Razorback: Open Source Python Library for Robust Processing of Magnetotelluric Data

P. WAWRZYNIAK, F. SMAI (BRGM)

INSTITUT CARNOT brgm

Géosciences pour une Terre durable
brgm

THE RAZORBACK LIBRARY

Smaï, F., & Wawrzyniak, P. (2020). Razorback, an open source Python library for robust processing of magnetotelluric data. Frontiers in Earth Science, 8, 296.

frontiers in Earth Science

TECHNOLOGY AND CODE
published: 02 September 2020
doi: 10.3389/feart.2020.00296

Razorback, an Open Source Python Library for Robust Processing of Magnetotelluric Data

*Farid Smaï and Pierre Wawrzyniak**

BRGM, French Geological Survey, Orléans, France

Magnetotellurics (MT) is a geophysical method that investigates the relationships among the different components of the natural electromagnetic field related to the geoelectric structure of the subsurface. Data can be contaminated by anthropic noise sources and suffer from transient noise to signal variations. Since the 80s, robust processing methods have been introduced to minimize the impact of noise on sounding quality. This paper presents Razorback, an open source Python library, implemented to handle, manipulate, and combine time series of synchronous data. This modular library allows users to plug in data prefilters and includes both M-estimator and bounded influence techniques, as well as a two-stage multiple remote reference. Validation of this library is performed on a real data set by comparing the results with those of an existing code. In contrast to standalone codes, the developed library allows for the design of complex and specific processing procedures. As examples, Razorback is used to perform (i) continuous time lapse processing and (ii) processing of one site in a peri-urban context. In the latter case, we have tested all possible combinations of remote reference stations in an MT array. Our phase tensor analysis shows that the bounded influence outperforms the M-estimator in reducing the impacts of man-made electromagnetic noise on magnetotelluric soundings. The Razorback library is available at https://github.com/BRGM/razorback. Jupyter notebooks for data handling and MT robust processing are available at https://github.com/BRGM/razorback/blob/doc/docs/source/tutorials/.

Keywords: magnetotellurics, time-series analysis, Fourier analysis, robust methods, Python, M-estimator, bounded influence, remote reference

OPEN ACCESS

Edited by:
Fiak Donner,
Hochschule Magdeburg-Stendal, Germany

Reviewed by:
Leonardo Guimarães Miqueletti, Universidade Federal Fluminense, Brazil
Pavel Pushkarev, Lomonosov Moscow State University, Russia
Jared Peacock, United States Geological Survey (USGS), United States

****Correspondence:***
Pierre Wawrzyniak
p.wawrzyniak@brgm.fr

Specialty section:
This article was submitted to Solid Earth Geophysics, a section of the journal Frontiers in Earth Science

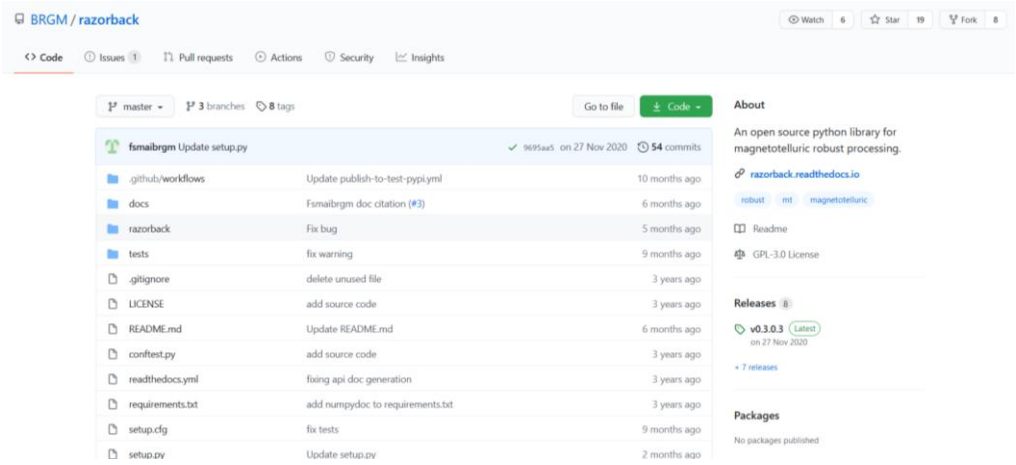
Received: *05 February 2020*
Accepted: *25 June 2020*
Published: *02 September 2020*

Citation:
Smaï F and Wawrzyniak P (2020) Razorback, an Open Source Python Library for Robust Processing of Magnetotelluric Data. Front. Earth Sci. 8:296. doi: 10.3389/feart.2020.00296

1. INTRODUCTION

The magnetotelluric (MT) method studies the relationships in the frequency domain among components of the natural electromagnetic (EM) field (Vozoff, 1972). MT fields are generated (i) by external geomagnetic sources (ionospheric currents) at frequencies under 1 Hz and (ii) by atmospheric lightnings propagating through the earth-ionosphere waveguide at frequencies above 1 Hz. Recorded at the ground surface, MT fields are supposed to be plane waves. As stated by Ward (1967), noise in EM fields can be either instrumental noise, "geological" noise or disturbance field EM noise. The latter is caused by fluctuations of the natural sources (mainly related to solar activity above 1 Hz) and artificial/man-made sources. In urban and industrialized areas, man-made EM sources contaminate MT fields and cause divergence from the plane wave model.

Github page
github.com/brgm/razorback



Géosciences pour une Terre durable
brgm

A PYTHON LIBRARY FOR MT PROCESSING ?

Why a Python library ?

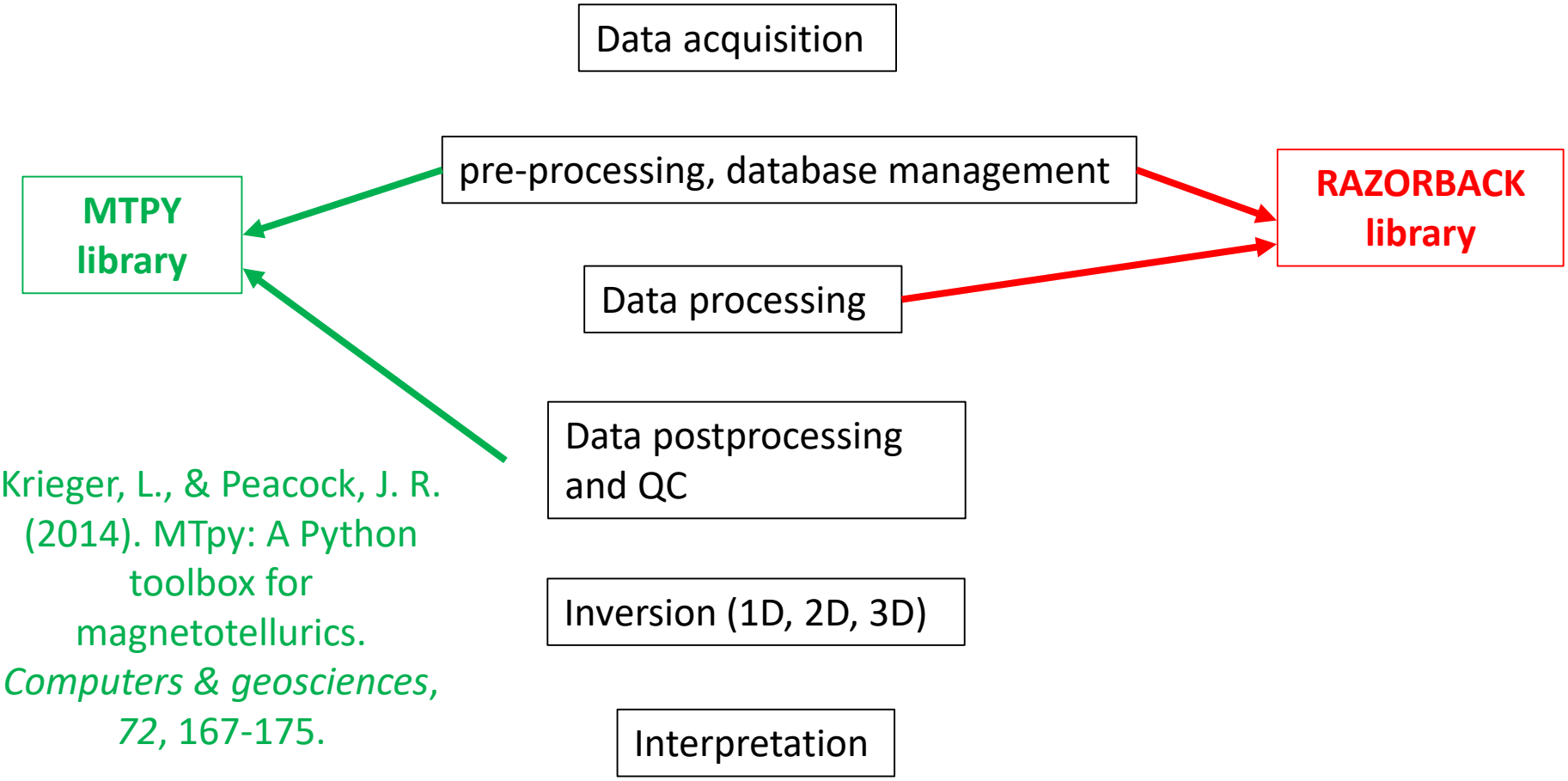
- Python is a free programming language, easy to learn, popular in science application.
- A library is a building block for complex applications, a toolbox for various situations.
  - Not a one-way solution imposed to all problems
  - Give users the keys to customize their tools
  - Connect to the rich Python science ecosystem

Why Open Source ?

- We hope an open source software will :
  - Improve code base quality thanks to more user feedbacks
  - Open to new features and application thanks to user contributions
  - Encourage bridges toward already-in-use tools in MT community

MT WORKFLOW

Non exhaustive



Krieger, L., & Peacock, J. R. (2014). MTpy: A Python toolbox for magnetotellurics. *Computers & geosciences*, *72*, 167-175.

RAZORBACK AIMS TO

- Help the MT user in managing huge datasets, MT arrays
  - Data exploration tools
  - Find synchronous files
  - Extract common parts
- Allow MT user to perform Transfer Function (TF) estimates in different configurations
  - single site processing
  - two stage remote reference processing (with multiple remote references)
- Perform different TF estimation methods
  - Ordinary least square (OLS)
  - M-estimator (ME)
  - Bounded Influence (BI)
- Allow Time Lapse processing

WHAT IT IS NOT

Razorback will not

- Perform data QC (looking at the time series, spectra, synchronization)
- Accomplish any miracles (corrupted data → biased TF estimates)
- Perform TF QC analysis, distortion analysis → MTPY
- Perform inversion

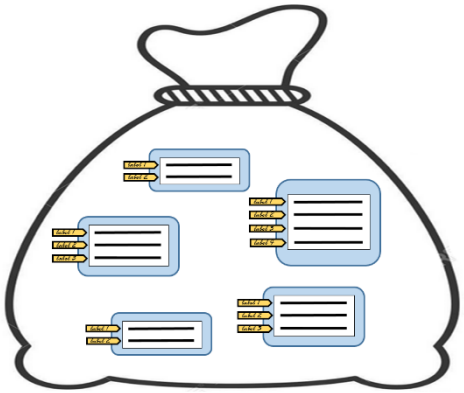
Razorback does not aims to

- Compete with other codes in terms of performance
- Substitute to other preexisting well known and recognized codes (RRMT, BIRRP,….)

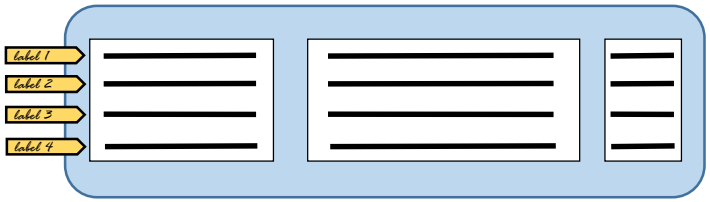
HANDLING TIME SERiES WITH THE RAZORBACK LIBRARY

Goals: Store, Explore and Manage a full MT survey

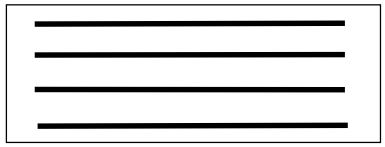
ELEMENTARY BRICKS TO HANDLE TIME SERIES



Inventory



SignalSet



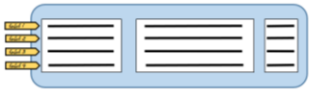
SyncSignal

- The MT user wants to:
  - store
  - manage
  - explore

  a full survey database
- We designed 3 hierarchical levels of structures
  - Inventory
    - ✓ takes care of the full MT survey
  - SignalSet
    - ✓ gathers several runs of a set of channels
    - ✓ to be passed to processing routines
  - SyncSignal
    - ✓ low-level time series structure
    - ✓ includes metadata of the time series

ELEMENTARY BRICKS TO HANDLE TIME SERIES

The SyncSignal object



- The most elementary time data structure that we consider are sampled signals
  - the sequence of values taken by the signal
  - the sampling rate
  - Metadata: the starting time, the calibration function (raw signal ≠ signal of interest), electrode dipole lengths, orientation

Synchronousness definition

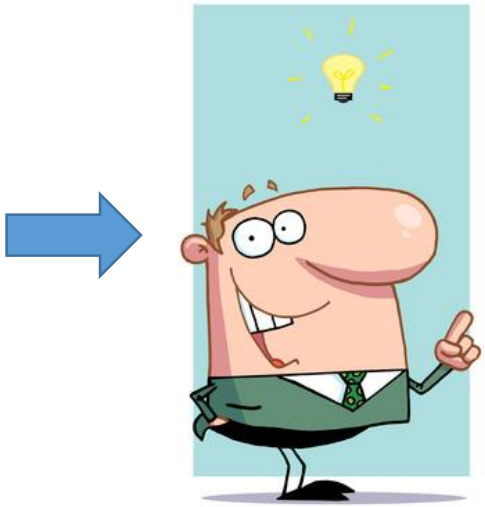
- Two signals are synchronous when they have :
  - the same sampling rate
  - the same starting time
  - the same length

Synchronous signals and associated metadata can be stored in a SyncSignal Object

ELEMENTARY BRICKS TO HANDLE TIME SERIES

SyncSignal restrictions

- SyncSignal are very simple structures
- The SyncSignal objects alone cannot gather the time series needed for a TF estimation :
  - Acquisition may be discontinuous (including time intervals without any data)
  - Consecutive acquisitions may have different sampling rates
- tracking the same channel across several runs is not easy



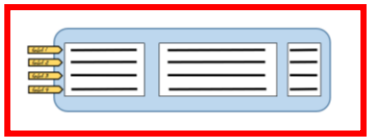
SignalSet

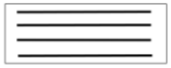
Géosciences pour une Terre durable
brgm

ELEMENTARY BRICKS TO HANDLE TIME SERIES

The SignalSet object

- Let's assume a MT station array :
  - Many sites → starting and ending times of acquisition differs
  - Acquisition may be discontinuous (including time intervals without any data)
  - consecutive runs of acquisition at different sampling rates.



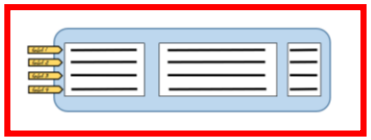


Data can be gathered in a SignalSet object, which reunites acquisition runs of the same channels.

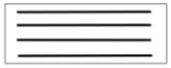
- What the **SignalSet** object do ?
  - It groups several channels and runs in one place
  - It adds labels on channels
  - It guarantees that :
    - ✓ runs don't overlap with each other in time domain
    - ✓ each run contains exactly all the same channels

- **SignalSet** object can be submitted to TF estimation procedures

SignalSet Object labeling and combination skills



- The labeling system :
  - is a meaningful way to retrieve one channel, or a group of channels, from a SignalSet object.
  - associates some character strings to groups of one or more indices → use names rather than indices.



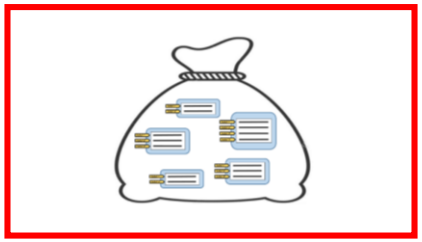
- Combination skills :
  - group or split existing SignalSet objects to produce new ones. (no duplication the data values, which allows building any combination without memory cost)
  - **Joining**: different acquisition runs of the same channels or merging distinct channels.
  - **Splitting**: selecting some channels and runs or narrowing the time range.

Restrictions

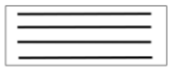
A SignalSet is good at gathering the data for a TF estimate procedure, but it cannot contain a full MT survey.

ELEMENTARY BRICKS TO HANDLE TIME SERIES

The Inventory



- An Inventory object is a container that gathers several SignalSet objects without any constraints.
- The MT user can explore and filter the dataset
  - User can build a new Inventory containing less data
    - ✓ by selecting some channels and runs
    - ✓ or by narrowing the time range.
  - User can produce a SignalSet that contains as many signals as possible → ***pack()*** function.
- In practice:
  - We start by constructing an Inventory object that gathers all the data acquired for the survey.
  - Then, for each site, we extract a sub-Inventory, produce the adapted SignalSet and estimate the TF.

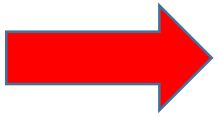


Géosciences pour une Terre durable
brgm

Memory optimization

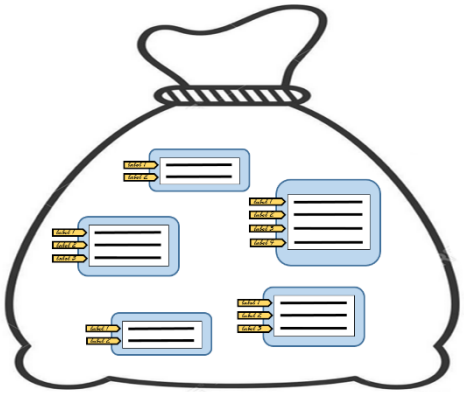
- Entire MT survey = huge data volume

- Razorback uses **Dask library** (Dask Development Team, 2016).
  - By storing the time data in a dask.array object rather than a classical (**Numpy**) array, the memory cost of the Inventory of an entire survey becomes negligible.
  - The time data files are only loaded when needed during the computation of the Fourier coefficients, and then they are unloaded.



Using the dask.array, the memory footprint of the processing no longer depends on the size of the survey

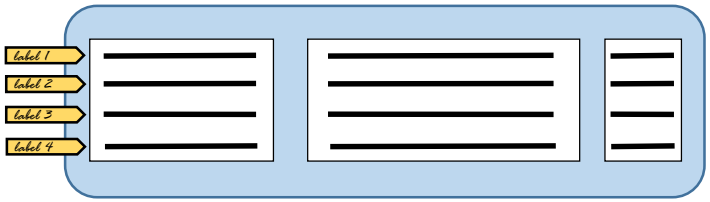
ELEMENTARY BRICKS TO HANDLE TIME SERIES: A SUMMARY



Inventory

- Store a full survey
- *filter/select* → reduced inventory
- *pack* → maximal (longest) SignalSet

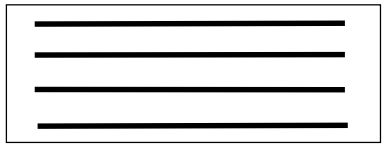
contains and recombines



SignalSet

- Gather time series for processing
- Labels on channels
- Channels × runs

organizes and labels



SyncSignal

- Store basic time series and metadata
- Internal use

Signal Set Tutorial

- https://github.com/BRGM/razorback/blob/doc/docs/source/tutorials/
- Signalset.ipynb

COMPUTING TRANSFER FUNCTIONS WITH THE RAZORBACK LIBRARY

Goals: operate easy single site and two stage remote reference robust processing

Framework for robust methods

- Robust methods are based on consecutive weighted least squares problem on the discrete Fourier coefficients.

- The weighting function has a strong influence on weighted least square
  - Stability : estimate is independent of the initialization
  - Robustness : estimate is not biased by abnormal data
  - BUT one weighting function alone cannot give both stability and robustness
  - → robust methods combine weighting functions to get stability and robustness

- Robust methods use sequences of weighting function in consecutive least square
  - Starting with the most **stable** weighting function to ensure global stability
  - Ending with the most **robust** weighting functions to eliminate abnormal data

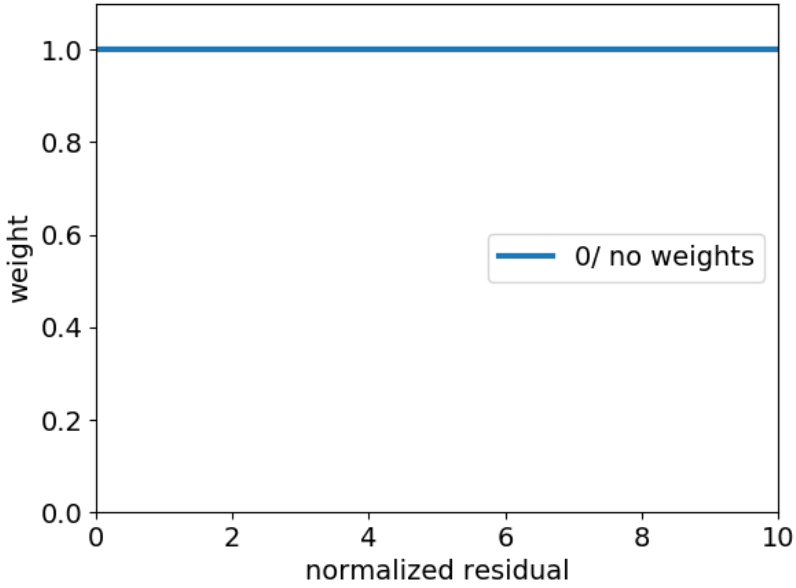
Transfer Function estimation with razorback

Ordinary Least Square

- The simplest weighting function
  - weight = 1
  - Also the more stable
  - But not robust at all

- Correspond to the ordinary least square

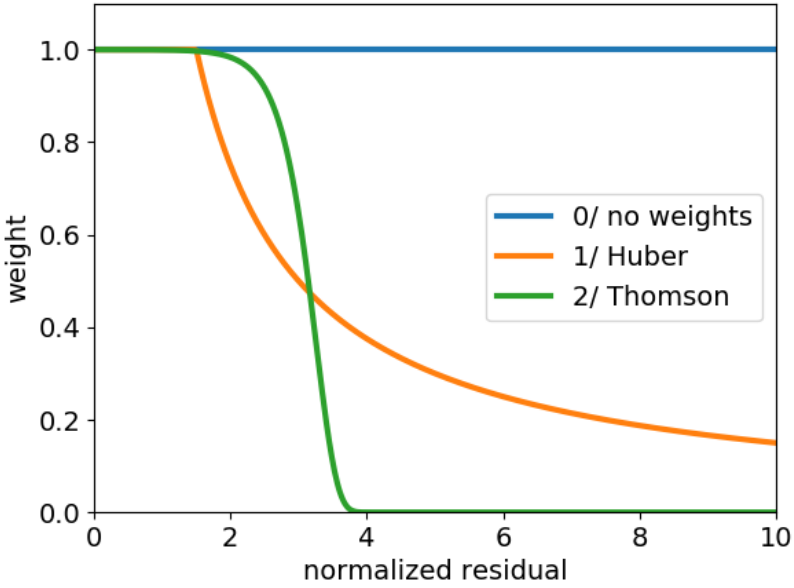
- Will be used as first weighting function for all other methods



Transfer Function estimation with razorback

M-Estimator

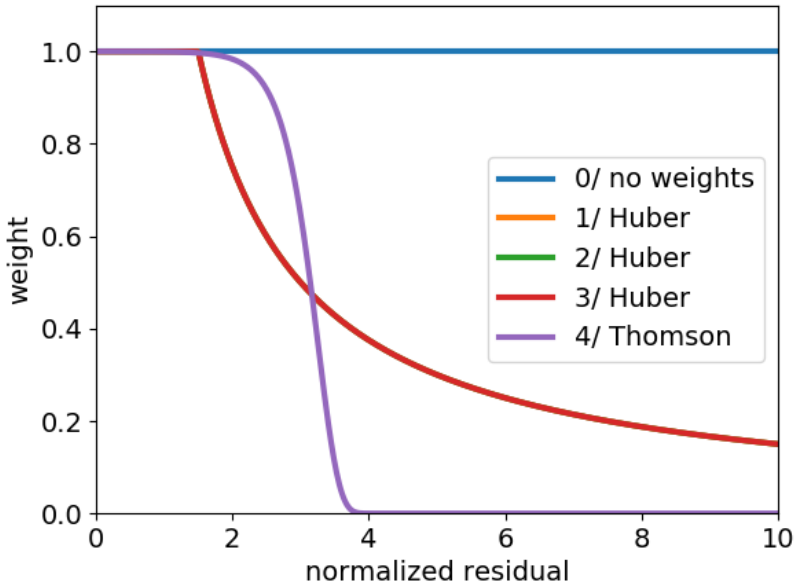
- Egbert and Booker, 1986
- Combine 2 weighting functions
  - (weight=1 for start)
  - Huber function
  - Thomson function
- Huber function
  - avoids the high sensitivity to outlier of (weight=1)
  - still ensures stability
- Thomson function
  - More robust
  - Not stable

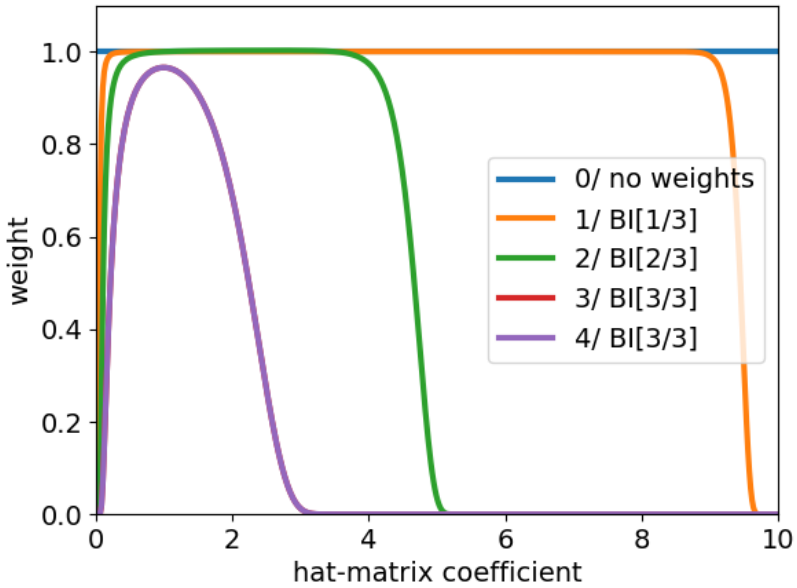


Transfer Function estimation with razorback

Bounded Influence

- Chave and Thomson (2003)
- The size of weight sequence is controllable
- Weighting functions not only rely on residual but also on hat-matrix coefficients
  - w = w_res * w_hat
- Hat-matrix coefficients of regular data follow a beta distribution
  - Coefficients deviating from that are probably outliers





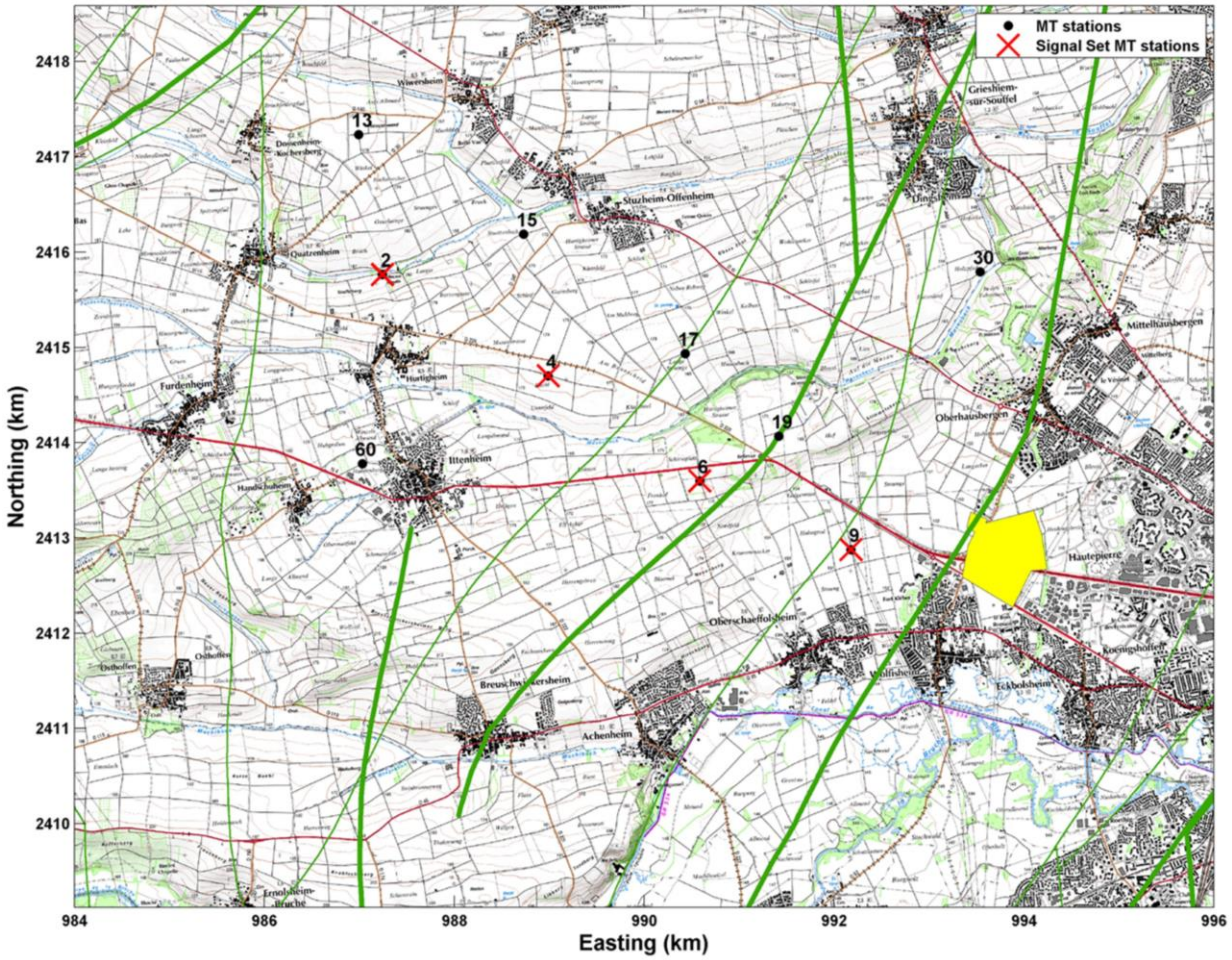
Transfer Function estimation with razorback

3 helpers functions

- **`signal.fourier_coefficients(freq, Nper, overlap, slepian_window(4))`**
  - Compute Fourier coefficients from a SyncSignal/SignalSet object
  - Low-level function
- **`transfer_function(output_fourier, input_fourier, weigths, …)`**
  - Compute TF estimate
  - Perform robust estimation according to **`weights`**
    - ✓ Ordinary least square, M-estimator, Bounded Influence, …
  - Low-level function

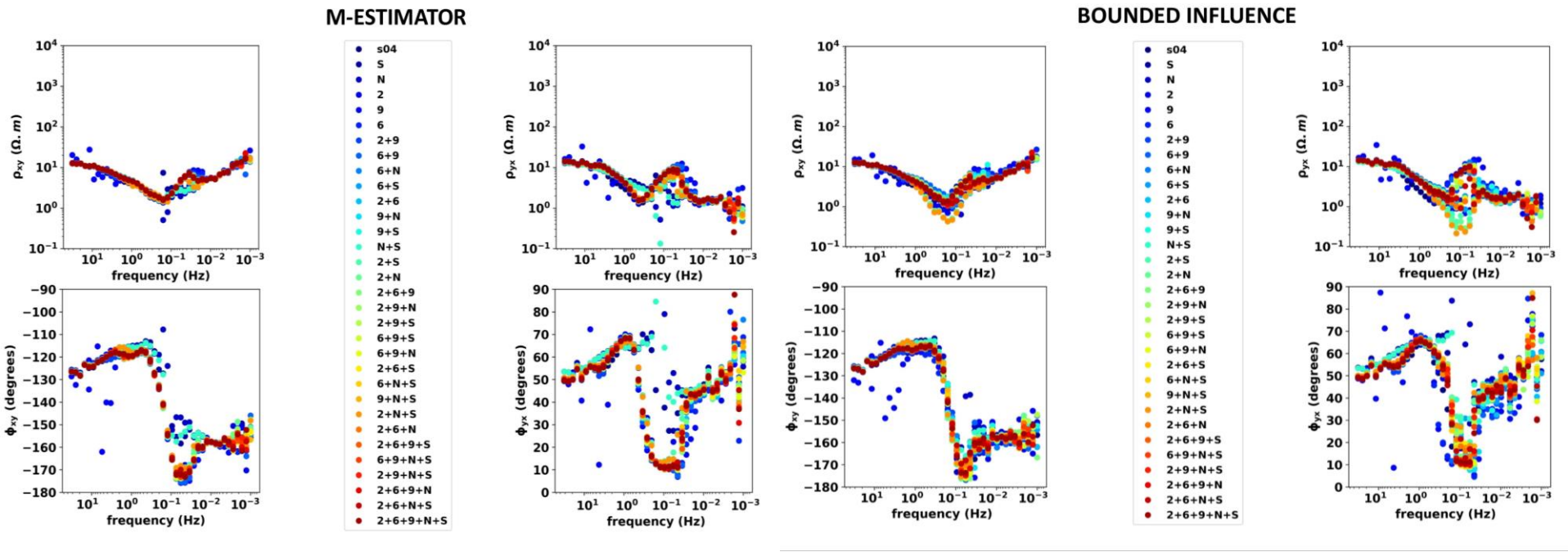
- **`impedance(signalset, list_freq, weights, …)`**
  - Impedance estimation from a labelled SignalSet object (local+remote)
    - ✓ Compute the Fourier coefficients
    - ✓ Perform robust estimation according to **`weights`**
    - ✓ Include 2-stage multi-remote method
  - Handle multiple runs at once, with different sampling rates
  - High-level function

Robust Processing Tutorial : Test Dataset



- Periurban context, western part of Strasbourg city (eastern part of france, german border)
- 4 synchronous MT stations (2, 4, 6, 9)
- 2 distant remote stations (30km North and South)
- Metronix stations only

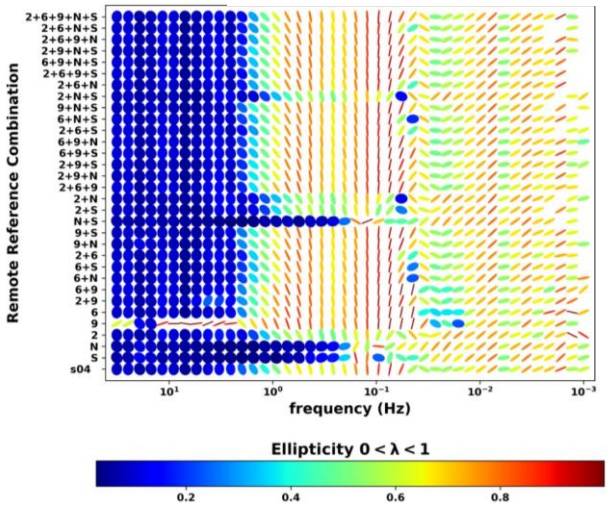
Testing all combinations of synchronous stations as remote references



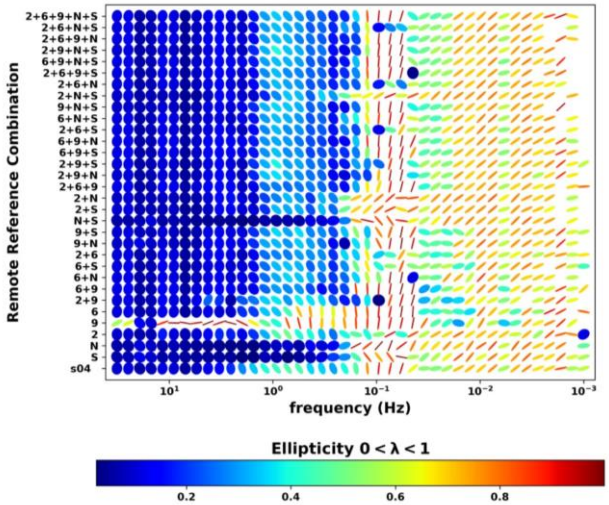
→Comparing M-Estimator and Bounded Influence for any combination of RR.

Testing all combinations of synchronous stations : phase tensor

M-ESTIMATOR



BOUNDED INFLUENCE



→Comparing M-Estimator and Bounded Influence for any combination of RR.

RAZORBACK PERSPECTIVES

THE LIBRARY EXISTS AND CAN BE USED

But there are many perspectives of developpment

We want to invite MT users to contribute and initiate team work on MT robust processing procedures.

Progress can be made

- Implementation of new data prefilter: DFT tables can be weighted before entering the processing routine.
- New or different procedures can be implemented
  - Implying new weighting functions
- Time-Lapse processing can be pushed.
- Inter-station TF can be performed

Testers are very much welcome!

→ ***Smaï, F., & Wawrzyniak, P. (2020). Razorback, an open source Python library for robust processing of magnetotelluric data. Frontiers in Earth Science, 8, 296.***

→ **github.com/brgm/razorback**