Robust Single Station, Remote Reference and Array Processing of MT24 Data

Markus Eisel[1]

August 30, 1998

[1]College of Oceanic and Atmospheric Sciences, Oregon State University, Corvallis

Contents

Introduction

In the past years more sophisticated methods for the processing of magnetotelluric (MT) time-series were developed to account for the problems arisisng from cultural noise. These developments include the remote reference (RR) method, data adaptive weighting (robustness), and new approaches like the simultaneous processing of all data recorded at several stations at the same time, the so-called multiple-station, or array processing (Egbert, 1997).

The MT24 system with its modular design and the capability of recording a large number of channels in arbitrary field setups, together with the possibility to synchronize systems at different locations via GPS produces datasets which require non-standard processing techniques and appear ideal candidates for the newly developed techniques.

The well established and widely used processing programs developed by Dr. Gary Egbert (1987, 1989, 1997) are adapted to the data format and structure of the MT24 system and integrated into the PC based MT24 data acquisition program. The complete processing package consists of three different programs.

- *DNFF* : windowing and Fourier-transformation program
- *TRANMT* : robust single site and remote reference MT transfer function estimation program
- *MMT* : robust multiple station transfer function estimation program.

This manual is divided into two parts. The first parts describes the usage of the programs, the input parameters and files, command line switches and the output files. Part two describes the algorithms used and highlights possible changes of the source code to adapt the codes to special cases.

Chapter 1

Users Manual

As the MT24 acquisition program so run all three processing programs under it Win95. All three programs can be started from the graphical user interface (GUI) of the MT24 data acquisition program *ACQ24* , . The performance of the routines is ruled by several configuration files which all have the extension *.cfg* and reside in the directory `C:\proc24\cfg` . Though the GUI uses a set of default parameters for configuration the user might want to choose different parameters for the processing. In any case it is possible to run the programs from a *DOS* command line window.

1.1 General Processing Concept

Before starting to describe the details of the processing programs it might be useful to give the user some information about the general concept on which these programs where developed. Both transfer function programs, *TRANMT* and *MMT* use the same Fourier-coefficient (FC) files as input, created by *DNFF* . As both programs are developed for processing data from simultaneous recordings at diffferent sites which did not start necessarily at the same time, a special procedure is implemented to allign simultaneously recorded segments properly. In additon the processing programs allow processing of subsets of the total Fourier-transformed time-series, specification of subsets of channels and regrouping of channels. Having this in mind the most time- and mass-storage-efficient way to perform the processing is to Fourier-transform complete runs including all channels of one run into one

FC-file. Lateron one can use the flexibility to define which channels from different runs and which time-windows shall be processed. Because the *ACQ24* interface probably does not support all these features this manual tries to highlight and describe these possibilities.

1.2 DNFF: Windowing and Fourier-Transformation

The program *DNFF* builds the direct interface to the MT24 dat format and structure. It is customized to read the binary MT24 time-series files and all additional information from either the *line definition file* (LDF), created with the *ACQ24* GUI or from the *run-file* and a configuration file which is described below.

DNFF Fourier-transforms the time-series based on a cascade-type decimation scheme using short overlapping time windows in several decimation levels to cover a wide frequency/period band. A key feature of *DNFF* is the aligning of time series from different sites/runs. This is essential for the remote reference and multiple station processing. Based on a survey reference time, stored in a file named *survey.clk* (in directory `C:\proc24\cfg`) time series which need not be started simultaneously are aligned exactly in time. **Make sure the survey start time is earlier than all run start times which are to be processed**.

1.2.1 Configuration Files

In order to run *DNFF* (either from the *ACQ24* program or from the command line) the following configuration files must exist and should be checked for proper configuration. All configuration files are expected to reside in `C:\proc24\cfg` .

- *survey.clk* : The *survey.clk* file defines the reference time for all data files to be processed. From the time difference between the reference time and the actual start time of the recording, based on the chosen time-window length and window-overlap (see below at *decset.cfg* , a unique set number for each time-segment is calculated to properly align recordings from different sites/runs even though they did not start simultaneously (but overlap).

The survey reference time is given in an ASCII file in 6 integer numbers as YEAR MONTH DAY HOUR MINUTE SECOND. Her is an example of this file

```
1997 11 13 12 00 00
```

- *emi-dnff.inp* : The *emi-dnff.inp* file tells *DNFF* which data to process and which additional configuration files to use. The following example serves to explain the structure of the *emi-dnff.inp* file:

  ```
  EMI ACQMT-24
  d:\mt24\default.srv
  d:\mt24\hc02\01Hmc02\01Hmc02.b02
  d:\mt24\hc02\01hmc02\hocm01.ldf
  c:\bin_mt24\decset_4.cfg
  c:\bin_mt24\pwset.cfg
  ```

  The first line is an identifier for *DNFF* to properly decide which type of data to process. The second line is the name of the survey file of the recording. Line three holds the name of the run-file which has information about the data to be processed. The name of the LDF is in line four. Lines five and six are the names of the configuration files for the decimation level parameters and pre-whitening parameters. In this form of the *emi-dnff.inp* file almost all information is read from the LDF. The resulting FC-file will be named according to the line-ID defined in the LDF with wxrtension '.fnn', where nn is the number of channels combined into the FC-file.

  There is an alternative form of the *emi-dnff.inp* file which is shown in the example above. This type of *emi-dnff.inp* file does not depend on the LDF and reads all information from files which are created during the acquisition process.

  ```
  EMI ACQMT-24
  d:\mt24\default.srv
  ```

```
d:\mt24\hc02\01Hmc02\01Hmc02.b02
8
d:\mt24\hc02\01Hmc02\02340001.t02
d:\mt24\hc02\02340001\02340002.t02
d:\mt24\hc02\02340001\02340003.t02
d:\mt24\hc02\02340001\02340004.t02
d:\mt24\hc02\02340001\02340005.t02
d:\mt24\hc02\02340001\02340006.t02
d:\mt24\hc02\02340001\02340007.t02
d:\mt24\hc02\02340001\02340008.t02
c:\bin_mt24\decset_4.cfg
c:\bin_mt24\pwset.cfg
```

In this type of *emi-dnff.inp* file the entry of the LDF is replaced by the number of channels to be FFT'd and a list of the time-series data file names. This file is easy to create without the *ACQ24* interface. *DNFF* reads all necessary information from the run-file and the data files. The FC-file name will be the name of the run-file with extension '.fnn' where nn is the number of channels in the FC-file.

- *decset.cfg* : The *decset.cfg* file specifies the number of decimation levels and parameters conneted with the decimation and storage of FCs. Parameters include the number of samples in the subwindows, the overlap of subwindows, decimation factors, filter coefficient for low-pass filtering and the number of FCs to store in the FC-file. Her is an example of a *decset.cfg* file:

```
4   0
128   32.    1    0    0    7    4    51  1
1.0000
128   32.    4    0    0    7    4    51  4
.2154 .1911 .1307 .0705
128   32.    4    0    0    7    4    51  4
.2154 .1911 .1307 .0705
128   32.    4    0    0    7    4    51  4
.2154 .1911 .1307 .0705
```

The first line gives the number of decimation levels and an offset to add to decimation level numbers - i.e. the first decimation level will be numbered *1+offset*. *offset* can be positive or negative, though in most cases *offset = 0*. But there are special cases where this might be changed. (As an example: Data sampled at 40 Hz at a permanent station is used for a remote reference for 10 Hz data for an MT survey. One way to do this would be to decimate the 40 Hz data by 4, and set *offset = -1*. Then decimation level 1 would be sampled at 10 Hz for both local and remote). For each decimation level there are two lines of parameters. For the first of these (e.g. the second line in this file, which begins 128 32. 1 ...) the parameters are, in order

(1) Number of points in window (128 for all decimation levels here)

(2) Number of points of overlap for adjacent sets (32.) (note that this is of type real and need not be a whole number; this can be used to keep data windows for different sampling rates "lined up". One of those features most users won't care to even think about.)

(3) Decimation factor (1 for level 1 (undecimated); 4 for other levels)

(4) Offset for centering decimation filter (0)

(5) Offset for starting sets (0)

(6,7) Parameters which define how missing data is treated (7,4)

(8) Number of Fourier coefficients (FCs) to save; only the lowest frequency FCs will be output. In the example given here there will be 64 FCs produced by FFTing a single 128 point data segment. By setting this parameter to a value less than 64 , FCs which are worthless or redundant (due to overlap of frequency range covered by adjacent decimation levels) can be eliminated from output file. Also, in the example cited above with 40 Hz and 10 Hz data, one could set the number of FCs to save for the first decimation level (numbered zero after adding *offset = -1*; In this case the FT will be skipped for this decimation level). to zero.)

(9) Number of filter coefficients for filtering before decimating to THIS level

The second line for each decimation level (3rd, 5th, 7th 9th lines in file) gives the filter coefficients for the digital low pass filters needed for filtering before decimation to THIS level.

Based on the number of samples specified in the LDF file *DNFF* checks if the number of decimation levels specified in the *decset.cfg* file is reasonable, i. e. if the time series is long enough to decimate it sa often as specified. If the number of sets in a decimation level falls short of a certain threshold (hard-coded in the program, see section 'Programmers Manual' for changing this) *DNFF* resets the number of decimation levels.

- *pwset.cfg* : The *pwset.cfg* defines the method for the pre-whitening. A pi-prolate window is used for tapering the time series data before FFTing. This window does not always provide sufficient protection against spectral leakage without pre-whitening. The program allows three options for pre-whitening, set by the flags in the *pwset.cfg* file – (1) no pre-whitening; (2) first difference pre-whitening; (This works fine for long period data (e.g. 5s sampling rate, 20 - 10000s periods)); and (3) adaptive, autoregressive pre-whitening. Normally it is safest to just use option 3, with the length of the AR filter set to 3 or 4. In the current version it is not necessary to specify the pre-whitening flag for each channel and decimation level. The first line specifies the number of decimation levels and the number of channels. Subsequent lines give the flags for each decimation level and each channel. *DNFF* issues a warning if there is a disagreement between the number of decimation levels specified in *decset.cfg* and *pwset.cfg* but popceeds using the number of decimation levels specified in *decset.cfg* . Here is an example of a *pwset.cfg* file:

  ```
  4 5
  -1 -1 -1 -1 -1
  -1 -1 -1 -1 -1
  -1 -1 -1 -1 -1
  -1 -1 -1 -1 -1
  ```

  The first line gives the number of decimation (4) levels and the number of channels (5). For each decimation level follows one line specifying the pre-whitening flags for each channel.

- *bad-record-file:*

1.2.2 System Parameters

TO perform a proper Fourier-transformation *DNFF* needs a number of parameters which describe the field-setup, the system type, etc. Depending on the type of the *emi-dnff.inp* file these parameters are either read from the LDF or from the run-file. *In both cases it is essential that at recording time the the system description tables are filled in properly. Wrong or missing specifications of parameters like orientation, gains, or filter types will lead to erratic processing results and might require a re-transformation with revised settings.* Even though it is not obvious in the first place, a proper entry for the station coordinates is very useful in the context of the multiple-station program. Refer to the *ACQ24* manual for setting the system and field-setup parameters.

1.2.3 Input Files

The input data are the time series files created by the MT24 data acquisition program. Each channel is stored in a single file which consists of a header of 4096 bytes followed by the data in an unformatted four-byte integer stream. Which data channels of a single run are to processed is specified by either the LDF or the *emi-dnff.inp* file.

1.2.4 Output: Fourier-Coefficient Files

The output of *DNFF* are the binary, fixe-record-length Fourier-coefficients (FCs) of the time-series of each channel and each time segment of points specified in the *decset.cfg* file (in the above example 128 points). In the remainder of this manual these time segments are referred to as **sets**. The FCs are orderend by frequency. In order to save storage space *DNFF* compresses the FCs by deviding by a common factor. For each set a unique set-number is calculated based on the *reference time* and start time of the recording so that simultaneously recorded data at different sites, Fourier-transformed and stored in different FC files can be alligned properly. The set-number is stored with each set. A general header preceeds the FC records holding information of number of channels, number of decimation level, sampling rates, channel names, coordinates, orientation, etc. This is the complete information encessary for the subsequent processing. For each frequency exist a frequency header holding the scaling factors, number of sets and a pointer to the record of the following frequncy. Both, *TRANMT* and *MMT* read the same FC files.

The structure of the FC file is as follows:

```
   -----------------------------------------------------------
  |                   File Header
  |___________________________________________________________


             ---------------------------------------------
            |                    Header record for frequency #1
            | Fourier            Record #1
            | Coefficients           .
            | for frequency          .
            |     # 1                    .
            |                    Record #N1
            |______________________________________________
            |
Decimation
Level #1
            |
            |
            |
            |
            |_____________________________________________
            |                    Header record for frequency #n
            | Fourier            Record #1
            | Coefficients           .
            | for frequency          .
            |     # 1                    .
            |                    Record #Nn
            |______________________________________________
             ----------------------------------------------
            |
            |   A similar block is repeated for each decimation level
            |
```

Each record is of length $(nch+1)*4$, where nch is the number of channels. The sequence of the channels is reordered by *DNFF* in a way that the the first two channels in the FC file are a pair of horizontal magnetic channels, namely Hx and Hy. The third channel will be Hz if available. The remaining channels are in the sequence given by either the run-file or the LDF. If multiple H-channels exist, *DNFF* moves the Hx and Hy appearing first in the list to positions 1 and 2. This specific sequence – Hx ,Hy ..... – is expected by the transfer function programs by default. Nevertheless, cases were some FC files contain electric channels only and others only magnetic channels can be handled as well.

The Fourier coefficient files are called **test1.f#** where # is an integer giving the number of channels of data. Note that the Fourier coefficients are kept in the original measurement coordinate system, with all filter corrections, count conversion etc., applied. Magnetic field Fourier coefficients are output in units of $nT(hz)^{-1/2}$ and electric field FCs are output in units of $(mV/km)(hz)^{-1/2}$. The header contains all necessary information needed to correct for sensor orientations, local magnetic declination, etc (provided the user inputs this info!).

By default the FCs are stored in a compressed 2-byte floating point binary format. The range of the floating point representation is determined automatically (and separately for each frequency) by the output program. When read by the proper routines, the FCs are complex numbers with proper physical units. This compression of the data can be suppressed by setting the logical parameter *lpack* to .false. in the include file **iosize.inc**. With *lpack* = .false. FCs are output as standard 8 byte complex numbers (i.e., two 4 byte reals). See the section on making dnff and changing parameters above. NOTE: if you change *lpack* here, you need to change it also for programs that read the FC files (i.e., *TRANMT* and *MMT*).

1.3 TRANMT

TRANMT calculates robust transfer functions (TFs) between two input channels (generally two orthogonal horizontal magnetic field channels) and a number of output channels in the frequency domain. The data channels can either belong to one site (single site) or two a local and a remote reference site (RR). The data input are the Fourier-coefficient files created by *DNFF* . The general philosophy behind the estimation scheme is described in Egbert and Booker, 1986. In addition there is an automatic "leverage control" feature (e.g., Chave and Thomson, 1989), and an option to use a hybrid coherence sorting/regression M-estimate, as described by Egbert and Livelybrooks. Error bars are computed using the asymptotic approach described in Egbert and Booker (1986).

1.3.1 Configuration files

TRANMT requires three configuration files. The major defines the FC files to process. It's default name is *tranmt.cfg* . When using the *ACQ24* interface program, this file is generated from the definitions in the chosen setup and LDF. The second file (*options.cfg*) passes a number of processing options to the program while the third one defines the frequnecy band averaging (*bs.cfg*). The names of the two last files are arbitrary. Default versions of these files are provided in the `C:\proc24\cfg` directory. In addition to these three mandantory files there are optional files through which the performance and output of *TRANMT* can be controlled. These are discussed in the section 'Command Line Options'. The following sections dexscribe the input files in more detail to outline the capability and flexibility of *TRANMT* .

- *tranmt.cfg* : The following example is a simple *tranmt.cfg* file for the single site processing of one typical MT FC file.

```
test
c:\\bin_mt24\\options.cfg
1
1 5
test.f5
n
```

Explanations for the entries referenced to the line numbers:

(1) A character string which is used as the root for the output file names. See section 'Output Files' for details about these files.

(2) is the name of the *options.cfg* file, generally with the full path name.

(3) gives the number of stations to process.

(4) holds the number of FC files for the first (and in this case the only) station and the number of channels of this site.

(6) is the name of the FC file

(7) n = no for not processing another site/set.

Here is a modified form as used in a RR case:

```
loc_ref_1
c:\\bin_mt24\\options.cfg
2
2 5
local_1.f5
local_2.f5
1 5
ref_1.f5
y
ref_loc_1
c:\\bin_mt24\\options.cfg
2
1 5
ref_1.f5
2 5
local_1.f5
local_2.f5
n
```

(3) The 2 specifies that data from two sites will be processed.

(4) The 2 tells *TRANMT* that there are two FC files from the first site

(5, 6) The names of the two FC files for site 1

(7) The number of FC files and number of channels of the second (the reference) site.

(8) The FC file name of the reference site

(9) a 'y' for yes that there is more to be processed.

(10-20) A section similar to the first with interchanged local and reference sites. This *tranmt.cfg* will basically run *TRANMT* twice. An even more unusal case will be discussed in conjunction with the *group.cfg* file in section 'Command Line Options'.

If *TRANMT* is running in RR mode or SS mode is controlled by the *options.cfg* file described below. In general *TRANMT* can process a number of sites – and FC files – in one run. If the SS flag is set, TF for all channels are calculated in reference to the first two channels of the first site, which are considered to be a pair of horizontal magnetic field channels. All TF are written to one Z-file. When running in RR mode the first two channels of site 1 are still considered as input channels and site 2 is considered to be the reference site. In this case the first two channels of site two are only used as reference channels and TFs are calculated for all but the first two channels of site 1, relativ to the two input channels (channel 1 and 2 of site 1). All this can be changed by using the -s command line option described below.

- *options.cfg* : The *options.cfg* file defines a number of parameters which control the behaviour of *TRANMT* . The name of the file is arbitrary and passed to *TRANMT* via the *tranmt.cfg* file (including the path to it's location). Standard *options.cfg* files for SS (*opts_ss.cfg*) and RR (*opts_rr.cfg*)processing are in the `C:\proc24\cfg` directory, but the user can create especially taylored derivatives of these and store them under different names in different directories. Here is a sample *options.cfg* file:

```
Robust Single station         <--     comment header
                              <--     FC input directory path
                              <--     output directory path
c:\\bin_mt24\\bs_4.cfg        <--     band averaging file
y                             <--     y/n robust/LS
n                             <--     y/n remote reference/single site
n                             <--     y/n e-field ref
```

```
n                            <--    y/n output coherence vs set no.  ...
                             <--     (if yes provide file name on this line)
0. 0. 0. 0. 0                <--    coherence sorting parameters
```

Here are some more explanations, keyed to the line numbers:

(1) Comment header line

(2) Path to the directory of the FC files. If the FC files are in the directory where *TRANMT* is run, this line is empty.

(3) Path to the directory where the output files shall be stored (see line 2 for the case of the working directory).

(4) Name of the band averaging file with full path name.

(5) yes/no switch for robust features

(6) yes/no switch for remote reference (no = single site)

(7) use electric fields as input (not supported in this version).

(8) yes/no switch for output of coherences vs set numbers. If yes provide a file name on the following line. This is sort of a ddebugging option.

(9) Coherence sorting parameters. Four real numbers and one integer are required in general. These are (in order):

    coh_target, cohp(1), cohp(2), coh_min, nu_min

    These parameters are used to specify coherence cut off levels for coherence pre-sorting. In this scheme coherence is calculated for wide frequency bands for each time segment, and only time segments which achieve a specified minimum coherence are used for further processing. This feature is most useful for single station dead-band data where noise in the magnetic components can be large enough to cause serious bias problems with single station estimates. See Egbert and Livelybrooks, Geophysics, 1996 for further discussion and justification. The five parameters specify a scheme for determining coherence cut off levels, which adapts to the number of data points and typical coherence levels. The scheme tries to trade off (in an ad hoc manner) between reducing bias and variance - i.e., we want to use only segments of "high enough" coherence, but at the same time keep a reasonable number of degrees of freedom in the estimates. If all paramters are zero, this feature is inoperative. If you use this feature, you will have to experiment with different parameter values to get a reasonable tradeoff between keeping enough data, and getting rid of enough noise.

Here is the meaning of the 5 coherence sorting parameters :

(1) *coh_target* is the target coherence; ideally we would like all time segments to achieve this coherence. (e.g., coht = .95)

(2-3) *cohp* two real parameters used to determine a target number of degrees of freedom in the transfer function estimate via :

> nu_target = cohp(1)*nu**cohp(2)
> where nu is the number of points available (e.g.: cohp(1) = 3., cohp(2) = .5)

(4) *coh_min* minimum acceptable coherence level. (e.g., .8)

(5) *nu_min* - minimum number of data points (e.g., 20)

If possible we would like to use at least nu_target data points, all from time segments with coherence at least coh_target. If there are not nu_target points in sets with coherence above the target coherence, we accept lower coherence sets, until nu_target points are available, or until coh_min (the minimum acceptable coherence) is reached. In general we don't accept sets with lower coherence unless this is necessary to get the minimum number of data points, nu_min.

- *bs.cfg* : This file defines the frequency band averaging and target frequencies at which transfer functions are estimates. This file is required. The name of the file is given in the *options.cfg* file, so the file name is arbitrary.

  An example band set up file:

```
28                                 No. of bands
```

```
1 34 45      band 1: dec. level; band limits (low & high)
1 27 33
1 21 26
1 16 20
1 13 15                       .
1 10 12
1 8 9
1 6 7                         .
2 18 22
2 14 17
2 11 13                       .
2 8 10
2 6 7
2 5 5
2 4 4
3 11 13
3 8 10
3 6 7
3 5 5
3 4 4
4 11 13
4 8 10
4 6 7
4 5 5
4 4 4
4 3 3
4 2 2
4 1 1
```

Note that this file has the frequencies ordered from highest to lowest with no overlaps (i.e. a reasonable decimation level has been chosen for each frequency band). In the output file, results for the bands will be printed in this order. By varying the form of this file any bands can be printed out in any possible order.

1.3.2 Command Line Options

There are seven command line options which can be used separately or in combinations when running *TRANMT* . When running *TRANMT* from the

DOS-shell the syntax is:

```
c:my\_data\_dir> tranmt -C<string>
```

where *C* is one of the following characters and *string* represents additional data which is required by some of the command line options (a file name a single integer or a set of integers).

- `-f<config-file>`: With the -f option the user can specify a different major configuration file. By default, *TRANMT* searches for a file named *tranmt.cfg* , first in the working directory and if not successful in the `C:\proc24\cfg` directory. `config-file` may include a path to a different directory.

- `-s<group-file>`: The -s option allows to specify a different grouping of the channels *TRANMT* is processing. The default is that the first two channels of site 1 are the predictors (input channels) and all remaining channels are predicted channels (output) in the SS case, in RR mode the first two channels of site 2 are the reference channels (and other channels of site 2 are ignored). All TFs of the outpur channels are written to a single Z-file. To change this default behaviour one can specify a *group.cfg* file of the following form:

  ```
  3 4               <--     input (predicting) channel numbers
  2                 <--     number of output (predicted) channels
  6 5               <--     output (predicted) channel numbers
  7 8               <--     remote reference channel numbers
  ```

  Here are some examples to ellucidate this: (1) consider an EMAP spread with a large number of electric field channels (e. g. 15) and two pairs of horizontal magnetic field channels, all packed into one FC file. For test purposes one wants to process the data one time with the first H-field pair as input channels and the second pair as reference channels, and in a second processing with interchanged H-pairs. I. e. the *tranmt.cfg* file looks like this:

```
test
c:\\bin_mt24\\options.cfg       <-- the options file declaring RR
1                               <-- only one site
1 19                            <-- with 15 E and 2 x 2 H channels
test.f19
n
```

Assume the sequence of channels to be HX1, HY1, EX1, EX2, EY1, EX3, EX4, EY2, EX5, EX6, EY3, EX7, EX8, EY4, EX9, EX 10, EY5, HX2, HY2. The corresponding re-grouping files for the processing described above would then be:

```
1 2
15
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19
```

and

```
18 2
19
15
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
1 2
```

Another case where this command line option is of great use is in the case where e. g. only E-fields were recorded at one site and only H-fields at a close by site. instead of mixing the separate sties together into one FC file one can use the `-s` option.

- `-x`: this ooption forces *TRANMT* to write the spectral density matrix (SDM) to a separate output file. The format of this file is compatible with the *GEOTOOLS ediwrite* utility, which translates the file into the EDI-format required by *GEOTOOLS* and other interpretation programs. Because *GEOTOOLS* only supports standard 5 (SS) and 7 (RR) channel MT-data, this option is restricted to work only with 5 or

7 channel processing. Nevertheless, in conjunction with the `-s` option or the `-L, -l` options (see below) one can write SDM files from multi-channel FC files. The name of the of the SDM file is derived either from the run-file or the line name defined in the LDF, with extension `.sdm`. In addition to the file *TRANMT* outputs a template entry for the *sites* file, a file required by the *ediwrite* utility, at the end of the processing.

- `-L<line-definition-file>`: This option is a variant of the `-s` re-grouping option. *TRANMT* reads the specified LDF and uses the definitions for local input channels and – if provided – reference channels from SITE 1. All remaining channels in the LDF are assumed to be output channels. This requires that the same LDF is used with *DNFF* for creating the FC file which is processed.

- `-l<site\_n>`: This option can only be used together with the `-L` option. `site\_n` specifies a site definition entry in the LDF from the `-L` option. *TRANMT* now only uses the channels specified in this site definition for processing, i. e. a standard 5 or 7 channel MT site. The `-x` option can be used together with the `-L` and `-l` options to output SDM files.

- `-S<SETS>`: This option lets the user specify ranges of subsets out of the total number of sets to be processed. `SETS` is either the name of a file or a sequence of integers which define the number of sub-ranges and the range boundaries of sets to be processed. See the next command line option for how to find out about the set numbers in the FC files. As explained in section 'Output: Fourier-Coefficients' *DNFF* calculates a unique set number for each of the short time-windows in order to allign data recorded simultaneously at different sites. In order to process only a subset of these time-windows *TRANMT* allows to specify subranges. This is useful if one has recorded very long time series and wants to only get a quick overview of the results or look at differences in the TFs from e. g. night-time and day-time recordings from a long-term experiment. The syntax of the `-S` option in the *integer-sequence mode* is:

  tranmt -Sn,i1,j1,i2,j2,....in,jn

wher n is the number of sub-ranges and the subsequent pairs $i1, j1 \ldots in, jn$ are the first and last set-numbers of each sub-range. In the *file-mode* `-S` is used together with a file name:

```
tranmt -Ssets.dat
```

and the file *sets.dat* contains the same information os the integer sequence, her an example:

```
4                    <-- number of sub-ranges
100 150              <-- first, last set-number of first sub-range
300 350              <-- first, last set-number of second sub-range
480 600              <--  ....
1200 1800
```

Using this option produces extra output as in the example below.

```
************************************************

                SET-WINDOWS FOR PROCESSING
SET-WINDOW:  1/1   1/2   2/1   2/2
DEZ-LEVEL
    1      16700 18000 19000 19500
    2       4175  4500  4750  4875
    3       1044  1125  1188  1219
    4        261   282   297   305
    5         66    71    75    77

************************************************
```

- `-p`: This is a helper option for the `-S` option. The set-numbers are derived from the time difference between the time the recording started and a reference time in terms of the sampling rate and the length of the time-windows and the overlap of these. This means that the set numbers in the FC files are not totally obvious. To find out about the start and end set numbers in certain FC files the user can run *TRANMT* with the `-p` option. *TRANMT* prints out the first and last set number for each FC file specified in the *tranmt.cfg* file and quits.

1.3.3 Input Files

Input to *TRANMT* are the Fourier coefficient files created by *DNFF* . The files to process are specified in the *tranmt.cfg* file. For each site the user can specify several FC files, the maximum number of files per site is set by the parameter `ntapemx` (see section 2.2). The maximum number of sites is set by the parameter `nstamx`. Even if running in single site mode the number of sites can differ from one (e. g. when estimating interstation magnetic transfer functions).

1.3.4 Ouput Files

tranmt outputs two files: A "Z–file" containing local TFs and error bars and an optional file containing cross-spectra for all predicted and reference channels in a format compatible with the GEOTOOLS **ediwrite** program (see command line option `-x` and below for details).

Depending on if *TRANMT* is running in single site or remote reference mode the extension of the Z-file is `.zss` or `.zrr`, respectively. Z–files contain TFs (either single station or remote reference) between a pair of local channels and 1 or more predicted channels (usually H_z, and/or E_x and E_y, but the file format supports more general array configurations). All TFs are in the measurement coordinate system. The Z–files also contain channel information (orientations, channel names, etc.) and the full covariance of signal and residuals. With these matrices it is possible to correctly compute error bars in any coordinate system. The multiple station program *MMT* also outputs Z–files of the same format. The Z–files produced for single station, remote reference, and multiple station processing are interchangeable. The same calculations are applied to the contents of any of these files to change coordinates or compute error bars, so this format makes it simple to combine TFs from single site and remote reference processing. The format of the Z–files, and instructions for rotating and computing error bars are given in the separate document **"Errors Bars for Transfer Function Elements in Z–files"**.

The cross-spectra files have the extension **sdm**. This output is optional and invoked using the command line option `-x`. As the format is fully compatible with the EDI-reformatting routine **ediwrite**, this option only supports 5 and 7 channel data, i. e. standard MT and MT plus remote reference setups.

The spectral density matrix is rotated into a geographic north coordinate system, based on the sensor orientations and the declination. These values need to be set correctly before FFTing the time series with *DNFF* . In addition to the output file **tranmt** prints out a line which can be used as a template for the *sites* file necessary to use **ediwrite**. *TRANMT* only checks for the number of channels to be equal 5 or 7, but not for the sequence of channels (GEOTOOLS requires HX, HY, HZ, EX, EY (HXR, HYR)). (subroutines: sdmwrite, ccmat, rotsdm).

1.4 MMT

This program is an extension of the remote reference robust processing and a detailed description of its theory can be found in Egbert (1997). The general idea of this approach is to use as many simultaneous recorded channels as available in order to reduce incoherent noise in single channels and define the (hopefully) quasi-uniform plane-wave source. The output of the program allows a detailed study of the array data with diagnosis for possible coherent noise which might cause not immediately obvious bias-effects leading to misinterpretation. The MT-transfer functions, the final goal of the procedure, are output as Z-files, the exact same format as *TRANMT* . In addition there are other files which allow easy estimates of non-standard TFs.

MMT uses the same files as input as *TRANMT* , i. e. the FC-files created with *DNFF* . Similar to the *tranmt.cfg* file the *array.cfg* file defines the input to *MMT* and the *bs.cfg* file to use for band averaging. *MMT* groups channels together for two different purposes: The first is to estimate levels of incoherent noise by predicting the signal in a fixed group of channels from all remaining groups. The second purpose is the estimation of local transfer functions. In the standard mode the grouping is taken from the *array.cfg* file, i. e. all channels in one FC-file build a group. Considering an array of several 5-channel MT sites every site would make one group and for each of those groups a Z-file with local impedances would be written. But the program is reasonably flexible and can handle a wide variety of field-setups including large EMAP-spreads, only magnetic field recordings and mic xtures of sites with only E- and only H-channels, to name some examples. It is also possible to include non-MT data into the processing e. g. seismic data temperature recordings, etc.

Because the default rules of defining the MT (plane-wave) signal – the strongest signal over all – not always is the right guess, the program allows for additional information provided by the user. E. g. the user might now that one or more sites are unaffected by coherent noise which then can be used to define the MT source. To find out such details it might be useful to first run *MMT* in a standard mode, then closely examine the resulting output and rerun the program with special settings. These setings are passed to the program via extra configuration files and command line options which are described below.

The computational requirements of the program are much bigger than those of *TRANMT* and processing of large arrays (many sites with many channels) might hit the limits of even high-end PCs. In many cases where coherent noise is no problem *TRANMT* might give as good or even better results than *MMT* with less effort. But in cases of severe noise *MMT* is able to reveal the existence of coherent noise and might offer tools to ultimately estimate reasonalbe transfer functions.

1.4.1 The configuration file

In the default mode *MMT* requires only one input file. The default namefor this file is *array.cfg* . This file tells *MMT* 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. Following is an example of a *array.cfg* file for a 2 station array.

```
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
```

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)

For a description of the *bs.cfg* file see section 1.3.1.

1.4.2 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.

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).

- 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 .

- 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.

- 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.

- 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.

Individual station files

- 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**).

Special 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.

1.4.3 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.

1.4.4 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.

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**

1.4.5 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.

–L[LDF_file]

Use the specified LDF file to regroup channels. This is an MT-24 specific feture. *MMT* reads the site-specifications from the LDF file and creates one Z-file for each site (see also the description of the -L option for *TRANMT*).

–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

Arguments which add to or change output files

–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:

$$\mathbf{X}_t = \sum a_{jt}\mathbf{W}_j \tag{1.1}$$

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**.

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 #

1.4.6 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.)

Chapter 2

Programmers Manual

All three programs are written in standard FORTRAN 77. The current versions are compiled using the WATCOM FORTRAN Compiler (Version 10.0). The `igetarg` subroutine, which is used in all three programs is not standard FORTRAN 77 and might have to be changed when porting to another platform/compiler (on SUN the aquivalent routine would be `getarg`).
Because the *ACQ24* data acquisition program interacts directly with the processing programs it is mandantory for the binaries to be in the directory `C:\\proc24\\bin`. Parallel to the `bin` directory is a `src` directory with subroutines for each program containing the source codes of the programs.

- **D** contains source code, *project* and *Makefiles* and include files for *DNFF* .
- **T** contains the code and related files for *TRANMT* ,
- **MMT** is the source codde directory for *MMT*
- **LAPACK** contains two subdirectories `blas_src` and `qr_src` in which are subroutines from the public domain linear algebra library LAPACK. The *project* and *Makefiles* as well as the libraries themselves reside in these suubdirectories. *MMT* uses a number of routines from these libraries.
- **include** is a directory which contains some general include files, `nchmx.inc` being the most important. This file defines the maximum number of channels per site which can be processed by the programs.

2.1 DNFF

DNFF builds the main interface to the MT24 data structure and format and therefore differs most from the standard versions of the program. While tayloring *DNFF* to the MT24 data the code was made easier to maintain and adapt to different field configurations. This is mainly supported by extracting parameters to be changed into include files from the code.
Changes of parameters of this program might be necessary if data shall be processed which differs largely from "standard" data, say extremly long time-series, a very large number of channels in each run and therefore FC-file.

2.1.1 Include files and parameters

There are two files in the source directory **D**: **params1.inc** and **params2.inc**. These files will have to be changed if you want to significantly change windowing or decimation options set in the **decset.cfg** file, but for most users, parameters in these files can be left alone.

params1.inc

These are general parameters used to define the size of arrays to allocate in the main program **dnff.f**.
nwmx

> Maximum length of windows to be FFTd (in samples)

nsmax

> Maximum number of sets to allow for (this is the sum over all sets in all decimation levels). The program stops FFTing once the number of sets exceeds *nsmax* and issues a warning. Because *DNFF* no longer writes a temporary scratch fiel to disk but rather keeps all data in RAM while processing, this number, together with the maximum number of channels can become critical in terms of required memory.

nbadmx

> Maximum number of bad record segments

nbytes

Default storage format for binary time series. Depending on NBYTES the the assumed format of the input binary data file is changed.

For *nbytes* = 4, integer*4 is assumed, while for *nbytes* = 2, integer*2 is assumed.

The user can change the storage format using the command line option -b(2or4)

lpack

logical parameter to control packing of FCs in file. If lpack = .true. each complex FC packed into a 4 byte integer.

params2.inc

These are parameters which control size of arrays used for decimation and related functions. This file is included in **decimate.inc**, which is included in **dcimte.f, decset.f, dnff.f, fcorsu.f, mk_offst.f, mkset.f, pterst.f**. **params2.inc** is also included in **getsp.f**.

ndmx

Maximum number of decimation levels to allow for.

nchmx

Maximum number of channels to be fft'd. This parameter is actually set in another include file *../include/nchmx.inc* which is included in *params2.inc* and also in the other programs of the **EMTF** package.

nfcmx

Maximum number of filter coefficients for decimation filter (should be about equal to the maximum decimation factor)

nfilmax

Maximum number of filters to correct for for a single data channel.

Changes in input data format

If for some reason the format of the input data and related files (the time-series files, the run-files or the line-definition-files) should change, changes would be necessary to *emi_gtsp.for*, *emi_rdhd.for* and *emi_rldf.for*. These files read all necessary parameters for the processing from the data and related files. The variables are defined in *emi_hdr.inc*.

2.2 TRANMT

As for *DNFF* there might be cases where parameters of *TRANMT* have to be change to allow for processing special cases. These parameters are set in the include file **iosize.inc**. This file is in the tranmt source directory. Following is a brief synopsis of parameters which might need to be changed for special purposes. Any parameters in the include files not explicitly discussed here should be left set as they are.

nstamx

> Maximum number of "*stations*" (i.e., channel groups). Note that a here station refers to the set of all channels grouped together in a FC file. There could be several FC files (same group of channels, but a different run) for one station. There could also be several channel groups used to estimate a local impedance tensor, e.g., if **E** and **H** were in separate FC files.

nchmx

> Maximum number of channels for a single channel group.

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 if parameters like *nstamx* and *nchmx* are large.

ncbmx

Maximum number of data points to allow for in wide coherence sort band. If coherence sorting is not used this can be equal to *ntfmax*. If coherence sorting is used this might need to be roughly 3 times as large as *ntfmax*.

nsetmx

Maximum number of sets (i.e., time windows totaled over all decimation levels) to allow for. This should be roughly one third of *ntfmax* (at least for the way I use the program typically).

nbmax

Maximum number of frequency bands to compute estimates for. Increase this if you significantly increase the number of frequency bands in the **bs.cfg** file.

ndmax

Maximum number of decimation levels to allow for. Should be set to whatever is used for **dnff** (compare to *ndmx* in **D/params2.inc**).

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 big enough to be consistent with the number of FCs being save by **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 for **dnff** in **D/params1.inc**. 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.) However, I do not think this is neccessary in most circumstances.

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. (This should never need to be changed for **tranmt**; but it might need to be changed for *MMT* .

The `-L` and `-l` options cause *TRANMT* to read from the specified line-definition-file. If for any case the format of the LDF should change, changes have to be made to the *read_ldf.for* subroutine in the *TRANMT* directory. This is the only place where *TRANMT* could be affected by changes of the MT24 data format and structure.
The output files are created by the routines *wrt_z.for* (Z-files) and *sdmwrite.for* (SDM-files). This is the place to look for changing output format.

2.3 MMT

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.inc**. 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*.

References

Chave, A.D., and D.J. Thomson, Some comments on magnetotelluric response function estimation, *J. Geophys. Res.*, **94**, 1989.

Egbert, G., and J.R. Booker, Robust estimation of geomagnetic transfer functions, *Geophys. J. R. Astron. Soc.*, **87**, 173-194, 1986.

Egbert, G.D. and D.W. Livelybrooks, Single station magnetotelluric impedance estimation: coherence weighting and the regression M-estimate, *Geophysics*, **61**, 964-970, 1996.

Egbert, G. D., Robust multiple station magnetotelluric data processing, *Geophs. J. Int.*, **130**, 475-496, 1997.

Lists of source code files

DNFF

afcor.for	dcimte.for	emi_rdbl.for	ft_subs.for	phs_shft.for
autocor.for	emi_rdhd.for	getsp.for	ptdist.for	badrec.for
decset.for	emi_rldf.for	inpu_bin.for	pterst.for	bdrcsu.for
demean.for	emi_swap.for	ltslv.for	resptbl.for	chdec.for
dnff.for	fcorsu.for	mk_offst.for	sort.for	cldrft.for
dtrnd.for	filtcor.for	cmove.for	emi_gtsp.for	freqout.for
mkset.for	dcfilt.for	emi_open.for	frstdif.for	out_pack.for
decimate.inc	iounits.inc	params2.inc	emi_hdr.inc	params1.inc
sys_par.inc				

TRANMT

bset.for	matmult.for	mult.for	rfhead.for	sindex.for
chdec.for	mcohc.for	pwrvf.for	rotsdm.for	sort.for
cmove.for	mkfdir.for	rbstrn.for	rtreref.for	stack.for
cohband.for	mkhat.for	rbstsdm.for	rxspclev.for	tranmt.for
cohsrt.for	mkrec.for	rdcndwt.for	savtf.for	trlsrr.for
corchng.for	mkrhat.for	read_ldf.for	sdmsort.for	unstack.for
edfwts.for	modobs.for	readfvg.for	sdmwrite.for	wrt_z.for
fop.for	movec.for	reorder.for	setup.for	wt.for
iosize.inc				

MMT

anfld.for	grad.for	multmtrn.for	read_ldf.for	sindex.for
canonic.for	ln_rbst.for	n_rbst.for	readfvg.for	timerr.for
cn_wt.for	minpwr.for	pc_coeff.for	refproj.for	var_adj.for
corchng.for	mk_fd_rec.for pc_out.for	rr_rbst.for	wrt_z.for	
extra.for	mk_list.for	prteig.for	rsp.for	wrtx.for
filtpc.for	mmt_mtrx.for	rbstk2.for	sep_s_n.for	geogcor.for
mmt_subs.for	rbstreg.for	setup.for	iosize.inc	nstamx.inc