" for

*College of Oceanic and Atmospheric Sciences, Oregon State University, Corvallis; egbert@oce.orst.edu

more complicated (e.g., non–plane wave) sources. The second goal is to get the best possible plane wave MT transfer functions (this is really the ultimate goal–but in a sense it really follows logically after the first goal). For this purpose one has to make some decisions about what best defines the MT (plane wave) signal. The program will follow default rules (i.e., the strongest signal overall), but this can lead to very poor results. Often the user might know that one or more sites are (relatively) unaffected by coherent noise cultural noise, while other sites are seriously contaminated. In a case like this the user can provide information to the program which can significantly improve the final results. This is not something the program can do automatically. The user must provide this information through command line options and control files. A final use of **multmtrn** is to closely scrutinize badly contaminated data to try and pull out usable results. The program can help the user to identify the "bad" data (time segments and/or stations) in some cases. There are command line options which activate some of these features. These are very experimental, and are probably useful in certain specialized situations. It is not expected that many users will find these experimental features useful.

The program is reasonably flexible about the numbers and types of channels in each station (i.e., FC file). However, output formats for some files are really sensible only for more–or–less conventional 3–5 channel MV or MT sites. However, the program works with, and has been extensively used with, EM profiling data with multiple E_x and E_y channels, and even with non–EM data. It is also possible to have **E** and **H** channels in separate FC files. Note however: **multmtrn** breaks channels into *groups* for two purposes: (1) to estimate levels of incoherent noise all channels from different groups are used to predict the channels of a fixed group; (2) to estimate *local* transfer functions (e.g., impedances). For both of these purposes it is simplest to use the default channel groupings defined by the FC files. Furthermore, for some purposes it is only possible to group channels which are contiguous (i.e., next to each other in the ordering implied by the FC files and the order of FC files in the **array.cfg** file. Finally, for some purposes, the program expects that local magnetic channels H_x and H_y should be the first two channels in any station grouping. Thus, if possible follow the conventions established in **EMTF** and make sure that the horizontal magnetic channels occur first in the FC files.

In principal, the program works for one station ... but this only makes sense if there are multiple E or H setups (e.g., an EM profiling data set with 10 channels–e.g., 2 orthogonal H and 4 setups of orthogonal E.) Even in this case there may be problems (suggestion: run with the option –GA).

The program produces a variety of output files, some of which have the same format as the outputs produced by tranmt. Roughly these are of three sorts (1) files which characterize the full array response, giving all inter-station and inter-component transfer functions, plus signal and noise characteristics for all channels. These files are either ASCII files with explanatory headers, or there are matlab scripts to open and read these files, and to extract and plot the most commonly useful things. (2) so called Z–files which give local transfer functions, plus all information needed to calculate error bars in any coordinate system. In general there is one of these files produced for each station grouping with more than two channels, which are the predicting channels under the assumption of quasi–uniform MT sources. By default station groupings are determined by the FC files (one FC file = one group), but this can be changed with a command line option. The

format of these files is identical to that of Z–files produced by single station or remote reference analysis conducted with **tranmt**. Again there are matlab M–files for reading and plotting Z–files. See also the document *Z–Files*. (3) output files containing intermediate or auxiliary results which have been useful during development of the programs (and which may be useful for some advanced/experimental applications. These are in general poorly documented. None of these files are produced by default, and many of the command line options are to produce these files. Note also that several of the files output are redundant and will be eliminated in future releases. The input control file and the most significant output files are described further below.

A final warning: This program is complicated–in many places more complicated than necessary. It will be impossible to understand many of the options for **multmtrn** without a thorough understanding of Egbert (1997). Note also that significant computer resources might be required to run **multmtrn**, particularly for large data files, or if there are many stations/channels. The program is still under development (very slowly due to funding limitations), and is only partially documented. Some options are not described anywhere, and obviously not all combinations of options have been tested, or even make sense. There are almost certainly bugs which will show up when the code is used by others. Please notify *egbert@oce.orst.edu* if you run into problems. I stress again that used blindly, this program is quite capable of yielding quite dreadful results. User beware.

2 Making the Executable

There is a simple Unix Makefile for compiling and installing **multmtrn**. Before making the program there are several things to be aware of. First, maximum number of data channels (for one station), and maximum number of station need to be set in the header file **nstamx.h**. Most of the parameters that need to be changed most commonly are in this include file. Here is a brief description of the meaning of the parameters:

- *nstamx* Maximum number of *station*. Note that a station consists of all channels grouped together in a FC file. Note that there could be several FC files (same group of channels, but a different run) for one station.
- *nchmx* Maximum number of channels for a single station grouping
- *ngrpmx* Maximum number of groups for local TF computations. Normally this could be the same as *nstamx*, since each station is a natural (and the default) group for local TF estimates. But this might not always be true.
- *nchemx* Maximum number of predicted channels for a single station. Normally this would be *nchmx* - *2*, but if the channel groupings used for local TF computations are modified by use of the –s option, this could change. E.g., with one set of **H** channels, and multiple sets of **E** channels, one could estimate all impedances in a single file, all predicted by the same H_x, H_y. In this case one could have *nchmx* = *3*, but *nchemx* would have to be at least as large as the total number of **E** channels.
- *nbmax* Maximum number of frequency bands to compute estimates for. Actually this seldom needs to be changed.

There are other parameters which might have to be modified in **iosize.h** if the size of FC files (typical number of decimation levels, time segment lengths, etc.) are changed. If you use the FC program "as is" you generally should not need to change parameters in this file; otherwise some changes to this might be needed. Note that this file is basically the same as the parameter include file used for **tranmt** (of the same name). Any parameters in the include files not explicitly discussed here should be left set as they are.

- *ntfmax* Maximum number of data points to allow for for a single frequency band. Increase for very long time series. It also might be necessary to decrease the size of this to get the program to fit into memory, particularly in parameters like *nstamx* and *nchmx* are large.

- *ndmax* Maximum number of decimation levels to allow for. Should be set to whatever is used in **dnff**.

- *nfreqmax* Maximum number of FCs saved for a single decimation level. For example, with 128 point sets this would be at most 64, but is often set to less. Again, this should be set as in **dnff**.

- *ntpmax* Maximum number of FC files for a single station. Each FC file would correspond to a different "run" of the same set of data channels. Often this could be set to 1.

- *lpack* This is a logical parameter set to *.true.* when the FC files are stored in packed integer format. In this format one complex number is stored in 4 bytes. Set this the same way it is set in **dnff**. Packed format is the normal usage. However, for data recorded with 24 bits resolution, you might want to use standard binary format (i.e., set *lpack = .false.* in both **dnff** and here.

- *lfop* This is a logical parameter normally set to *.false.* Setting this to *.true.* makes the program open and close all FC files before and after every read. This is necessary on some systems when the number of stations in the array is too large, since some systems limit the total number of files that may be opened in a Fortran program.

A second point to note in making the executable is that the program links to the **lapack** library. Source code for this public domain library can be obtained from

http://www.netlib.org.

For completeness I have included the necessary routines in ../lapack, including the full set of "blas", or basic linear algebra subroutines. If you don't have lapack on your system, go into the source directories **blas_src** and **qr_src**, and make **libblas.a** and **libqr.a** . Then in this directory (MMT), edit the Makefile as indicated (in the Makefile comments) to make paths to the necessary libraries correct (if lapack is on your system, you might still need to edit the library path.) With all of this done, just type *make multmtrn*.

3 The array configuration file array.cfg

Here is an annotated array control file for "multmtrn" . This file tells which data files to use, and also controls relative weighting of normalized data vector components for determining the reference fields for plane wave (quasi–uniform) source transfer functions.

```
2                                <===  # of stations in the array
../CF/bs_nod.cfg                     <=== band set up file (as for tranmt)
1 10                              First station; number of FC files, # of ch.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1.        inverse weights for each channel
../FC/f10spa121.ts4                - FC file (one for each file for station #1)
S12                                - Site name (used in some output file names)
1 5                              Second station; number of FC files, # of ch.
1. 1. 1. 1. 1.                                as for station # 1
../FC/f5remw11.ts4
R02                                - Site name (end of station 2)
A12TS4                              <===  Array name ... used for output files
                                   <===OPTIONAL :
                                   <===OPTIONAL :
                                   <===OPTIONAL :
                                   <===OPTIONAL :
```

Note on optional arguments: As soon as the EOF is reached, the program stops looking for options. To specify the second optional argument, you also provide the first, etc. The optional arguments are:

- θ, rotation angle for x–axis for output in **array_name.cfg** file
- maximum number of iterations for robust iterations (obsolete)
- A series of lines to control set numbers to be included in analysis; The first line gives the number of lines ND to follow. For each of the following ND lines the range of set numbers to use for one decimation level is given as: id, iband(1,id), iband(2,id)

Note on weighting for each component: set inverse weighting to zero to omit a component from definition of reference (this is equivalent to assigning a very large inverse weight)

4 Output Files

Following is a brief description of the *standard* output files produced by multmtrn ... additional output files are generated when certain command line options are used. These are discussed separately. Also, there are MATLAB routines for reading all of these files. This allows for plotting, merging results from different processing bands/sites, reformatting for input into other programs, or whatever other manipulations might interest the user. This matlab interface is currently only in the initial stages of development, but if you're at all

familiar with matlab, this will be the easiest way to interact with these output files. See the separate matlab documentation and the help facility in the EMTF matlab directory for more details. (Note: the matlab functions and scripts use other functions and scripts from various of the subdirectories (matlab/IN, matlab/MISC, etc.). It will be easiest to put all of these subdirectories (i.e., matlab/* in the matlab search path (defined by the environment variable MATLABPATH, or by the matlab command "path").

Note that there are two general classes of output files: those which refer to array results (things which make sense for the whole set of stations together) and those which refer to a single station (e.g., MT impedance estimates). The latter sort of files are output for EACH appropriate station in the array. Where possible these have the same format as that used for output from "tranmt". This makes it relatively easy to merge results from array processing, with more conventional robust transfer function results.

Output files for array results all have a common root, with the file type identified by a 1–3 character suffix (e.g., *.M , *.TER , *.S0). Note that in early versions of this code the file type was put at the beginning of the file name (e.g., M_* , TERR_* , S0*). This change has been made for greater consistency with PC file naming conventions.

Note: There is an advanced feature of the code which allows for processing of "sub–arrays" ... that is, portions of the data which are available for only a subset of stations. We will not discuss this (only partially debugged) feature here, but note that when this feature is invoked, additional "sub–array" output files will be produced. For completeness we note these here. I cannot guarantee any functionality of any aspect of this advanced feature.

4.1 Standard array output files

Note that the file root **array_name** used in the example output file name examples given here will be replaced by whatever is given at the end of the **array.cfg** file (i.e., in the last required line of this file; see above).

4.1.1 array_name.M

This ASCII file gives eigenvalues for each period band, along with the vectors which span the coherence space (i.e., the columns of W in Egbert (1997). Output is in a tabular form, with numerous explanatory headers. Results are given in a separate table for each period. All "significant" eigenvectors are output; the number of vectors per band is variable. See the example file for further explanations. This file might be useful for an initial look, but is in an awkward format for further manipulation/plotting. Note that before outputting the columns of W, H components are scaled to be of order one, and E components are converted to the same units as H (nT) by assuming a 100 ohm m apparent resistivity (i.e., for plane wave sources over with an apparent resistivity of 100, E components will also be of order 1; for rho = 10**4, E components will be of order 10, etc.). For vectors which do not have much plane wave content, E could be much bigger than H, and things could get ugly Also, note that the output format is really only reasonable for "standard" arrays. In particular it was designed for MV arrays with many 3 channel sites. with the scaling described above, it also works OK for multiple 5 channel MT arrays. However, for

for more general arrays table headers are wrong, and some entries may fail to fit into the allowed format (so you'll get lots of *****). Formats can be changed by editing subroutine prteig.f .

4.1.2 array_name.S0

This is a "RAW" BINARY SDM file which contains all information needed to specify the array (station locations and IDs, orientations and IDs for each data channel, etc.), together with estimates of incoherent noise variances, and the SDM of the cleaned data. This file can be read by the matlab scripts **IN/sdm_init** and **IN/sdm_in**. Note that when the sub–array feature is used results for sub–arrays will be output in files named **array_name.S1**, **array_name.S2** etc. (Not well tested; probably does not work). Also note: In previous releases **array_name.S0** files contained the leading eigenvectors/values instead of the full SDM. These, along with canonical coherences/covariances, eigenvalues for subsets of channels, etc. are easily computed in matlab. An interactive/GUI set of scripts for analysis of the SDM in matlab using results output in this file. (If you have matlab: look at/try **sdm_plot.m**) This file is now the main full array output file.

4.1.3 array_name.SN

This ASCII files gives a frequency domain summary of all signal and noise spectra, including eigenvalues (but excluding eigenvectors). Useful for plotting signal and noise spectra; could be eliminated, since this info is also in **array_name.S0** . This file can be read into matlab with the command sn_in('array_name.SN'), and plots can be made in matlab using **sdm_plot.m**. See the EMTF matlab document for more details.

4.1.4 array_name.Pw

This binary file contains the full estimated "array TF" for an assumed plane wave source. All info necessary for computing TFs relative to any chosen pair of reference channels (along with error bars) is given in the file. Read with matlab routines **Pw_hd** and **Pw_in**. Matlab routine **Pw_plot** can be used to rotate and plot components of array TFs with error bars. The command line option –L suppresses this output.

4.2 Individual station files

4.2.1 Z–files

These files contain estimates of local transfer function. They are identical in form to the Z–files output by **tranmt**. By default they contain transfer function estimates relating the last $nch - 2$ channels at each site to the first 2 channels at that site. File names are determined from an input root specified in the **array.cfg** file, and a station identifier. Local TF files computed by **multmtrn** have the prefix **.zmm**. For a standard 5 channel MT station, the first two channels are the horizontal magnetic channels, so these files contain vertical field TFs, along with the impedance tensors. For an EMAP site all E–field TFs are given. In all cases everything is output in the MEASUREMENT coordinate system,

along with all necessary orientations. For each frequency band two complex Hermitian matrices are also output (one 2x2 matrix specifying the "input signal covariance", plus a $nch-2 \times nch-2$ matrix specifying the residual covariance for the predicted channels). Together these matrices can be used to compute error bars for components expressed in any rotated coordinate system. This file can be read into matlab with routine Z_in. Apparent apparent resistivities and phases can be calculated using matlab routine apresplt. See "*Matlab M-files for EMTF and multmtrn*" for details.
Also: see Z_files.ps , a postscript document file containing a more detailed description of the Z–file format
Note: With the option –s (see below under options) it is possible to change the default grouping of channels by stations, so that channels in separate FC files can be combined into a single group. This allows, for instance, for H fields in one file to be used as the "local reference" fields for one or more sets of E fields present in separate file(s).
Note: These error matrices are computed differently for different processing schemes (e.g., single station, Remote reference, multiple station), but formulae used to convert these matrices to error bars are identical for all schemes. Z–files can thus be merged across processing schemes.
Note: in the example here there are two Z–files produced, one for a 10 channel EMAP site (**S12_A12TS4.zmm**) and the other for a 5 channel MT site which was installed as a remote (**R02_A12TS4.zmm**).

4.3 Specialized output files

Output of these files is triggered by a command line option. These specialized/experimental files are described briefly along with the corresponding command line options below.

5 Running multmtrn

By default information about the array (number and location of data files, names for output files, some program control options) are found in the file **array.cfg** (e.g., see the example above). The name of this main program control file, and a number of processing and output features can be changed with the following command line options. Note that the following only provides a summary, using terminology and ideas developed in detail in Egbert (1997; "Robust Multiple Station Magnetotelluric Data Processing, GJI, in press"). Understanding this paper is more–or–less a prerequisite for understanding many of the options. Options marked with an asterisk (*) are unlikely to be of interest to most users, and are more for debugging/testing. Many of these options are only partially (often barely) tested, and are too hard for me to try explaining at this point. Most are probably not worth pursuing, and will probably disappearΦΦΦpear from future releases. Among these latter sort of options, not all possible combinations have ever been tried or even make sense.

6 Command Line Arguments

There are a number of command line options which either modify the way the estimates are computed, or modify outputs from the program.

6.1 General purpose command line arguments

−−

"multmtrn −−" generates a summary of usage and command line options

–f[array_file]

change default array file name to **array_file**

6.2 Arguments which control estimation options

–n

turn off all robust features (runs MUCH faster, but sometimes you get what you pay for)

–m

turn off robust RMEV (downweighting of outliers in individual data channels), but still do robust (rotationally invariant) SDM stack)

–i

change default max # of iterations for robust regression for each inner loop estimate of local noise; Default is set in main program as *itmax_ln_d*

–o

Change default # of outer loops for local noise estimation/individual channel outlier cleanup; Default is set in main program as *itmax_cln_d*

–I

change default max # of iterations of iterations used for robust regression for final TF estimate; Default is set in main program as *itmax_rrr_d*

–T

turn ON automatic timing error correction In this case a file called **array_name.TER** is sought; if this is found timing offsets are read from the file for each station and used to phase shift appropriate channels. If the file is not found, the program estimates the timing corrections and writes a file of this name. If times are all correct, the timing shifts should be zero. With default usage, multmtrn tries to open this file to find any needed timing shifts (in seconds).

If the file does not exist, the program estimates timing shifts by looking at the variation of phase with frequency of inter station magnetic field TFs near the Nyquist frequency. The estimated timing shifts are then output in this file, so any further runs for this array can skip the timing error estimation step. This file also provides a way for the user to intervene and manually adjust the timing correction. Note that timing corrections are now turned off by default; use the –T option to turn this on.

–N

Don't transform eigenvectors in **.M** output file (default is to output linear combinations of the first two eigenvectors for which the magnetic fields are, on average, polarized N–S and E–W.

–G[option]

change channel grouping for coherent noise variance estimation. option should be one of:

T $\Longrightarrow$ all components of a type at a single site (default)

S $\Longrightarrow$ all components at a single site

A $\Longrightarrow$ each component by itself

–g *

try to sort out plane wave/gradient sources geometrically (NOT FOR SMALL (# of stations OR spatial extent) ARRAYS!!!!)

–R#

Use projection of magnetic fields from station # into coherent signal/noise space to define plane wave reference.

–s[file_name]

change default definition of channel groupings for individual station TF output. By default channels are grouped by "stations", with all channels in a single FC file are assumed to be a single station. For each such station with at least 3 channels an individual station TF file (named **STA_ARRAY.zmm**) is output. With this option the groupings of channels to use for these Z–files are given in a file, denoted here as [file_name]. Here is an example used for an array in which the E and H channels were in separate FC files. There were 3 H setups (2 with 2 channels, 1 with 3), and 3 2 channel E setups, for a total of 6 FC files. The **array.cfg** file for this array looked like this:

```
6
../bs_nod
1 2
```

```
1. 1.
../../FC/H1.f2
S1H
1 2
1 1
../../FC/E1.f2
S1E
1 3
1. 1. 1.
../../FC/H2.f3
S2H
1 2
1. 1.
../../FC/E2.f2
S2E
1 2
1. 1.
../../FC/H3.f2
S3H
1 2
1 1
../../FC/E3.f2
S3E
TEST3
```

To have S1H and S1E together treated as a station (and S2H S2E, etc.), run multmtrn with the option –scgrp.cfg where file **grp.cfg** contains the following text:

```
3                <===== # of channel groupings
S1               <===== name for first group
4                <===== number of channels in first group
1 2 3 4          <===== channel numbers; following order in array.cfg
S2               <====  name for second group ... etc.
5
5 6 7 8 9        <===== channel numbers for group two; files f3H2, f2E2
S3
4
10 11 12 13
```

To always use the two H channels at the first site for a reference, and to omit the Hz TF at site 2 **grp.cfg** would be changed to:

```
3            <===== # of channel groupings
S1           <===== name for first group
```

```
4             <===== number of channels in first group
1 2 3 4       <===== channel numbers; following order in array.cfg
S2            <===== name for second group,  etc.
4
1 2 8 9       <===== channel numbers, group two; for files f3H2, f2E2
S3
4
1 2 12 13
```

Note in particular that channels may be used in multiple groups, or omitted.

–M[SNR_min]

Only use sets with signal to noise ratio (as determined from 2 dominant eigenvectors) exceeding input argument SNR_min

6.3 Arguments which add to or change output files

–L

Don't output **array_name.Pw** file. By default this file is output, but it is often not used for routine processing.

–z

Rotate channels into a common coordinate system before outputting in **array_name.M** file. Coordinate rotations require channels to be paired. When this option is invoked the program tries to make reasonable assumptions about channel pairings. In particular it assumes that an H_x is always followed by an H_y, and an Ex by an E_y. If this isn't the case (e.g., in an EMAP type configuration) things will be screwed up.

–c[option]

where options is one of H, R, or N ... output canonical coherence file. The file will be named **array_name.CC**. The options essentially control normalization of channels...are as follows:

R = canonical coherence, the default if only –c is specified, normalizes each data channel by the square root of its total variance.

N = canonical covariance, normalizes each channel the estimated incoherent noise scale

H = canonical covariance also, but with H channels expressed in nT, and E channels scaled to nT using a crude estimate of the array average impedance

The output file has the format:

$\Longrightarrow$ ASCII

$\Longrightarrow$ one header record with two integers : total number of channels (all stations, number of frequency bands)

$\Longrightarrow$ followed by one more "record" for each frequency band : period, canonical coherences (or canonical covariances) for each station relative to all other stations. In general there are as many canonical coherences at each site as there are data channels. Thus, with 3 5 channel sites, the first 5 channels are coherences of site 1 relative to sites 2&3, the next 5 are for site 2 relative to 1&3, etc. Up to 15 coherences are plotted on each line. Multiple lines are used if the total number of constituents exceeds 15. If there are more components at one site than there are at all other sites combined (e.g., 2 sites, one with 10 the other with 5 channels, as with remote reference EM profiling) some coherences will be zero.

–C

output correlation matrices for data, and for all incoherent noise groupings. (For incoherent noise groupings, the correlation matrix is estimated from the correlation among residuals for all channels in each group, as predicted by all channels in all other groups. Note that the correction applied to variances to make these nearly diagonal is not allowed for in this calculation).

The output file has the format:

$\Longrightarrow$ ASCII file ... with lost of "informative" headers ... self explanatory?

* –P

Output principal component (PC) "time series" file **array_name.PC**. The frequency domain array data vectors bfX_t can be expanded as a linear combination of the significant PCs:

$$\mathbf{X}_t = \sum a_{jt}\mathbf{W}_j \tag{1}$$

The PC file contains the coefficients a_{jt}, which can be used to reconstruct the coherent part of the Fourier coefficients for all channels and time segments.

The output file has the format:

$\Longrightarrow$ Binary ; see output routines in **pc_out.f**.

* –t

Output transfer function files used in local noise variance estimation

* –r

Output the **array_name.PC** file AND output the raw array data vectors bfX_i (in a single file, still called **array_name.PC**).

The output file has the format:

$\Longrightarrow$ Binary ; see output routines in **pc_out.f**.

6.4 Arguments for coherent noise downweighting

All this stuff is very experimental, and will only be useful in some special cases, e.g., as discussed in Egbert, 1997.)

–a *

omit data sets specified at end of array.cfg file only on first coherent noise downweighting iteration; on subsequent iterations coherent noise/signal ratios are used to decide which data should be omitted (This is a way to use the coherent noise downweighting feature when a time window of "clean" data can be specified a priori. On the first only data sets in a specified time range are used for initial processing; based on initial results, a preliminary separation of the total coherent signal into coherent noise and MT signal is made. Based on this separation weights are determined for all data sets, and a refined estimate is computed.

–u *

assume signal and coherent noise are uncorrelated for computing weights

–w# *

change default number of iterations for coherent noise downweighting to #

–p# *

change default cutoff for coherent noise downweighting to #

7 Problems

The ASCII output file **array_name.M** is a good place to look to see if everything is working. In particular inter station magnetic TFs should be close to one. If everything looks garbled, and your sure the data is pretty good, timing errors are a likely problem. Double check that timing is OK, first by plotting. If this checks out, run with the –T option. This should correct small timing errors, which can be hard to see in the time series even when they are significant enough to mess up inter station TFs significantly. Also: make sure channel orientations are correct in FC file headers!

If there are problems with run time errors, first make sure the parameters in nstamx.h are set correctly. My experience is that this is the most common cause of run time errors (and sometimes of garbled results, even when execution terminates normally).

the –C option output correlation matrices ... so you can check the simple coherence between any pair of channels.)

8 Contact for Questions/Updates

Questions, comments and requests for updates can be addressed to:

Gary D. Egbert
Associate Professor
College of Oceanic and Atmospheric Sciences
Oregon State University
Oceanography Admin Bldg 104
Corvallis, OR 97331-5503

email: egbert@oce.orst.edu
phone: (541) - 737 - 2947
fax: (541) - 737 - 2064