

1 **MT - How to ...**
2 **—**
3 **Help for MT data processing and analysing**

4 Kent Inverarity, Jared Peacock, Lars Krieger*

5 2012

*University of Adelaide – School of Earth and Environmental Sciences – Mawson Building – Adelaide

¹ Introduction

² This is a collection of guides (*How to ...*) for various steps in the processing of MT data. Most
³ steps are based on the MTPy package.

⁴ January 25, 2013, LK

Contents

1	1	Structure	4
2	1.1	4
3	2	In the field	5
4	3	Raw-data-/time-series processing	6
5	3.1	Processing – step by step	6
6	3.1.1	Concatenate time series recorded by EDLs - dayfiles	7
7	3.1.2	Concatenate time series recorded by EDLs - arbitrary length	7
8	3.1.3	Time series data calibration	8
9	3.2	E field logger data	8
10	4	Modelling (1-/2-/3-D)	9
11	5	Visualisation	10
12	6	Helpful information, tools, and scripts	11

¹ 1 Structure

² 1.1

DRAFT

¹ 2 In the field

DRAFT

3 Raw-data-/time-series processing

3.1 Processing – step by step

The processing starts with the raw time series data files from the data loggers. Additionally, a configuration-file is needed. This contains all sort of meta data and further information. The configuration file is an ascii text file of the form

```
[station_name_1]
keyword1 = value1
keyword2 = value2
.
.
[station_name_2]
.
.
```

Few basic keywords are required:

- latitude (deg)
- longitude (deg)
- elevation (in meters)
- sampling_interval (in seconds)
- station_type (MT, E, B)

Depending on the type of station the following entries are required.

E-field recorded:

- E_logger_type (edl/ellogger)
- E_logger_gain (factor/gain level)
- E_instrument_type (electrodes)
- E_instrument_amplification (applied amplification factor)
- E_Xaxis_azimuth (degrees)
- E_Xaxis_length (in meter)
- E_Yaxis_azimuth (degrees)
- E_Yaxis_length (in meter)

B-field recorded:

- B_logger_type (edl)
- B_logger_gain (factor/gain level)
- B_instrument_type (coil, fluxgate)
- B_instrument_amplification (applied amplification factor)

Further information can be provided using more keywords. The content of the configuration file will be stored as meta data in the final EDI file (if not restricted by the EDI format).

3.1.1 Concatenate time series recorded by EDLs - dayfiles

Functions for concatenation are in the *mtpy.utils.filehandling* module. The reference to the module is given here as *FH*. It is assumed that the files are ascii-formatted, and are named according to the EDL standard. This is

`<stationname><year><month><day><hour><minute><seconds>.<channel>;`

and the channels in question are *ex,ey,bx,by,bz*, case insensitive.

The data in the files are assumed to be either in single-column form (instrument counts), or in two-column form: *single column time stamp (e.g. epochs or datetime-string) - instrument counts*

1. Find the sampling rate

You have to know the duration of your single files (e.g. 10 mins, 1 hour,...). Chose a file, of which you know that it contains complete information for the full duration. Then obtain the sampling period with the function

`FH.get_sampling_interval_fromdatafile(filename, duration in seconds)`

2. Put all files of interest into one folder (preferably sorted by station).

3. For generating dayfiles call

`FH.EDL_make_dayfiles(foldername, sampling , stationname = None)`

This generates a subfolder called *dayfiles*. If the given files cannot be merged continuously, several files are created for the same day.

4. Output dayfiles have a single header line, which starts with the character `#`. The content of the line is

stationname ; channel ; sampling interval ; time of first sample ; time of last sample.

Time stamps are given in epoch seconds.

3.1.2 Concatenate time series recorded by EDLs - arbitrary length

..

Decimate time series

Generally speaking, MT deals with the spectral analysis of time series. Therefore it is beneficial to process these time series in their full length. Unfortunately, computer hardware limitations (memory) sometimes do not allow to just read in a complete set of time series. One possibility to overcome this issue is a pre-processing step on the time series: "decimation". The data are downsampled by a given factor, so thhe number of samples is samll enough to be fully processed at once.

Although the name suggests it, the actual decimation factor does not have to be "10". The factor is to be chosen in a way that the loss of information is minimised. If for instance the sampling rate is 100Hz, but the instrument sensitivity is only sufficient in a range < 20Hz, a decimation with a factor of 5 effectively causes no harm to the data.

1 Be careful!!

2 It may seem that a decimation by a factor of n just means to take every n -th element of a time series. But unfortunately, this can result in spurious phase information. In order to avoid any side effects, a more sophisticated algorithm has to be used.

3 The functions dealing with the decimation of time series can be found in `mtpy`.

6 3.1.3 Time series data calibration

7 The conversion of time series from lists of raw *instrument-output/-counts* into time series of data values with an actual physical meaning and appropriate units is called *calibration*. The functions for the calibration are contained in the module `mtpy.processing.calibration` (here `Cal` in short).

11 The calibration depends on the instrument as well as on the respective data logger. The given time series are multiplied by various factors, which are unique for each kind of instrument, logger, and their software setup. Additionally, possible spurious offsets can be removed.

14 The basic unit for the magnetic flux density is Tesla ($[\mathbf{B}] = T$), the unit of the electric field is Volt per meter ($[\mathbf{E}] = \frac{V}{m}$). For obtaining more convenient numbers, these units are often scaled e.g. to nT or $\frac{\mu V}{m}$.

17 Using the EDL or "elogger" datalogger, the time series values are given in microvolts μV . The amplification factors of the instruments as well as the gain factor of the data logger are given in the configuration files. The basic function `Cal.calibrate()` carries out a calibration for one given data array. In order to calibrate a data file, use `Cal.calibrate_file()` instead. For calibrating several time series at once, follow these steps:

- 22 1. Put all raw time series files into one folder.
- 23 2. Setup a proper configuration file, containing the necessary information on all included stations.
- 25 3. Call the wrapper script
26 `calibratefiles.py <directory name> <configuration file> [optional: <output`
27 `directory>]`

28 The result is a collection of calibrated data files located in the output directory. If no explicit output directory is given, a new subdirectory "calibrated" is generated within the input directory.

31 The units are microvolts per meter ($\mu V/m$) for the electric field and nanoteslas (nT) for the magnetic field.

33 The calibrated files have a single header line, starting with the common commenting symbol "#". The header line contains the fields

35 `stationname, channel, unit of data, sampling interval, timestamp first sample ,`
36 `timestamp last sample, latitude, longitude, elevation`

37 Hence from this point on all basic information are present, which are necessary for continuing the MT processing, even if the configuration file should be lost.

39 In order to be able to distinguish between latitude and longitude values, the standard position format is applied: latitude values are given as 2-digit numbers $\pm dd.ddddd$, whereas longitudes are in the 3-digit form $\pm ddd.ddddd$. The positions are given to 5 decimal digits, this means an accuracy of $< 2m$.

¹ 3.2 E field logger data

DRAFT

4 Modelling (1-/2-/3-D)

DRAFT

5 Visualisation

DRAFT

6 Helpful information, tools, and scripts

DRAFT