

MTPy for magnetotelluric data analysis and visualisation

Alison Kirkby

on behalf of:

Jared Peacock, Fei Zhang, Rakib Hassan, Brenainn Moushall, Yingzhi Gou, Lars Krieger, Geoscience Australia MT team (Jingming Duan, Wenping Jiang, Darren Kyi, Adrian Hitchman), many other scientists, developers, and bug-finders, past and present...

Outline

Background

Overview of the modules

Installing MTPy

Using MTPy – practical demo

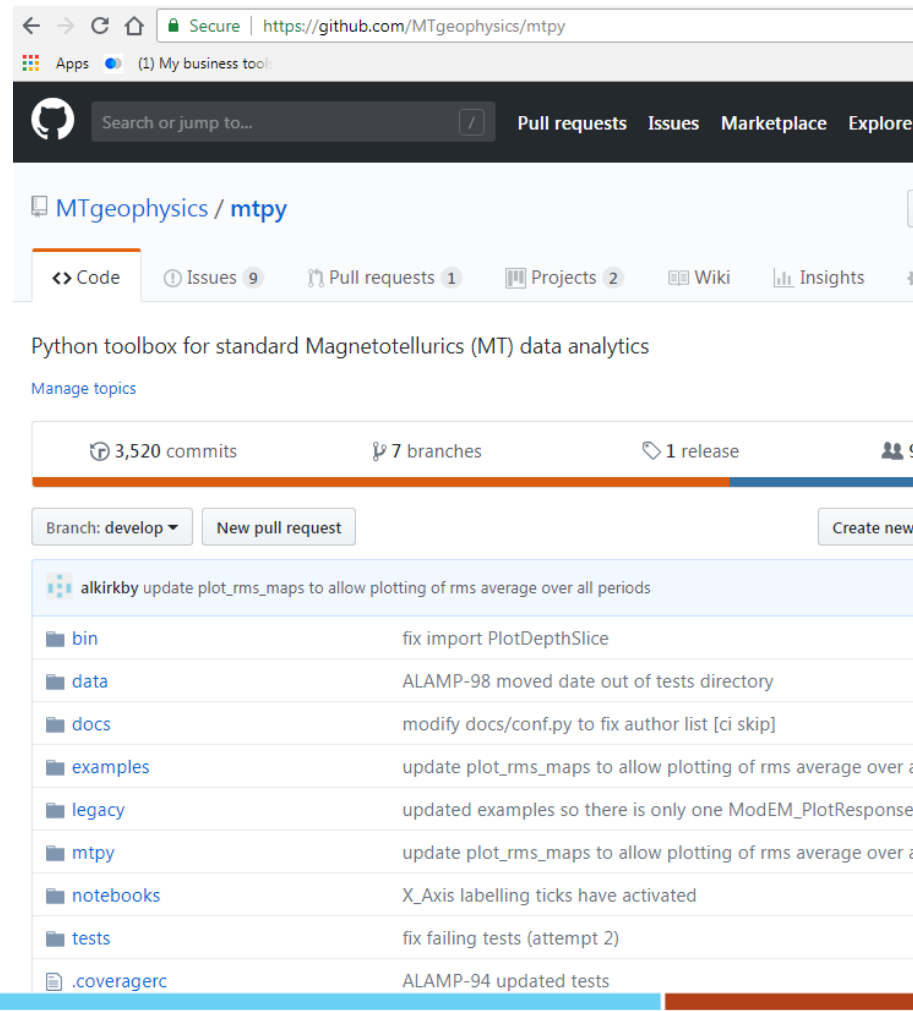
Background on MTPy

Initiated in 2013 at the University of Adelaide

- Jared Peacock, Lars Krieger, myself, others....

2016 - Geoscience Australia commenced development

- Clean up
- Testing suite & documentation
- Functionality development
- Apply software engineering “best practice”



Secure | https://github.com/MTgeophysics/mtpy

Apps (1) My business tool

Search or jump to... Pull requests Issues Marketplace Explore

MTgeophysics / mtpy

<> Code 9 Issues 1 Pull requests 2 Projects 1 Wiki Insights

Python toolbox for standard Magnetotellurics (MT) data analytics

Manage topics

3,520 commits 7 branches 1 release

Branch: develop New pull request Create new

alkirkby	update plot_rms_maps to allow plotting of rms average over all periods
bin	fix import PlotDepthSlice
data	ALAMP-98 moved date out of tests directory
docs	modify docs/conf.py to fix author list [ci skip]
examples	update plot_rms_maps to allow plotting of rms average over a
legacy	updated examples so there is only one ModEM_PlotResponse
mtpy	update plot_rms_maps to allow plotting of rms average over a
notebooks	X_Axis labelling ticks have activated
tests	fix failing tests (attempt 2)
.coveragerc	ALAMP-94 updated tests

Background on MTPy

2017 - Moved to a new repository

<https://github.com/MTgeophysics/mtpy>

- Merged 2 versions

GA continues maintenance/development in collaboration with Jared Peacock at USGS

Look out for MTPy v. 2.0

← → ↻ 🏠 Secure | <https://github.com/MTgeophysics/mtpy>

Apps (1) My business tool

Search or jump to... / Pull requests Issues Marketplace Explore

MTgeophysics / mtpy

<> Code 9 Issues 1 Pull requests 2 Projects Wiki Insights

Python toolbox for standard Magnetotellurics (MT) data analytics

Manage topics

3,520 commits 7 branches 1 release 9

Branch: develop New pull request Create new

alkirkby update plot_rms_maps to allow plotting of rms average over all periods

bin	fix import PlotDepthSlice
data	ALAMP-98 moved date out of tests directory
docs	modify docs/conf.py to fix author list [ci skip]
examples	update plot_rms_maps to allow plotting of rms average over a
legacy	updated examples so there is only one ModEM_PlotResponse
mtpy	update plot_rms_maps to allow plotting of rms average over a
notebooks	X_Axis labelling ticks have activated
tests	fix failing tests (attempt 2)
.coveragerc	ALAMP-94 updated tests

MTPy access and usage

General Public License (3.0) – free, open source

However, as with most open source software:

- We provide no warranty
- If you modify the code, please link to the original

If you use MTPy we would appreciate it if you acknowledge the use of the software and provide a link to the repository, you can cite:

Krieger, L. & Peacock, J. (2014). A Python Toolbox for Magnetotellurics. *Computers and Geosciences*, v.72, p167-175

Kirkby, A., Zhang, F., Peacock, J., Hassan, R., & Duan, J. (2019). The MTPy software package for magnetotelluric data analysis and visualisation. *Journal of Open Source Software*, 4(37), 1358-1358. doi:10.21105/joss.01358

Please let us know of any bugs via the GitHub 'Issues' page

MTPy is....

A collection of tools to (hopefully) help you carry out processing, analysis, modelling, and visualisation of MT data

A work in progress

Largely script based

MTPy is not...

Perfect

Complete

Bug-free

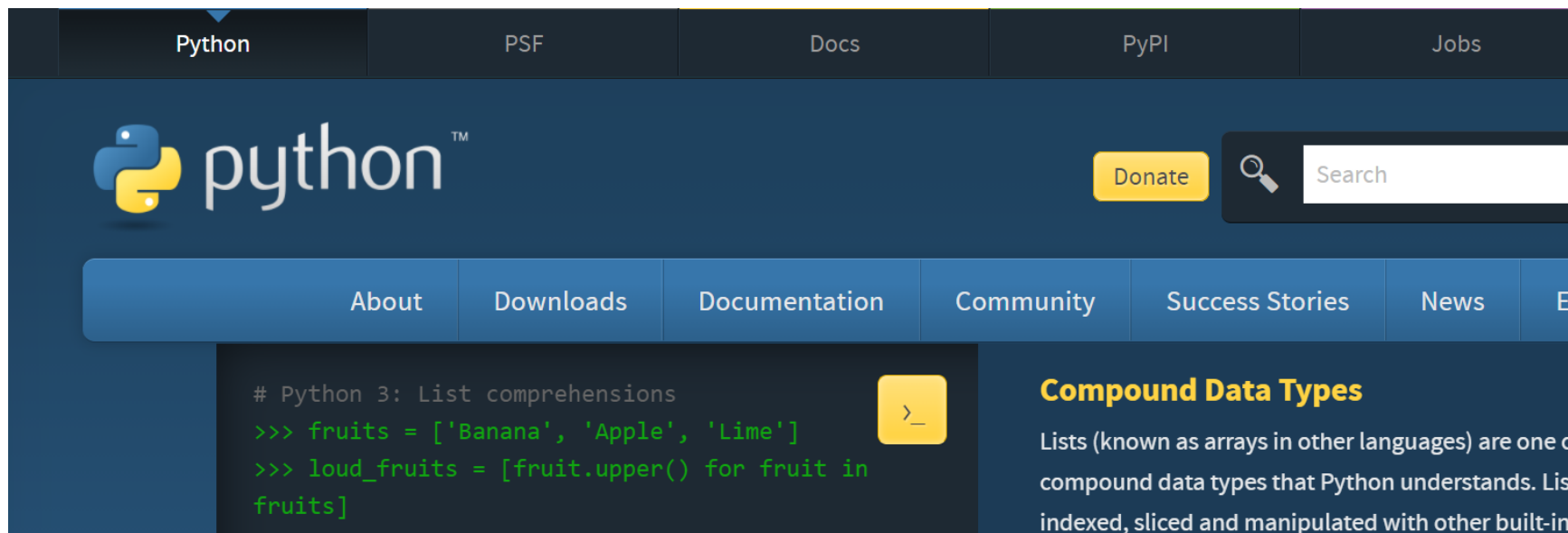
A black box

(i.e. we strongly advise to check your outputs and understand what you are doing)

Python

Useful to have some background knowledge of Python to get the most out of MTPy

Lots of (free and paid) online resources and courses



The image shows a screenshot of the Python.org website. At the top, there is a dark blue navigation bar with links for Python, PSF, Docs, PyPI, and Jobs. Below this is a large blue banner featuring the Python logo and the word "python" in white. To the right of the logo is a yellow "Donate" button and a search bar with a magnifying glass icon and the text "Search". Below the banner is a secondary navigation bar with links for About, Downloads, Documentation, Community, Success Stories, News, and E. The main content area is split into two columns. The left column contains a code snippet for Python 3 list comprehensions:

```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
```

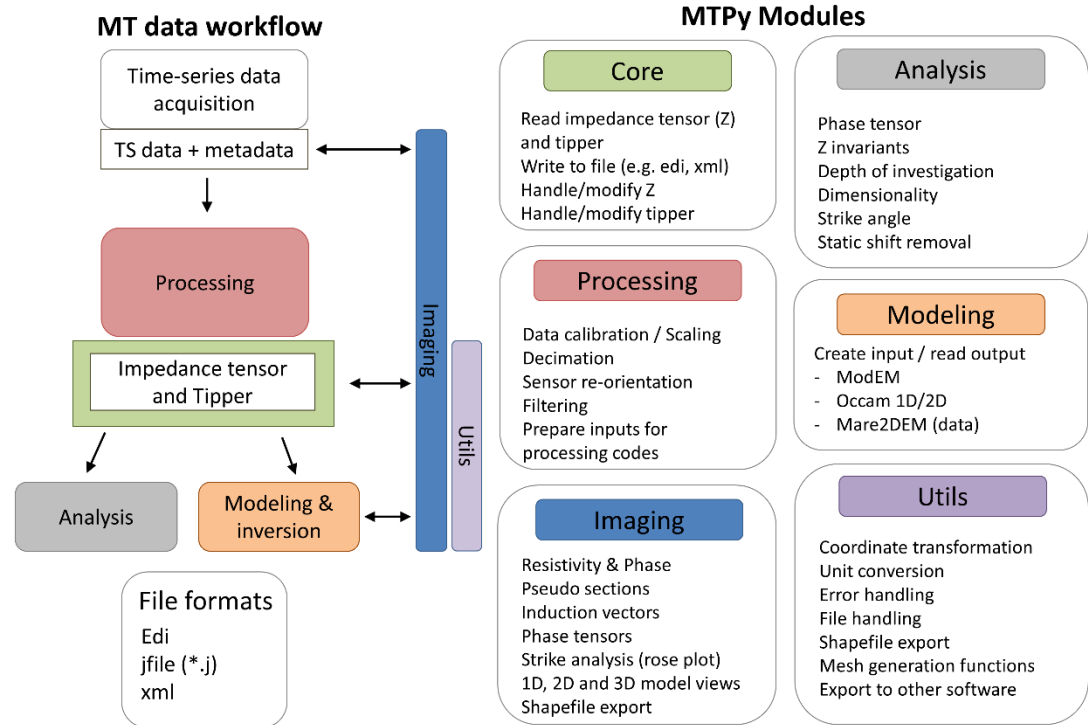
 The right column has a yellow header "Compound Data Types" and text explaining that lists are one of the compound data types that Python understands, and they are indexed, sliced, and manipulated with other built-in functions.

MTPy overview

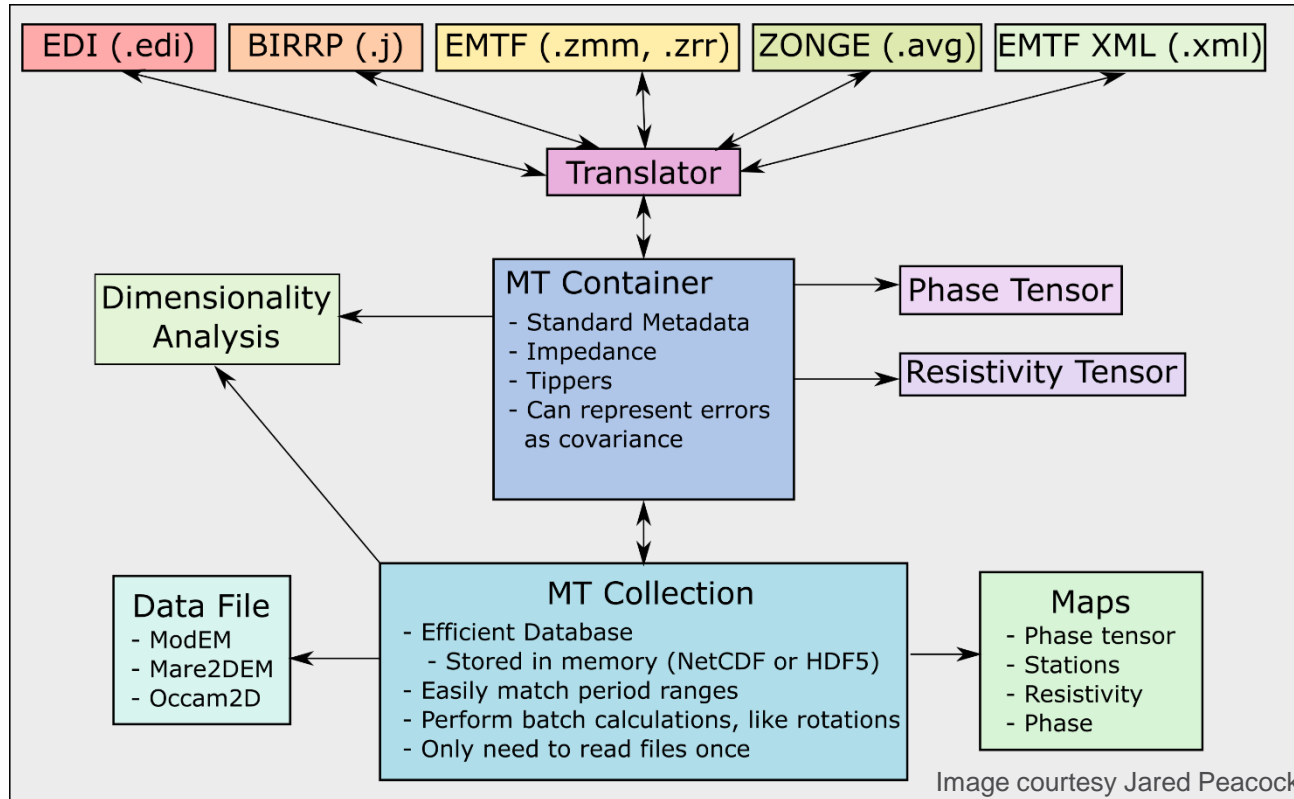
- Structured around the main analysis steps for MT
- Core, Processing, Analysis, Modeling, Imaging + Utils to support all functionality
- Interlinked
- This presentation will cover Core, Analysis, Imaging, Modeling, and Utils

MTPy structure (modified from Krieger and Peacock, 2014)

MTPy Key Functionality



MTPy v2.0 – draft structure



Core

Basic handling of frequency domain (processed) data

Reading in of station metadata (e.g. location etc)

Conversions from Z to apparent resistivity and phase and vice versa (via **utils** modules)

Functions include: reading edi file, rewriting, handling edi file set. Allows you to rewrite the data with some change made (e.g. interpolate onto different frequencies).

Some of the core modules link to **imaging** to allow plotting.

Analysis

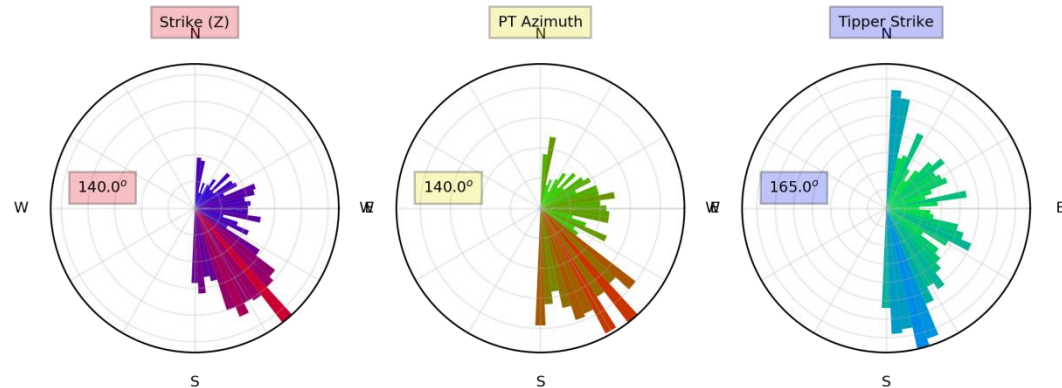
Covers analysis of MT data completed prior to modelling

Dimensionality, strike analysis, phase tensors, static shift calculation, penetration depth, invariants

Links with **core** for reading the data, **utils** to carry out some of the analyses and **imaging** to view the results

In this presentation we will cover:

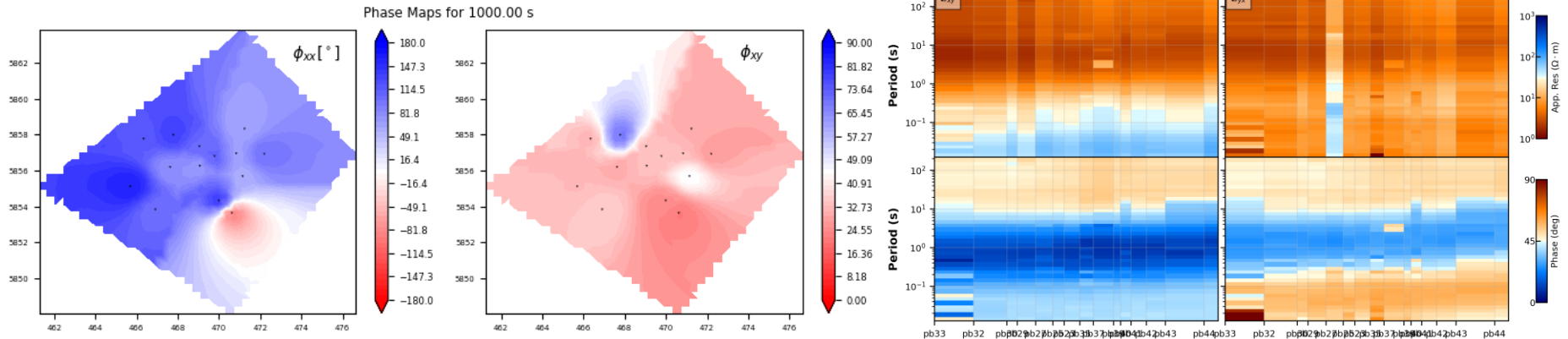
- dimensionality
- strike analysis
- phase tensor (plots)
- penetration depth



Imaging

Plotting of impedance/resistivity/phase data as well as the results from modules within **analysis**, **modeling**. Includes:

- Data plots (e.g. resistivity/phase at a single site and on maps)
- **Analysis** plots (e.g. strike angle, phase tensors, penetration depth)
- **Model** plots (e.g. data/response, resistivity model maps/slices)



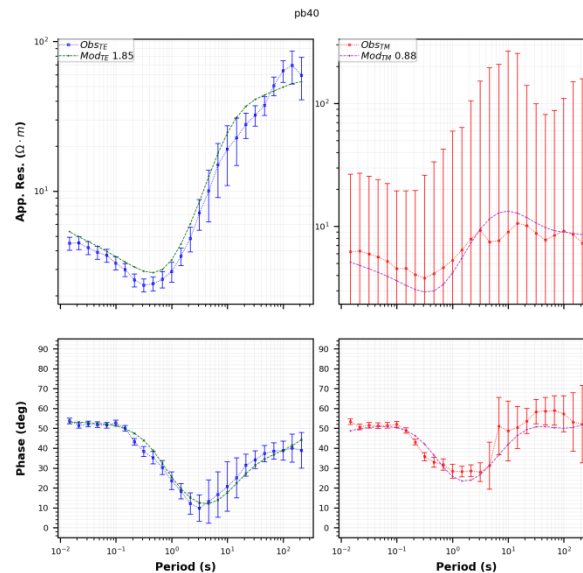
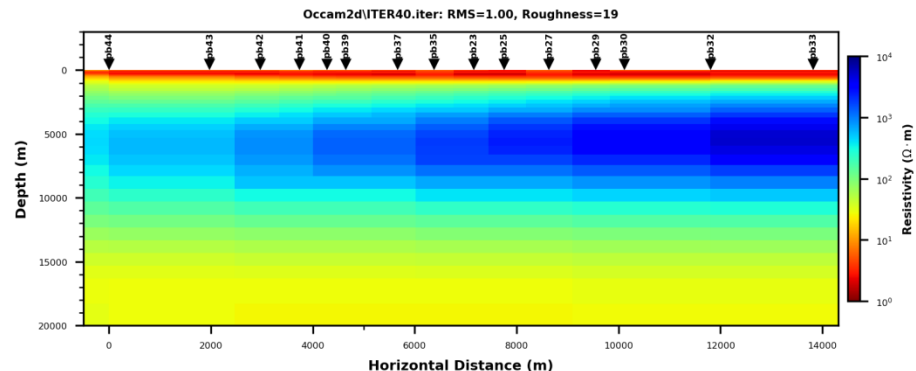
Modeling

Wrapper for commonly used modelling/inversion packages

- Occam1D
- Occam2D
- Mare2DEM (data file only)
- ModEM

Creates input files and allows visualisation of outputs.

Also allows conversion of outputs to other formats (e.g. Gocad sgrid)



Utils

Contains much of the functionality that underlies all other modules, e.g.:

- Projections and transformations (gis_tools)
- Conversion to external formats (shapefiles, gocad, convert_modem_data_to_geogrid)
- Filehandling
- Calculator

+ more....

- array2raster.py
- Ascii_replace_res_values.py
- basemap_tools.py
- calculator.py
- concatenate_input.py
- configfile.py
- convert_modem_data_to_geogrid.py
- edi_folders.py
- exceptions.py
- filehandling.py
- gis_tools.py
- gocad.py
- matplotlib_utils.py
- mesh_tools.py
- mtpy_decorator.py
- mtpylog.py
- plot_geotiff_imshow.py
- plot_rms_iterations.py
- shapefiles.py
- shapefiles_creator.py

https://github.com/MTgeophysics/mtpy

github.com/MTgeophysics/mtpy

Search or jump to... Pulls Issues Marketplace Explore

MTgeophysics / mtpy

Unwatch 25 Unstar 62 Fork 28

Code Issues 30 Pull requests 1 Actions Projects 3 Wiki Security

develop

Go to file Add file Code

alkirkby change dtype of station names array so that it displays... yesterday 4,268

bin	tweak installation script to get env right	2 years ago
data	ALAMP-98 moved date out of tests directory	3 years ago
docs	Add images for new install guide	11 months ago
examples	another attempt at fixing tests	6 days ago
legacy	Add original mare2d script to legacy	7 months ago
mtpy	change dtype of station names array so that it di...	yesterday

About

Python toolbox for standard Magnetotelluric (MT) data analysis

Readme

GPL-3.0 License

Releases 2

v1.0 Latest on May 21, 2019

https://github.com/MTgeophysics/mtpy

← → ↻ 🏠 github.com/MTgeophysics/mtpy 🔍 ☆ 👁 ⚙️ 👤 Error ⋮

📁 Apps 📁 Suggested Sites 📁 Imported From IE

- Kirkby, A.L., Zhang, F., Peacock, J., Hassan, R., Duan, J., 2019. The MTPy software package for magnetotelluric data analysis and visualisation. *Journal of Open Source Software*, 4(37), 1358. <https://doi.org/10.21105/joss.01358>
- Krieger, L., and Peacock, J., 2014. MTPy: A Python toolbox for magnetotellurics. *Computers and Geosciences*, 72, p167-175. <https://doi.org/10.1016/j.cageo.2014.07.013>

Overview

A Python Toolbox for Magnetotelluric (MT) Data Processing, Analysis, Modelling and Visualization

- Home Page: <https://github.com/MTgeophysics/mtpy>
- API Documentation: <http://mtpy2.readthedocs.io/en/develop/>
- Issue tracking: <https://github.com/MTgeophysics/mtpy/issues>
- Installation Guide (Wiki Pages): <https://github.com/MTgeophysics/mtpy/wiki>
- User Guide: <https://github.com/MTgeophysics/mtpy/blob/develop/docs/MTPy%20User%20Guide.pdf>

Note that this repository has superseded the [geophysics/mtpy](#) and [GeoscienceAustralia/mtpy2](#)

https://github.com/MTgeophysics/mtpy

The screenshot shows the GitHub repository page for `MTgeophysics/mtpy`. The browser address bar displays the URL. The repository name is `MTgeophysics / mtpy`. The page includes navigation tabs for `Code`, `Issues` (30), `Pull requests` (1), `Actions`, `Projects` (3), `Wiki` (circled in red), and `Security`. The `Wiki` tab is highlighted with a red circle. Below the navigation, there are buttons for `Go to file`, `Add file`, and `Code`. The repository description is "Python toolbox for standard Magnetotelluric (MT) data analysis". The `Releases` section shows version `v1.0` as the latest release, dated May 21, 2019. A table of files and folders is visible, including `bin`, `data`, `docs`, `examples`, `legacy`, and `mtpy`.

File/Folder	Description	Last Updated
alkirkby	change dtype of station names array so that it displays...	yesterday 4,268
bin	tweak installation script to get env right	2 years ago
data	ALAMP-98 moved date out of tests directory	3 years ago
docs	Add images for new install guide	11 months ago
examples	another attempt at fixing tests	6 days ago
legacy	Add original mare2d script to legacy	7 months ago
mtpy	change dtype of station names array so that it di...	yesterday

Wiki (Installation guide)

<https://github.com/MTgeophysics/mtpy/wiki/MTPy-installation-guide-for-Windows-10-and-Ubuntu-18.04>

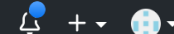
← → ↻ 🏠 github.com/MTgeophysics/mtpy/wiki 🔍 ☆ 👁 ⚙️ 👤 ⋮

📱 Apps 🌐 Suggested Sites 📁 Imported From IE



Search or jump to...

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



[MTgeophysics](#) / [mtpy](#)

👁 Unwatch 25 ☆ Unstar 62 🍴 Fork 28

[Code](#) [Issues 30](#) [Pull requests 2](#) [Actions](#) [Projects 3](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Home

[Edit](#) [New Page](#)

Bren Moushall edited this page on Jun 3, 2020 · 17 revisions

Welcome to the MTPy wiki!

MTPy Installation Guides

- [Installation Guide \(02-04-2020\)](#)

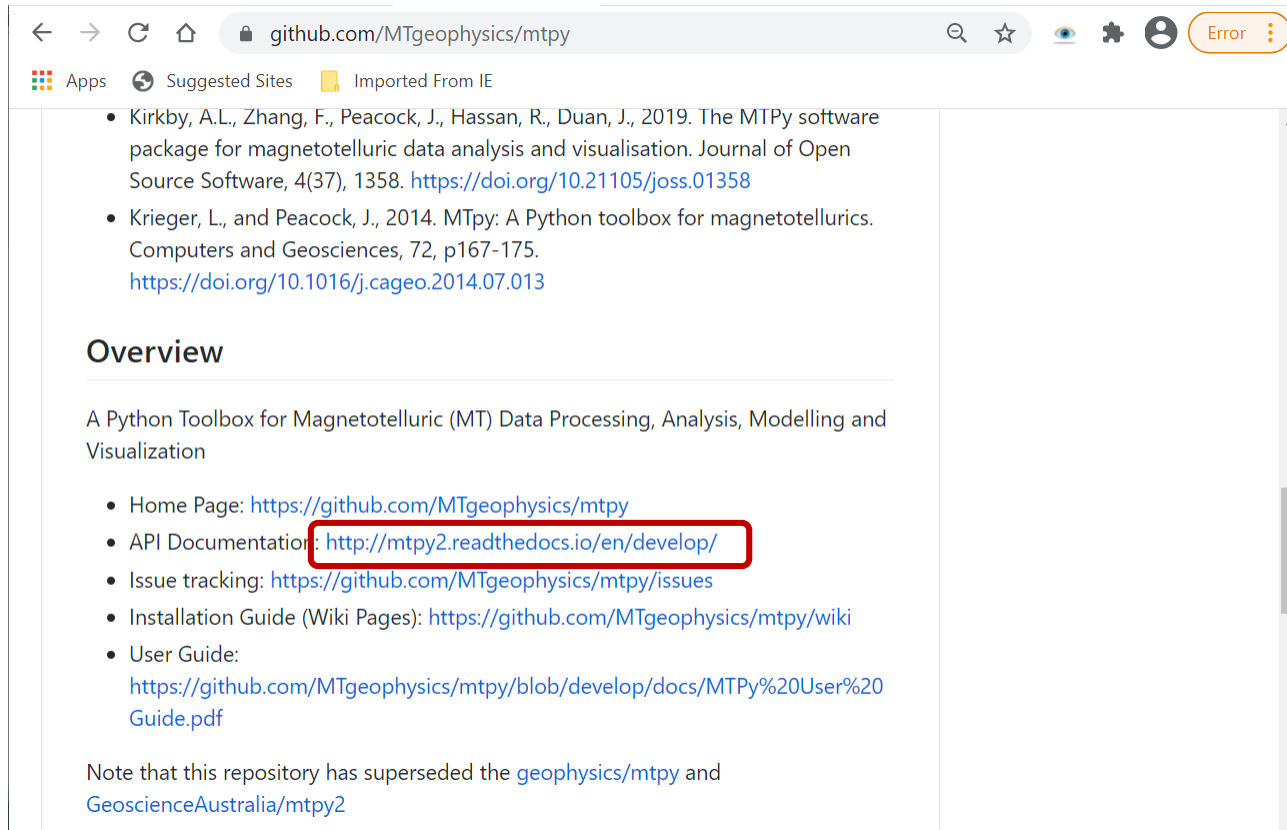
Pages 5

Find a Page...

[Home](#)

[MTPy installation guide for Linux system](#)

https://github.com/MTgeophysics/mtpy



github.com/MTgeophysics/mtpy

Apps Suggested Sites Imported From IE

- Kirkby, A.L., Zhang, F., Peacock, J., Hassan, R., Duan, J., 2019. The MTPy software package for magnetotelluric data analysis and visualisation. Journal of Open Source Software, 4(37), 1358. <https://doi.org/10.21105/joss.01358>
- Krieger, L., and Peacock, J., 2014. MTPy: A Python toolbox for magnetotellurics. Computers and Geosciences, 72, p167-175. <https://doi.org/10.1016/j.cageo.2014.07.013>

Overview

A Python Toolbox for Magnetotelluric (MT) Data Processing, Analysis, Modelling and Visualization

- Home Page: <https://github.com/MTgeophysics/mtpy>
- API Documentation: <http://mtpy2.readthedocs.io/en/develop/>
- Issue tracking: <https://github.com/MTgeophysics/mtpy/issues>
- Installation Guide (Wiki Pages): <https://github.com/MTgeophysics/mtpy/wiki>
- User Guide: <https://github.com/MTgeophysics/mtpy/blob/develop/docs/MTPy%20User%20Guide.pdf>

Note that this repository has superseded the [geophysics/mtpy](#) and [GeoscienceAustralia/mtpy2](#)

Documentation

<https://mtpy2.readthedocs.io/en/develop/>

Pulled from comments in the modules

Some modules ****may**** be missing

If it's missing, let us know

```
Editor - C:\mtpywin\mtpy\mtpy\modeling\modem\plot_slices.py
mt.py x z.py x plot_penetration_depth3d_2.py x plot_geology.py x plot_slices.py x
25
26 __all__ = ['PlotSlices']
27
28
29 class PlotSlices(object):
30     """
31     * Plot all cartesian axis-aligned slices and be able to scroll through the model
32     * Extract arbitrary profiles (e.g. along a seismic line) from a model
33
34     :Example: ::
35
36         >>> import mtpy.modeling.modem as modem
37         >>> mfn = r"/home/modem/Inv1/Modular_NLCG_100.rho"
38         >>> dfn = r"/home/modem/Inv1/ModEM_data.dat"
39         >>> pds = ws.PlotSlices(model_fn=mfn, data_fn=dfn)
40
41     =====
42     Buttons          Description
43     =====
44     'e'              moves n-s slice east by one model block
45     'w'              moves n-s slice west by one model block
```

Welcome to MTPy's documentation! Indices and tables

next | modules

Table Of Contents

Welcome to MTPy's documentation!
Indices and tables

Next topic
Package Core

This Page
Show Source

Quick search
 Go

Welcome to MTPy's documentation!

Contents:

- Package Core
 - Module z
 - Module TS
 - Module MT
 - Module EDI
 - Module EDI_Collection
 - Module XML
 - Module JFile
- Package Analysis
 - Module Distortion
 - Module Geometry
 - Module Phase Tensor
 - Module Static Shift
 - Module Z Invariants
- Package Modeling
 - Module ModEM
 - Module Occam 1D
 - Module Occam 2D
 - Module Winglink
 - Module WS3DINV
- Package Imaging
 - Penetration Depth
 - Module Plot Phase Tensor Maps
 - Module PlotPhaseTensorPseudoSection
 - Module MTPlot
 - Plot MT Response
 - Visualization of Models
- Package utils
 - Shapefile Creator
 - GIS Tools
 - Other Tools

https://github.com/MTgeophysics/mtpy

← → ↻ 🏠 github.com/MTgeophysics/mtpy 🔍 ☆ 👁 ⚙️ 👤 Error

🌐 Apps 🌐 Suggested Sites 📁 Imported From IE

🐙 Search or jump to... / Pulls Issues Marketplace Explore 🔔 + 🌐

📁 MTgeophysics / mtpy 📺 Unwatch 25 ⭐ Unstar 62 🍴 Fork 28

<> Code **🔔 Issues 30** 🔄 Pull requests 1 ▶️ Actions 📁 Projects 3 📖 Wiki 🛡️ Security ...

🔗 develop ▾ Go to file Add file ▾ [Code](#) ▾ About ⚙️

alkirkby change dtype of station names array so that it displays... yesterday 4,268

bin	tweak installation script to get env right	2 years ago
data	ALAMP-98 moved date out of tests directory	3 years ago
docs	Add images for new install guide	11 months ago
examples	another attempt at fixing tests	6 days ago
legacy	Add original mare2d script to legacy	7 months ago
mtpy	change dtype of station names array so that it di...	yesterday

Python toolbox for standard Magnetotelluric (MT) data analysis

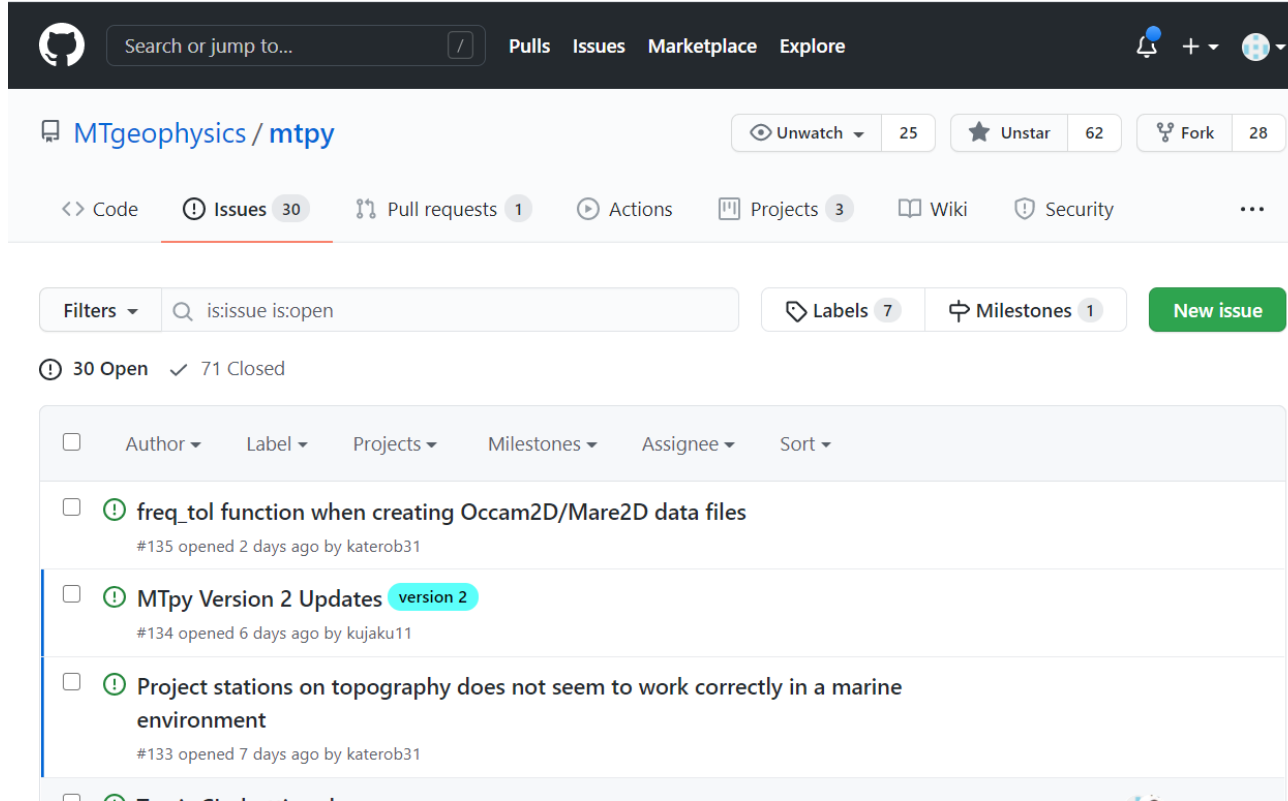
📖 Readme

📄 GPL-3.0 License

Releases 2

📦 v1.0 **Latest** on May 21, 2019

Issues



The screenshot shows the GitHub interface for the repository MTgeophysics / mtpy. At the top, there is a search bar and navigation links for Pulls, Issues, Marketplace, and Explore. Below this, the repository name is displayed along with statistics for Unwatch (25), Unstar (62), and Fork (28). A secondary navigation bar includes links for Code, Issues (30), Pull requests (1), Actions, Projects (3), Wiki, and Security. A filter bar shows the search query 'is:issue is:open' and buttons for Labels (7) and Milestones (1), with a prominent green 'New issue' button. The main content area displays a list of issues with columns for checkboxes, Author, Label, Projects, Milestones, Assignee, and Sort. Three issues are visible: #135 'freq_tol function when creating Occam2D/Mare2D data files', #134 'MTPy Version 2 Updates' (marked as 'version 2'), and #133 'Project stations on topography does not seem to work correctly in a marine environment'.

Search or jump to... / Pulls Issues Marketplace Explore

MTgeophysics / mtpy Unwatch 25 Unstar 62 Fork 28

<> Code Issues 30 Pull requests 1 Actions Projects 3 Wiki Security ...

Filters is:issue is:open Labels 7 Milestones 1 New issue

30 Open ✓ 71 Closed

<input type="checkbox"/>	Author	Label	Projects	Milestones	Assignee	Sort
<input type="checkbox"/>	🚨	freq_tol function when creating Occam2D/Mare2D data files				
<input type="checkbox"/>	🚨	MTPy Version 2 Updates version 2				
<input type="checkbox"/>	🚨	Project stations on topography does not seem to work correctly in a marine environment				

Getting started - installing MTPy

We recommend installing the following:

- Anaconda Python distribution: <https://anaconda.org/anaconda/python>
- Git shell (if on windows) (e.g. git for windows <https://git-scm.com/download/win>)

We suggest installing these into a common directory so it does not interfere with other installations

Once these are installed, you can clone the MTPy repository

```
>> git clone https://github.com/MTgeophysics/mtpy.git
```


Getting started - installing MTPy

Install dependencies

```
cd mtpy
```

```
>> conda install gdal libgdal geopandas netcdf4 pyyaml
```

```
>> pip install obspy
```

To get updated MTPy (recommend doing this every day):

```
>> git pull
```

Examples folder

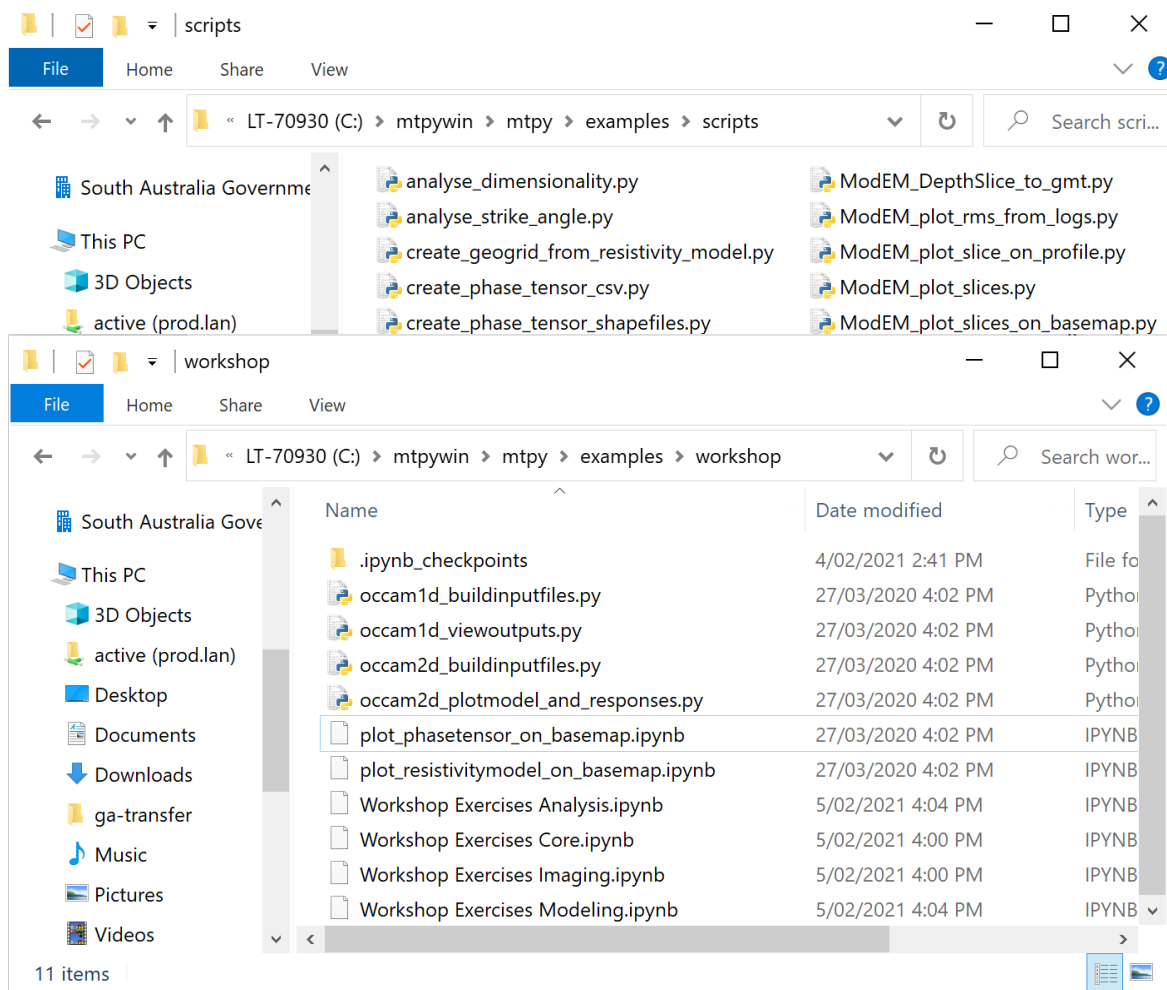
mtpy/examples/scripts

- Example scripts

mtpy/examples/notebooks &

mtpy/examples/workshop

- Example notebooks



Running MTPy

We recommend using either:

- Jupyter notebook, OR
- Development environment (e.g. Spyder, PyCharm)

Jupyter notebook

Web browser based

Run python code or comments

Run one cell at a time and it stores the result in memory

Can write individual cells to a python script

```
%%writefile example.py
```

Can also export to latex, html – great for demonstration

The exercises for this workshop are in Jupyter format

can run each of the cells, modifying the inputs to suit your requirements. Most of these examples have been written to be self contained.

In Jupyter, you can add the following line to the top of any cell and it will write the contents of that cell to a python script: `%%writefile example.py`

You can also select multiple cells and copy them to a new Jupyter notebook.

Many of the examples below make use of the matplotlib colour maps. Please see https://matplotlib.org/examples/color/colormaps_reference.html for colour map options.

Core

These first few examples cover some of the basic functions and tools that can be used to look at data contained in an edi file, plot it, and make changes (e.g. sample onto different frequencies).

Reading an edi file into an MT object

```
In [1]: # import required modules
from mtpy.core.mt import MT

# Define the path to your edi file
edi_file = "C:/mtpywin/mtpy/examples/data/edi_files_2/Synth00.edi"

# Create an MT object
mt_obj = MT(edi_file)
```

The `mt_obj` contains all the data from the edi file, e.g. impedance, tipper, frequency as well as station information (lat/long). To look at any of these parameters you can type, for example:

```
In [2]: # To see the latitude and longitude
print mt_obj.lat, mt_obj.lon
```

-19.01 136.01

Spyder

Interface for working with Python code

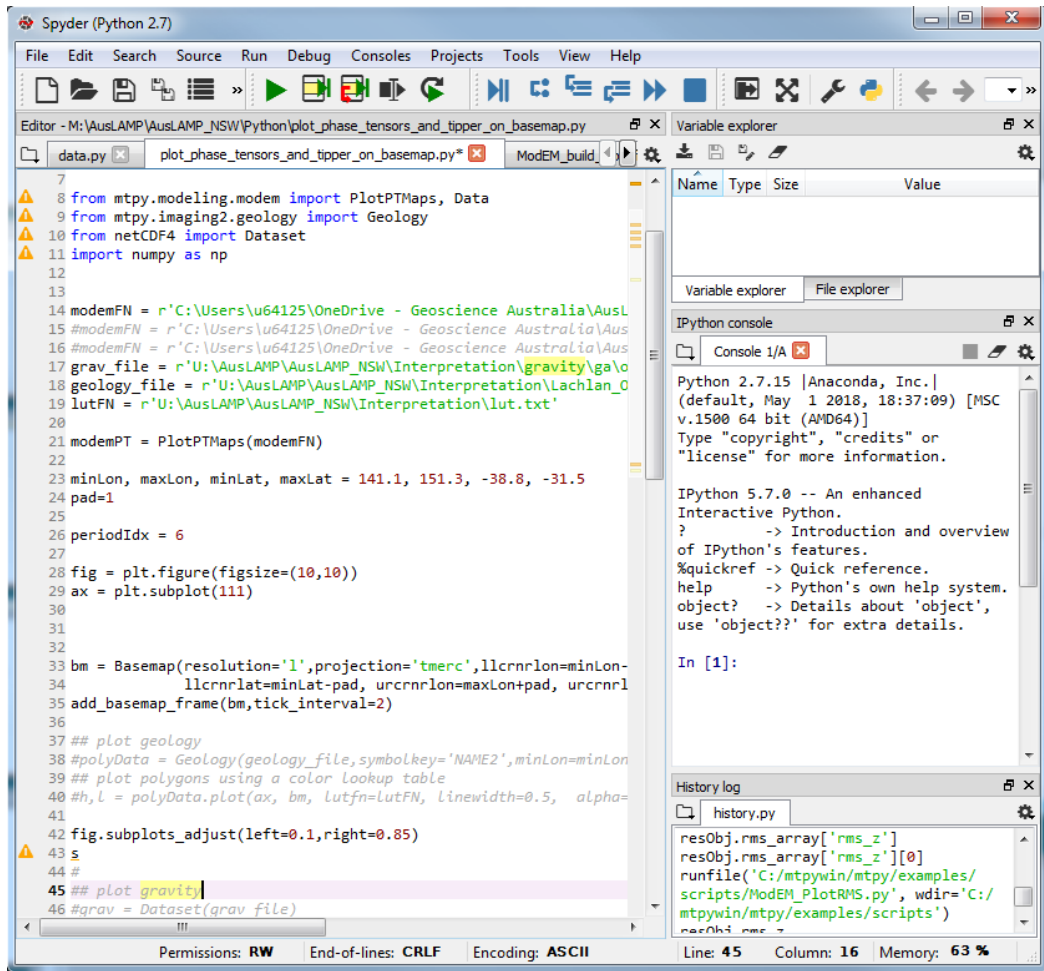
Interactive – warnings for syntax errors etc

Easy to read code

Stores variables in the console allowing you to inspect them later.

Options for pop-out / inline plots

Great for day-to-day use



The screenshot displays the Spyder Python IDE interface. The main window is titled "Spyder (Python 2.7)". The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains various icons for file operations, running, and debugging. The editor window shows a Python script named "plot_phase_tensors_and_tipper_on_basemap.py" with the following code:

```
7
8 from mtpy.modeling.modem import PlotPTMaps, Data
9 from mtpy.imaging2.geology import Geology
10 from netCDF4 import Dataset
11 import numpy as np
12
13
14 modemFN = r'C:\Users\u64125\OneDrive - Geoscience Australia\AusL
15 #modemFN = r'C:\Users\u64125\OneDrive - Geoscience Australia\Aus
16 #modemFN = r'C:\Users\u64125\OneDrive - Geoscience Australia\Aus
17 grav_file = r'U:\AusLAMP\AusLAMP_NSW\Interpretation\gravity\gao
18 geology_file = r'U:\AusLAMP\AusLAMP_NSW\Interpretation\Lachlan_0
19 lutFN = r'U:\AusLAMP\AusLAMP_NSW\Interpretation\lut.txt'
20
21 modemPT = PlotPTMaps(modemFN)
22
23 minLon, maxLon, minLat, maxLat = 141.1, 151.3, -38.8, -31.5
24 pad=1
25
26 periodIdx = 6
27
28 fig = plt.figure(figsize=(10,10))
29 ax = plt.subplot(111)
30
31
32
33 bm = Basemap(resolution='l',projection='tmerc',llcrnrlon=minLon-
34 llcrnrlat=minLat-pad, urcrnrlon=maxLon+pad, urcrnrl
35 add_basemap_frame(bm,tick_interval=2)
36
37 ## plot geology
38 #polyData = Geology(geology_file,symbolkey='NAME2',minLon=minLon
39 ## plot polygons using a color lookup table
40 #h,l = polyData.plot(ax, bm, Lutfn=LutFN, Linewidth=0.5, alpha=
41
42 fig.subplots_adjust(left=0.1,right=0.85)
43 s
44 #
45 ## plot gravit
46 #grav = Dataset(grav_file)
```

The Variable explorer on the right shows a table with columns Name, Type, Size, and Value. The IPython console shows the Python version (2.7.15) and the IPython version (5.7.0). The History log shows the execution of the script.

Installation of Spyder/Jupyter

Neither of these are installed in the MTPy package so you will need to install them.

```
>> conda install jupyter
```

AND/OR

```
>> conda install spyder
```

```
>> jupyter notebook &
```

```
>> spyder &
```

A word on python objects

Most functions in MTPy underpinned by objects

MT object

- MT.lat, MT.lon, MT.elev, other standard metadata
- Z object (impedance tensor) and Tipper object

Z object

- Z.freq - Frequency array
- Z.z, Z.z_err, Z.resistivity, Z.resistivity_err, Z.phase, Z.phase_err, etc...

Apple



```
Apple.colour = red
Apple.width = 0.07 (m)
Apple.height = 0.08 (m)
Apple.compute_volume()
```

Practical use cases (using AusLAMP data!)

1. Reading, writing an edi file
2. Simple analysis tools (dimensionality, strike angle)
3. Plotting an edi file
4. Penetration depth (Niblett-Bostick transform)
5. Data plots (phase tensor, pseudosections)
6. Inputs and outputs for inversion

Dataset

AusLAMP data from New South Wales and Victoria

If you want to see more.....

Data available from:

<http://dx.doi.org/10.11636/Record.2020.011>

(NSW) and

<http://dx.doi.org/10.11636/Record.2018.021>

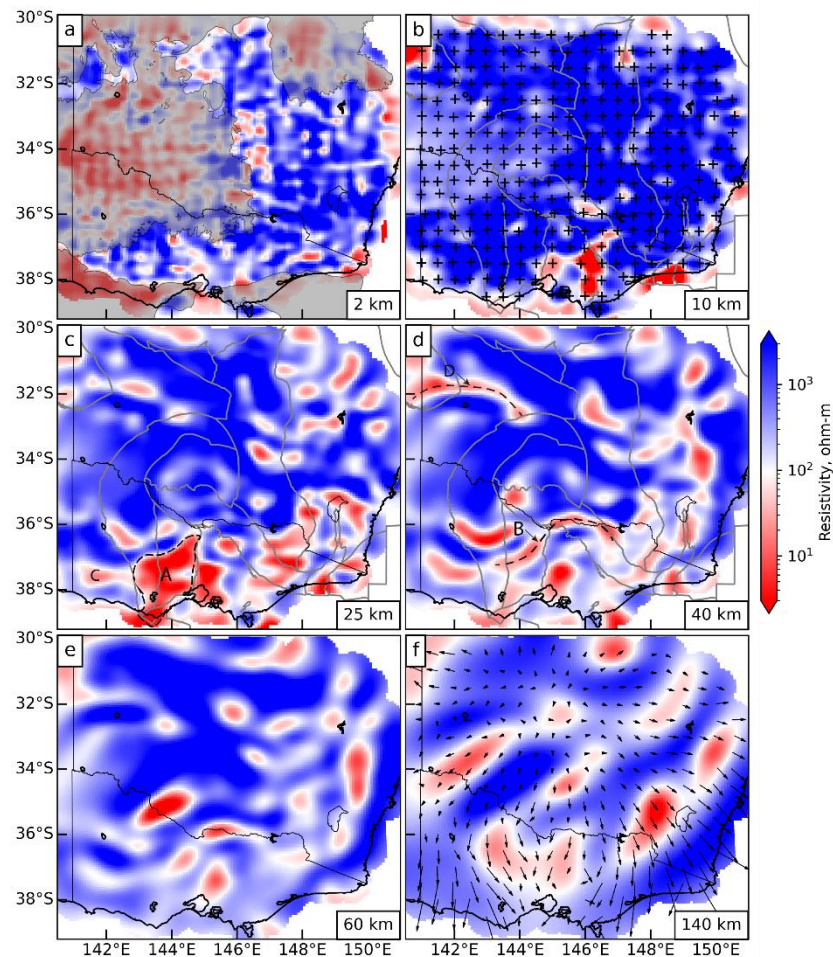
(Victoria)

Model available from:

<http://dx.doi.org/10.26186/131889>

Paper (Kirkby et al 2020) available from:

<https://doi.org/10.1016/j.tecto.2020.228560>



Read an edi file into an MT object

```
# import required modules
from mtpy.core.mt import MT

# Define the path to your edi file
edi_file = r"C:\edifiles\E15.edi"

# Create an MT object
mt_obj = MT(edi_file)
```

Get station location

```
# To see the latitude and longitude
print(mt_obj.lat, mt_obj.lon)
```

```
-30.497572222222225 148.04341111111111
```

```
# To see the easting, northing, and elevation
print(mt_obj.east, mt_obj.north, mt_obj.elev)
```

```
600131.4714907524 6625614.777263684 134.0
```

Interrogate the data

```
# e.g. to see the frequencies in the file
print(mt_obj.Z.freq)
```

```
[2.00000e-01 1.56250e-01 1.25000e-01 1.00000
e-01 7.93651e-02 6.32911e-02
 5.00000e-02 3.96825e-02 3.16456e-02 2.51256
e-02 1.99203e-02 1.56250e-02
 1.25000e-02 1.00000e-02 7.93651e-03 6.32911
e-03 5.00000e-03 3.96825e-03
 3.16456e-03 2.51256e-03 1.99203e-03 1.56250
e-03 1.25000e-03 1.00000e-03
 7.93651e-04 6.32911e-04 5.00000e-04 3.96825
e-04 3.16456e-04 2.51256e-04
 1.99203e-04 1.56250e-04 1.25000e-04 1.00000
e-04 7.93651e-05 6.32911e-05
 5.00000e-05 3.96825e-05 3.16456e-05 2.51256
e-05 1.99203e-05 1.56250e-05
 1.25000e-05 1.00000e-05]
```

Interrogate the data (cont'd)

```
print(mt_obj.Z.z)
```

```
[[[-5.456737e-02-0.0147217j  2.296910e+00+0.6397  
567j ]  
  [-1.869760e+00-0.9006823j  3.510076e-01-0.0603  
598j ]]  
  
[[-1.001216e-01-0.06415853j  2.333180e+00+0.6731  
247j ]  
  [-1.789870e+00-0.8923373j  3.530071e-01-0.0458  
7761j]]
```

```
print(mt_obj.Z.z_err)
```

```
[[0.03469457 0.05178892]  
 [0.02801203 0.04181375]]  
  
[[0.01620979 0.02159036]  
 [0.01234237 0.01643919]]  
  
[[0.00705637 0.00857404]  
 [0.00551617 0.00670258]]
```

```
print(mt_obj.Z.resistivity)
```

```
[[1.00940774e-02 1.79648772e+01]  
 [1.36108586e+01 4.00845113e-01]]  
  
[[5.65626070e-02 2.35873031e+01]  
 [1.59996019e+01 5.06875071e-01]]  
  
[[1.19901742e-01 2.95245908e+01]  
 [1.91212334e+01 6.85415226e-01]]  
  
[[2.03307091e-01 3.50150890e+01]  
 [2.17167443e+01 8.51211575e-01]]
```

```
print(mt_obj.Z.phase)
```

```
[[ -147.34800433  16.09290219]  
 [-153.50151171  -7.40478723]]  
  
[[ -137.29146041  16.25317148]  
 [-153.13346078   0.85364649]]  
  
[[ -118.14735977  16.63432523]  
 [-152.02876069   5.70966883]]  
  
[[  -99.20005198  17.31974842]  
 [-149.55405135  13.14943136]]
```

Editing data

E.g. interpolate to 5 periods per decade

```
# 5 periods per decade from 10^-4 to 10^3 seconds
from mtpy.utils.calculator import get_period_list
new_freq_list = 1./get_period_list(10,3e3,5)

# Create new Z and Tipper objects containing interpolated data
new_Z_obj, new_Tipper_obj = mt_obj.interpolate(new_freq_list)

# Write a new edi file using the new data
mt_obj.write_mt_file(
    save_dir=r'C:\tmp',
    fn_basename='E15_5ppd',
    file_type='edi',
    new_Z_obj=new_Z_obj, # z object
    new_Tipper_obj=new_Tipper_obj, # tipper object
    longitude_format='LONG', # write longitudes as 'LONG'
    latlon_format='dd' # write as decimal degrees
)
```

'C:\\tmp\\E15_5ppd_2.edi'

Dimensionality

based on phase tensor ellipses (Caldwell et al 2004, Bibby et al 2005)

```
# Import required modules
from mtpy.analysis.geometry import dimensionality

#Have a look at the dimensionality
print(dimensionality(z_object = mt_obj.Z,
                    skew_threshold = 3, # threshold for 3D
                    eccentricity_threshold=0.1 # threshold for 2D
                    ))
```

```
[1 1 1 1 1 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2
 2 2 2 1
 2 3 1 1 1 1 1]
```

Strike angle

```
from mtpy.analysis.geometry import strike_angle

# calculate strike
strike = strike_angle(z_object = mt_obj.Z,
                      skew_threshold = 3, # threshold for 3D
                      eccentricity_threshold=0.1 # threshold for 2D
                      )

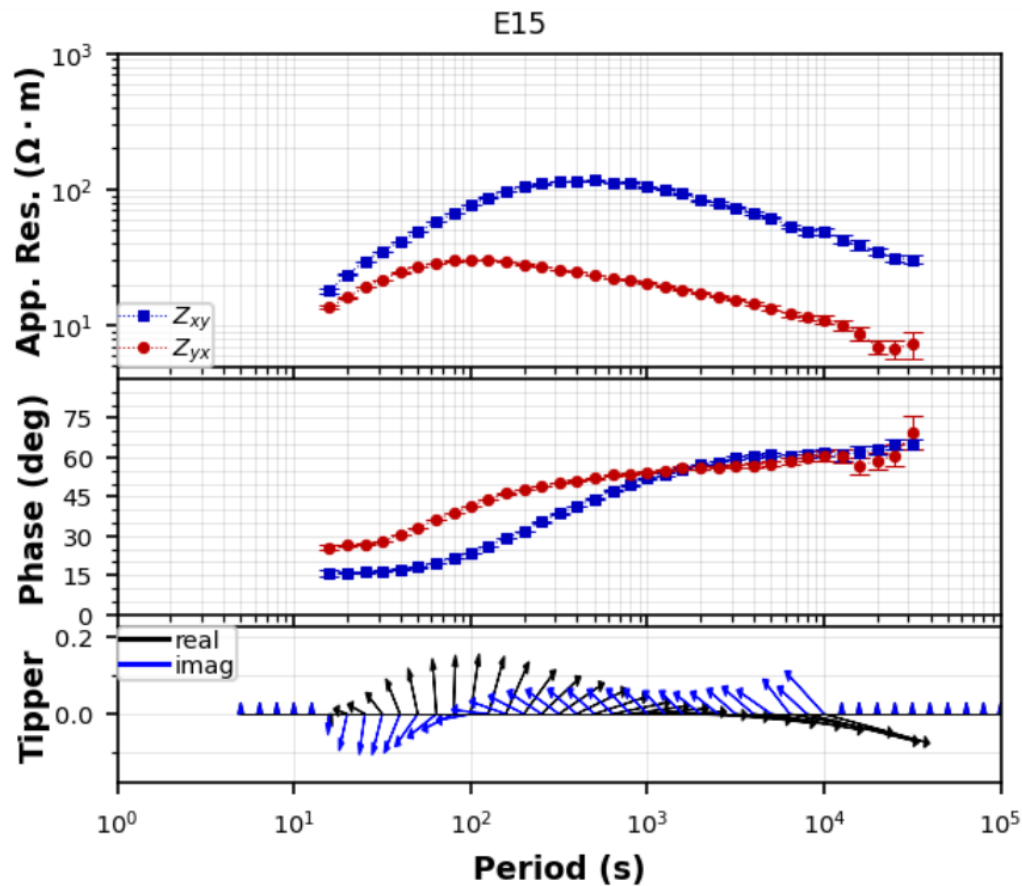
# display the strike angle for each frequency
# two columns because of 90 degree ambiguity in strike
print(strike)
```

```
[[-4.24687911  85.75312089]
 [ 0.53546544 -89.46453456]
 [ 1.71204294 -88.28795706]
 [ 2.14208192 -87.85791808]
 [          nan          nan]
 [ 2.2151065  -87.7848935 ]
 [-0.50130871  89.49869129]
 [          nan          nan]
```

Many other attributes, e.g. skew angle β , invariants, etc....

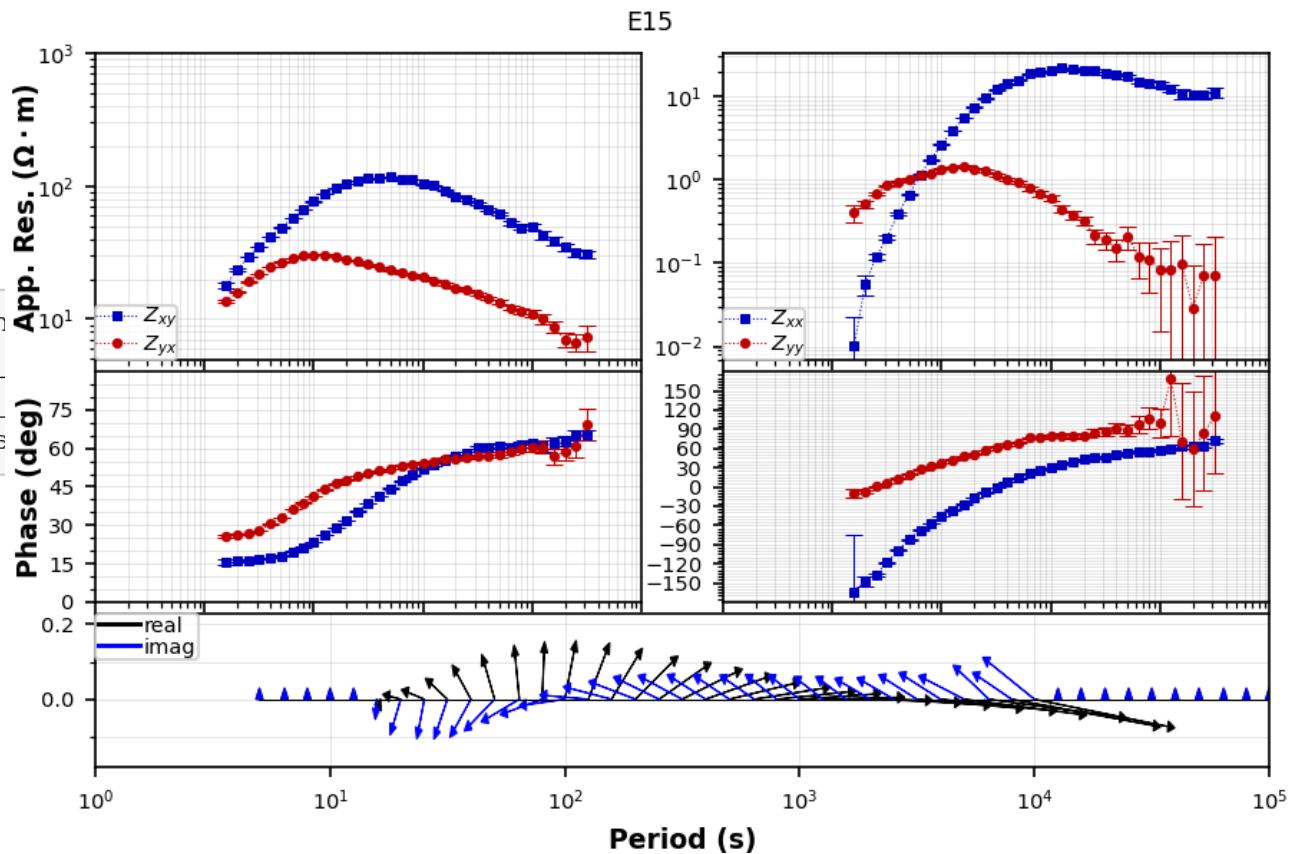
Plotting

```
ptobj = mt_obj.plot_mt_response(plot_num=  
    plot_tipper = 'yr',  
    plot_pt='n',  
    fig_size=(4,3.5))
```



Plotting (4 components)

```
ptobj = mt_obj.plot_mt_respon  
plot_  
plot_  
fig_s
```

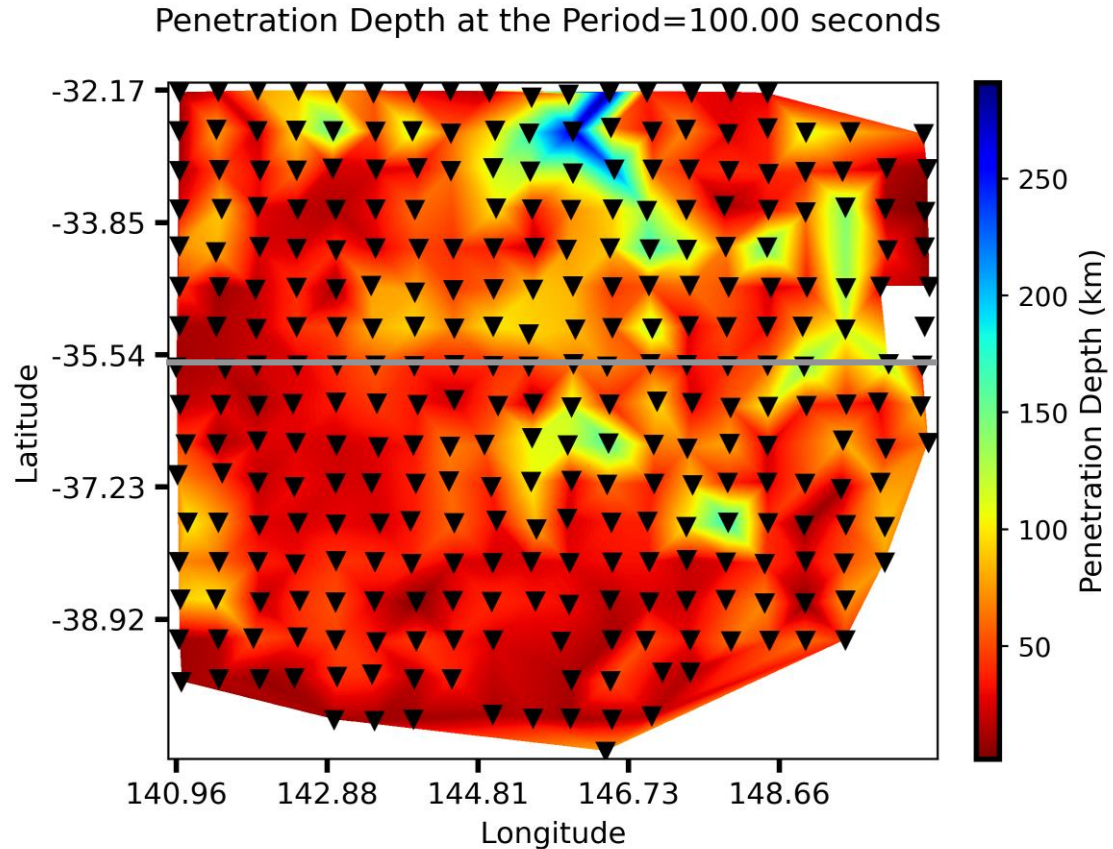


Penetration depth (3d)

```
# Import required modules
from mtpy.imaging import penetration_depth3d

# Set path to edi files to include in the plot
edi_path = r'C:/data/edifiles'

# Create plot for 100s (determinant)
pen3d.plot_latlon_depth_profile(edi_path,
                                100., 'det',
                                showfig=True,
                                ptol=0.2,
                                savefig=True,
                                savepath=r'C:/data/pen3d.png',
                                fontsize=11,
                                fig_dpi=400,
                                file_format='png')
```



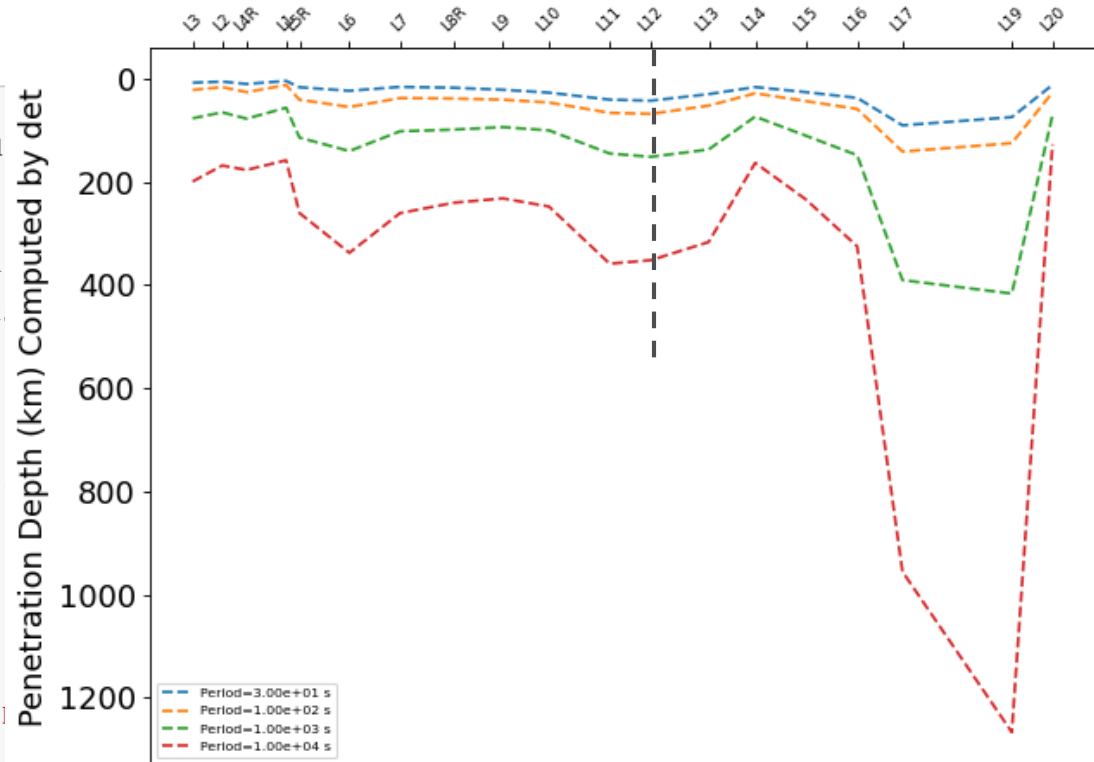
Penetration depth (2d)

```
# Import required modules
from mtpy.imaging import penetration_depth2d

# path and list of edi files
edi_path = r'C:/data/edifiles'
edi_list = [os.path.join(edi_path, ff) for ff
            os.listdir(edi_path) if ff.start

# Choose indices of periods to plot
period_list = [30., 100., 1000., 10000.]

# Plot profiles for different modes ('det', '
pen2d.plot2Dprofile(edi_path, period_list,
                    edi_list=edi_list,
                    ptol=0.2,
                    zcomponent='det',
                    period_by_index = False,
                    save=True,
                    savepath=r'C:\tmp\pen2d.
                    )
```



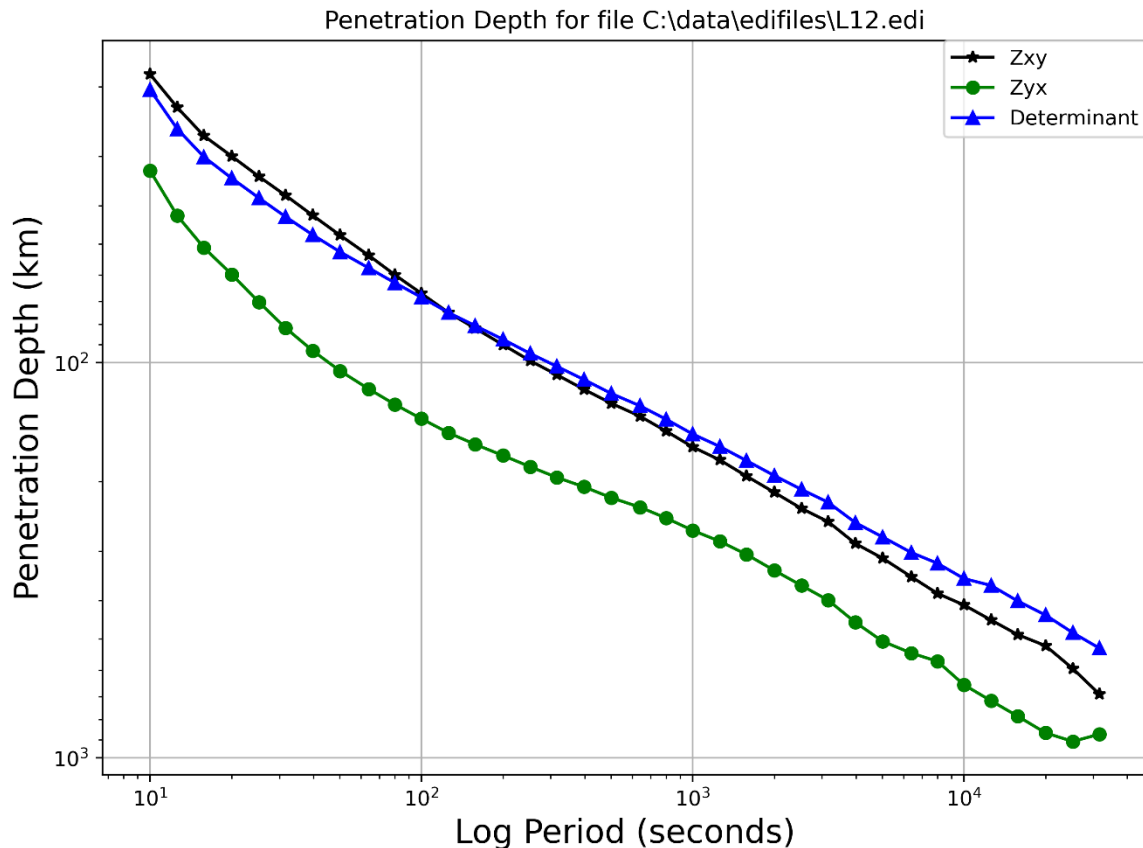
Penetration depth (1d)

```
# Import required modules
import os
from mtpy.imaging import penetration_dep

# Define edi file
edi_file = r"C:/data/edifiles/L12.edi"

# Define file to save to
save_file = r"C:/tmp/penetration_depth1c

# Create a plot of penetration depth and
pd1d.plot_edi_file(edi_file,
                   savefile=save_file,
                   fig_dpi=400)
```



Phase tensor maps

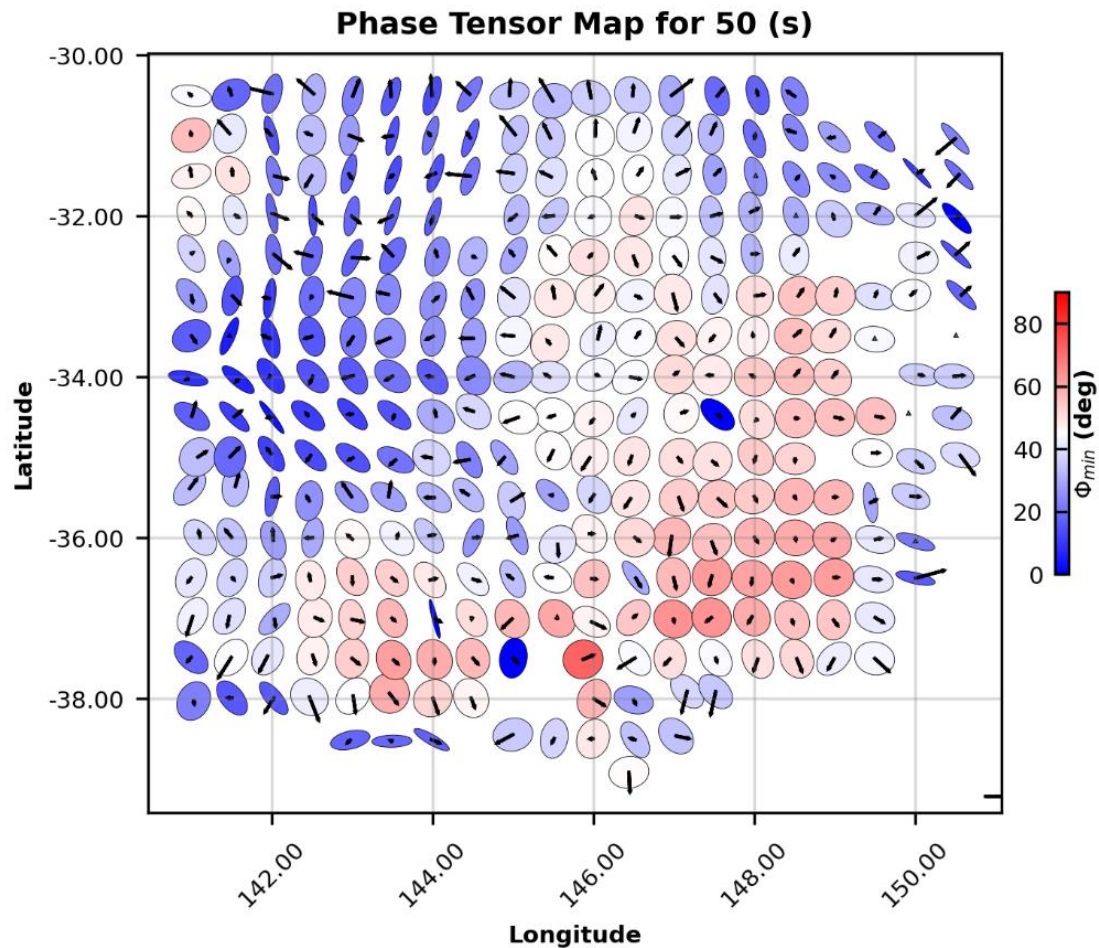
(Caldwell et al 2004, Bibby et al

```
# Import required modules
from mtpy.imaging.phase_tensor_maps import PlotPhaseTensorMaps
import os

# directory containing edis
edi_path = r'C:/edifiles'

# gets edi file names as a list
edi_list = [os.path.join(edi_path, ff) for ff in \
            os.listdir(edi_path) if ff.endswith('.e

# generate plot
ptmap = PlotPhaseTensorMaps(fn_list = edi_list,
                             plot_freq = 0.02, # frequency to p
                             fig_size = (4,3), # dimensions of f
                             xpad = 0.5, ypad = 0.5, # pad around
                             plot_tipper = 'yr', # 'y' + 'r' and
                             edgecolor='k', # colour or None
                             lw=0.2, # linewidth for the ellipse
                             minorticks_on=False,
                             ellipse_colorby='phimin', # or 'phim
                             ellipse_range = [0,90], # [min,max,s
                             ellipse_size=0.5, # ellipse scaling
                             arrow_size=0.5,
                             ellipse_cmap='bwr' # matplotlib col
                             )
```

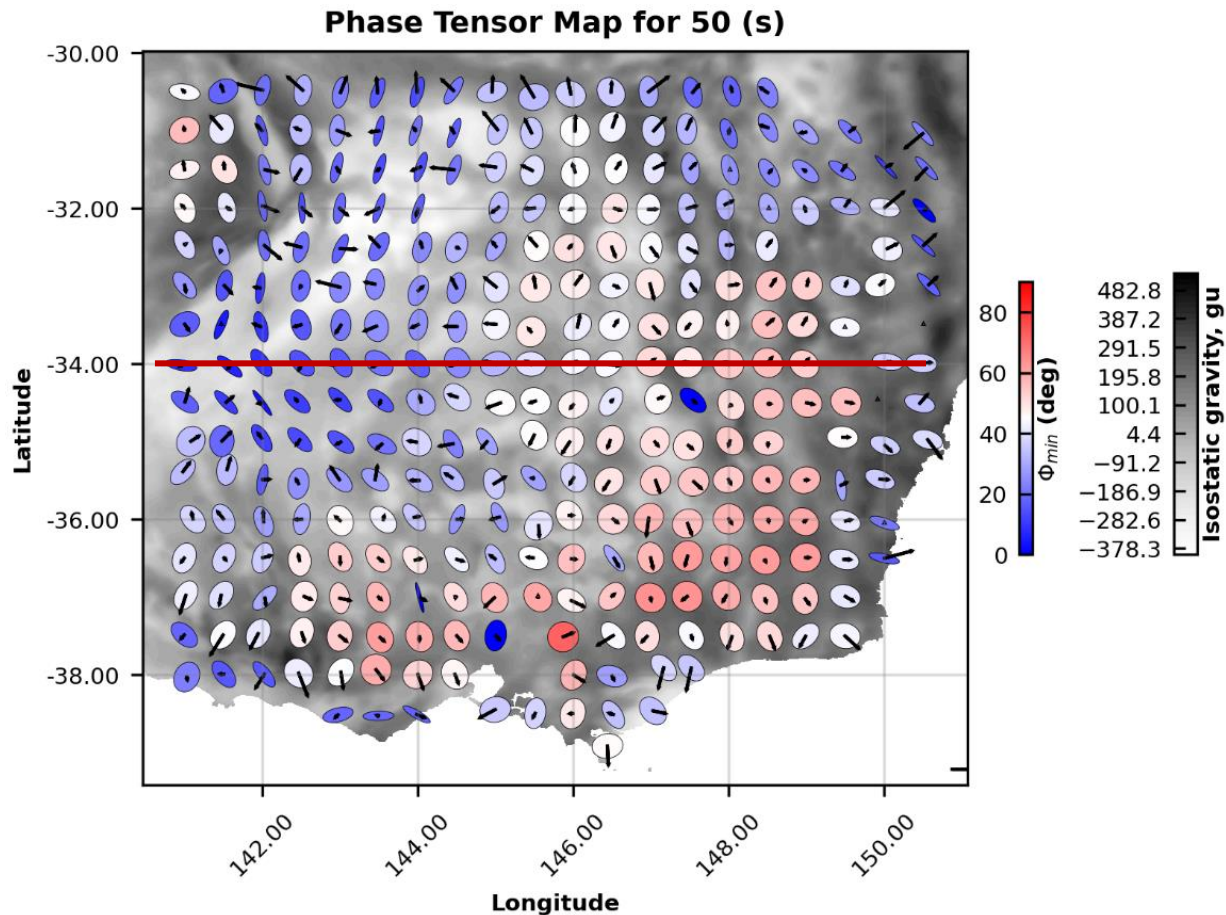


Phase tensors + gravity (for example)

```
# Import the netCDF4 module
from netCDF4 import Dataset
import numpy as np

# load a netcdf file
grav = Dataset(r'C:\data\Isostatic_v2_2')
glon, glat = np.meshgrid(grav.variables
                        grav.variables
vals = grav.variables['Band1'][:])

ptmap.plot(show=True,
           raster_dict={'lons':glon,
                       'lats':glat,
                       'vals':vals,
                       'levels':50,
                       'cmap':'Greys',
                       'cbar_title':'I.',
                       'cbar_position'
```



Phase tensor section

```
# import required modules
from mtpy.imaging.phase_tensor_pseudosection import
PlotPhaseTensorPseudoSection
import os

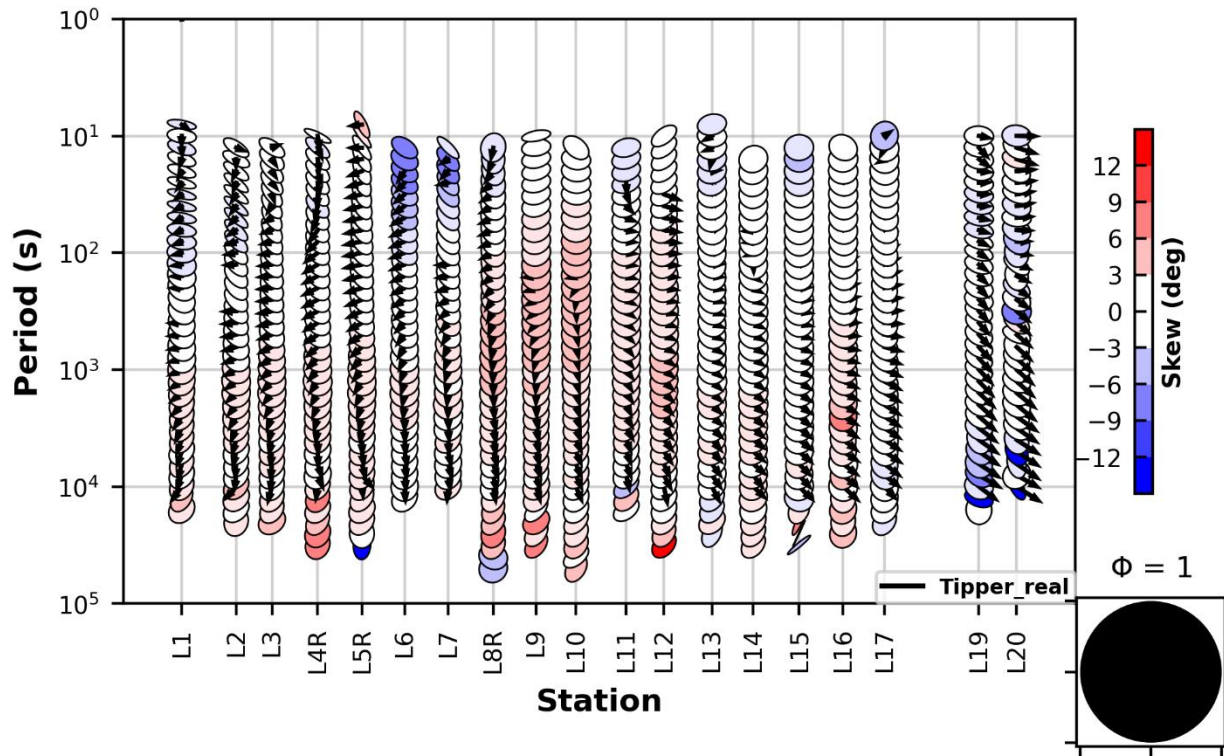
# path to edis and list
edi_path = r'C:/data/edifiles'
edi_list=[os.path.join(edi_path,edi) for e
          if ((edi.endswith('.edi')) and e

# dictionary containing ellipse properties
ellipse_dict = {'ellipse_colorby':'skew_se
                'ellipse_range':[-12,12,3]

# create a plot object
ptsection = PlotPhaseTensorPseudoSection(f
    linedir='ew', # 'ns' or 'e
    stretch=(6,8), # plot (x,y
    plot_tipper = 'yr', # plot
                  # 'ri'
    ellipse_dict = ellipse_dic
    )

ptsection.plot()

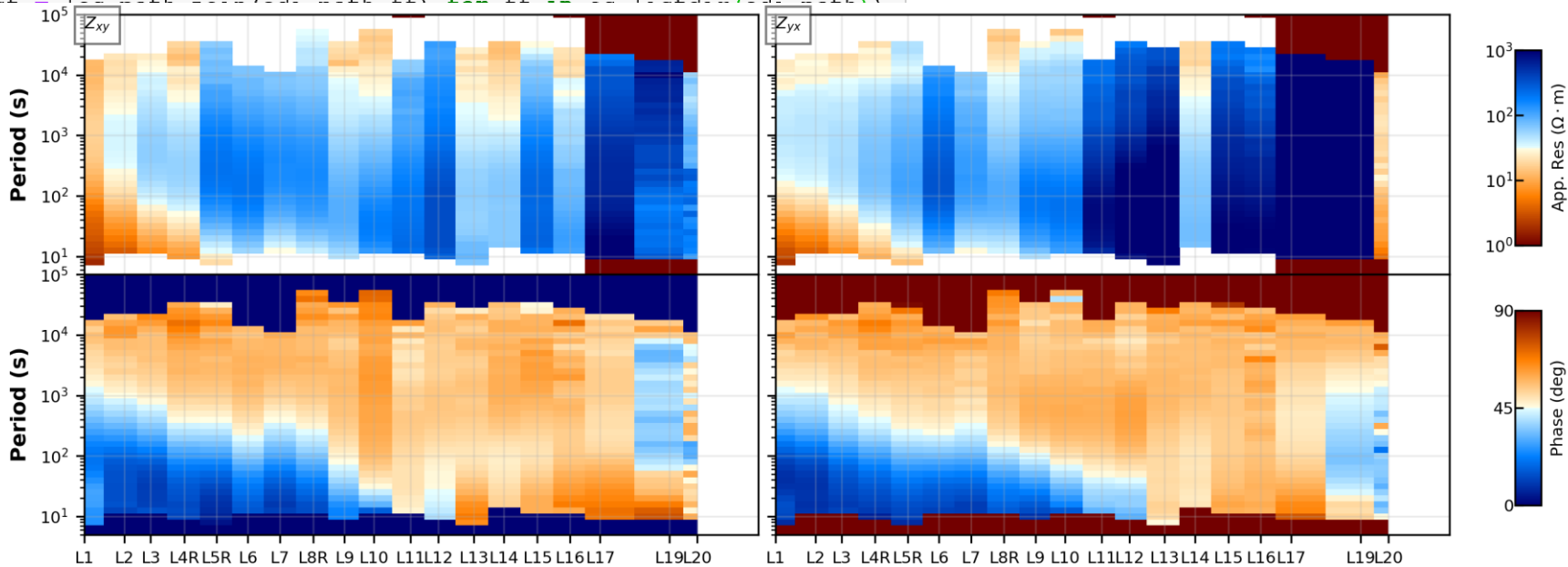
ptsection.save_figure(save_fn = r'C:/tmp/P
                      fig_dpi=400)
```



Resistivity, phase pseudosection

```
# Import required modules
import os
from mtpy.imaging.plotpseudosection import PlotResPhasePseudoSection

edi_path = r'C:\data\edifiles'
```



Creating inversion inputs

Available for:

Occam 1d

Occam 2d

Mare2DEM (data file only)

ModEM 3D

Occam 1D

This example demonstrates how to create input files for inversion of MT data using the Occam 1D code (Key 2009, Constable et al 1987) - <http://marineemlab.ucsd.edu/Projects/Occam/1DCSEM/>.

There are three input files.

- Data file, containing the data for inversion
- Model file (which contains the model mesh)
- Startup file - called by Occam1d and contains the data and model file names together with some control parameters for the inversion.

Occam 1D

```
# import required modules
import os
import mtpy.modeling.occam1d as mtc1d

# full path to edi file and save path
edi_file = r'C:\data\edifiles\E6.edi'
savepath = r'C:\tmp\Occam1D_inv'

# create data file
ocd = mtc1d.Data()
ocd.write_data_file(edi_file=edi_file,
                    mode='te', # 'te', 'tm', or 'det'
                    save_path=savepath, # save path
                    res_errorfloor=1.5, # error floor, %
                    phase_errorfloor=0.75, # error floor, °
                    remove_outofquadrant=True)
```

Wrote Data File to : C:\tmp\Occam1D_inv\Occam1d_DataFile_TE.dat

```
# create model file
ocm = mtc1d.Model(n_layers = 100, # number of layers
                 target_depth = 150000, # model depth
                 # before padding
                 bottom_layer = 400000, # total depth
                 z1_layer=200 # 1st layer thickness, m
                 )
ocm.write_model_file(save_path=savepath)
```

Wrote Model file: C:\tmp\Occam1D_inv\Model1D

```
# create startup file
ocs = mtc1d.Startup(data_fn=ocd.data_fn,
                   model_fn=ocm.model_fn,
                   max_iter=200, # maximum iterations
                   target_rms=1.0
                   )
ocs.write_startup_file()
```

Wrote Input File: C:\tmp\Occam1D_inv\OccamStartup1D

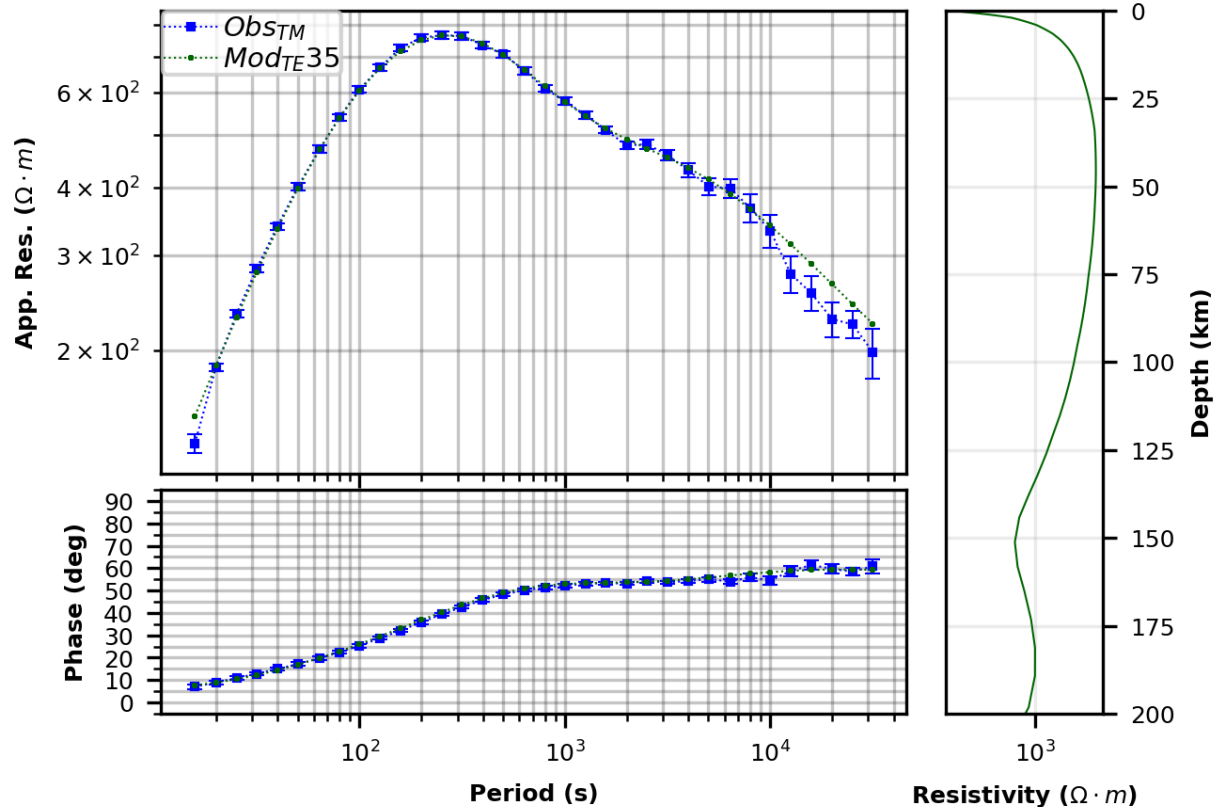
Occam1d – viewing the outputs

```
import mtpy.modeling.occam1d as mtoc
import os.path as op

# working directory
wd = r'C:/tmp/Occam1d_inv'

# model, data, iter and response file
modelfn = op.join(wd, 'Model1D')
datafn = op.join(wd, 'Occam1d_DataFile')
iterfn = op.join(wd, 'ITER_35.iter')
respfn = op.join(wd, 'ITER_35.resp')

# plot the model and response
pr = mtoc1d.Plot1DResponse(data_te_f
                           model_fn
                           resp_te_f
                           iter_te_f
                           depth_lim
                           fig_size =
```



ModEM

The example below demonstrates how to set up input files for 3D inversion using the ModEM inversion code (Egbert and Kelbert 2012; Kelbert et al. 2014)

The ModEM code has three compulsory input files:

- Model file - contains information on the mesh size and starting resistivity
- Data file - contains the impedance tensor and tipper data at each frequency and station locations
- Covariance file - contains model covariance parameters, which control smoothing

ModEM – Data file

- Created first
- Mesh then built around the stations
- Quite a few parameters to tweak, some shown here

```
import os
from mtpy.modeling.modem import Data
from mtpy.utils.calculator import get_period_list

# path containing edi files
edipath = r'C:/data/edifiles'

# find all files in edipath
edi_list = [os.path.join(edipath,ff) for ff in\
             os.listdir(edipath) if (ff.endswith('.edi'))]

# period list to interpolate to (e.g. 4 periods per decade)
period_list = get_period_list(1e-4,10,4)
```

```
# create a data object
do = Data(edi_list=edi_list,
          inv_mode = '1', # invert for Z + tipper
          save_path=r'C:/tmp/ModEM_inv',
          period_list=period_list,
          error_type_z='floor_egbert', # egbert = % of
                                     # sqrt(Zxy*Zyx)
          error_value_z=5., # error floor %
          error_type_tipper = 'floor_abs',
          error_value_tipper=0.01,
          model_epsg=28355) # epsg code for model grid

# write data file
do.write_data_file()
```

ModEM – Model file

Create a model object (reads data object)

Creates a grid centred on the stations

A number of parameters can be tweaked

Option to add topography (arctgis ascii grid) e.g. etopo1

<https://www.ngdc.noaa.gov/mgg/global/>

```
from mtpy.modeling.modem import Model
```

```
# create a Model object
```

```
mo = Model(station_locations=do.station_locations,  
           cell_size_east=7500,  
           cell_size_north=7500,  
           pad_north=7, # number padding cells N-S  
           pad_east=7, # number padding cells E-W  
           pad_z=6, # number of vertical padding cells  
           pad_num=6, # number of cells before padding  
           pad_stretch_v=1.6, # incr. factor padding (vertical)  
           pad_stretch_h=1.9, # incr. factor padding (horizontal))
```

Number of stations = 302

Dimensions:

e-w = 156

n-s = 159

z = 101 (without 7 air layers)

Extensions:

e-w = 2530300.0 (m)

n-s = 2552800.0 (m)

0-z = 973200.0 (m)

Stations rotated by: 0.0 deg clockwise positive from N

** Note ModEM does not accommodate mesh rotations, it assumes all coordinates are aligned to geographic N, E therefore rotating the stations will have a similar effect as rotating the mesh.

```
# project the stations on the topography
```

```
do.project_stations_on_topography(mo)
```

ModEM – Covariance file

Controls smoothing

Reads model file you have just created

Input smoothing value, number of times to apply smoothing

```
from mtpy.modeling.modem import Covariance

# create a covariance file
co = Covariance()
co.smoothing_east = 0.6
co.smoothing_north = 0.6
co.smoothing_z = 0.6
co.write_covariance_file(model_fn=mo.model_fn)
```

Reading C:/tmp/ModEM_inv\ModEM_Model_File.rho

ModEM – check

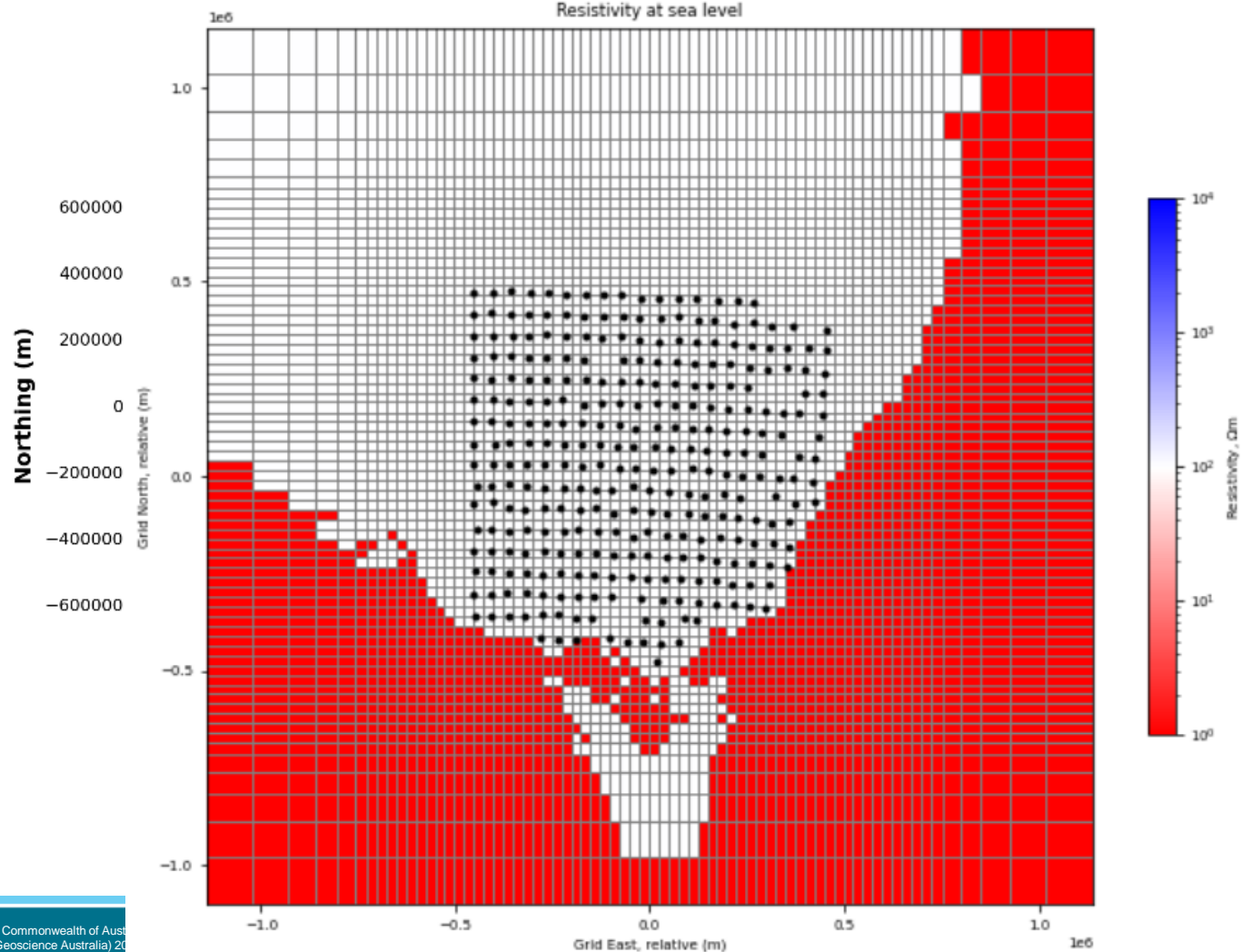
Good idea to open in addition....

Data file – can use t

Model – couple of q

```
mo.plot_mesh()
```

```
mo.plot_sealevel_re
```



ModEM objects

```
from mtpy.modeling.modem import Model
```

```
wd = r'C:\data\modemfiles'
```

```
model_fn = os.path.join(wd, 'Modular_MPI_NLCG_005.rho')
```

```
mObj = Model()
```

```
mObj.read_model_file(model_fn=model_fn)
```

```
mObj.res_model
```

```
array([[3.00000841e-01, 3.00000841e-01, 3.00000841e-01, ...,
        8.83026395e+01, 9.33783949e+01, 9.70125159e+01],
       [3.00000841e-01, 3.00000841e-01, 3.00000841e-01, ...,
        8.49217293e+01, 9.13273222e+01, 9.60433834e+01],
       [3.00000841e-01, 3.00000841e-01, 3.00000841e-01, ...,
        8.31827504e+01, 9.01937379e+01, 9.54822150e+01],
```

```
mObj.grid_east
```

```
array([-1276400., -923600., -737900., -640200., -588800., -5
61700.,
       -547500., -540000., -532500., -525000., -517500., -5
10000.,
```



https://unsplash.com/photos/gDPaDDy6_WE

ModEM objects

```
from mtpy.modeling.modem import Data
```

```
data_fn = os.path.join(wd, 'ModEM_Data.dat')
```

```
dObj = Data()
```

```
dObj.read_data_file(data_fn)
```

```
dObj.center_point.east, dObj.center_point.north
```

```
(array([378901.8964358]), array([6158864.73539792]))
```

```
dObj.period_list
```

```
array([[6.40e+00, 1.00e+01, 1.58e+01, 2.52e+01, 3.98e+01, 6.40e+0  
1,  
1.00e+02, 1.58e+02, 2.52e+02, 3.98e+02, 6.40e+02, 1.00e+0  
3,  
1.58e+03, 2.52e+03, 3.98e+03, 6.40e+03, 1.00e+04, 1.58e+0  
4,  
2.52e+04, 3.98e+04])
```



https://unsplash.com/photos/gDPaDDy6_WE

ModEM – Visualisation of outputs

Many tools to visualise outputs:

- Plotting depth slices and vertical slices
- Overlay with other datasets (e.g. geological linework, seismic)
- Export slices to ArcGIS, Gocad sgrid
- Visualise phase tensors (data, model and residual)
- RMS maps
- Plots of data and model response at individual stations

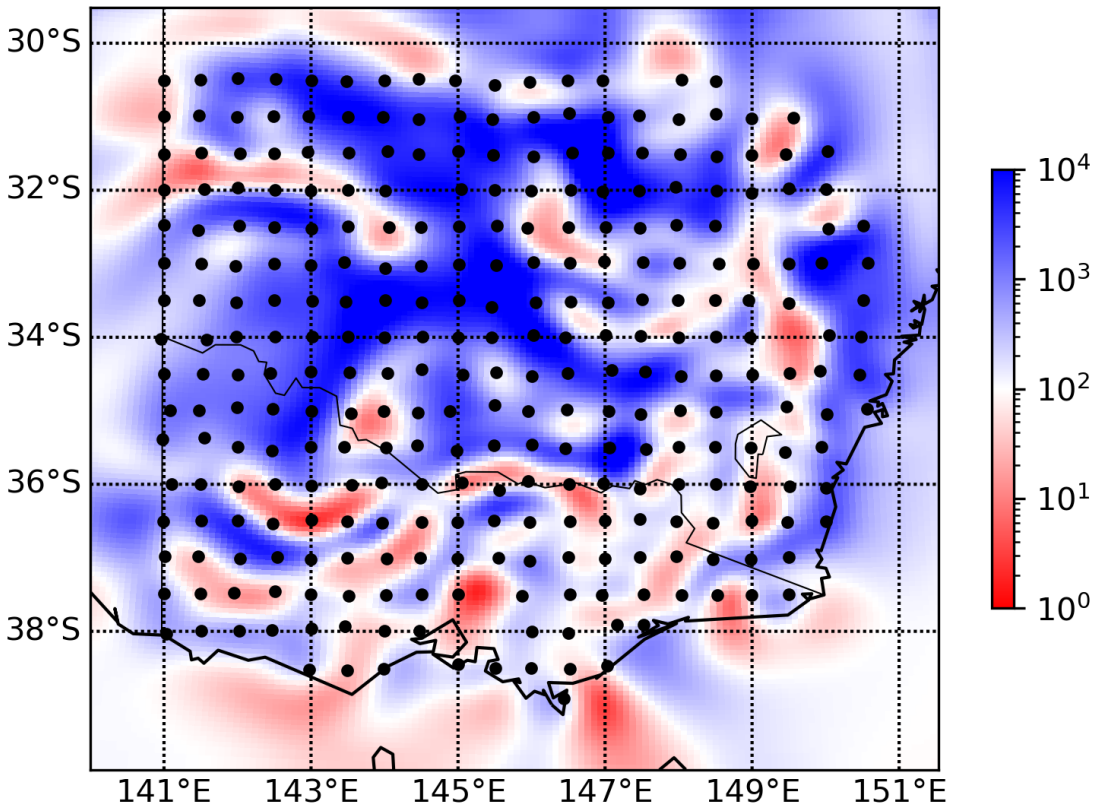
Depth Slice

```
import os
from mtpy.modeling.modem import PlotSlices

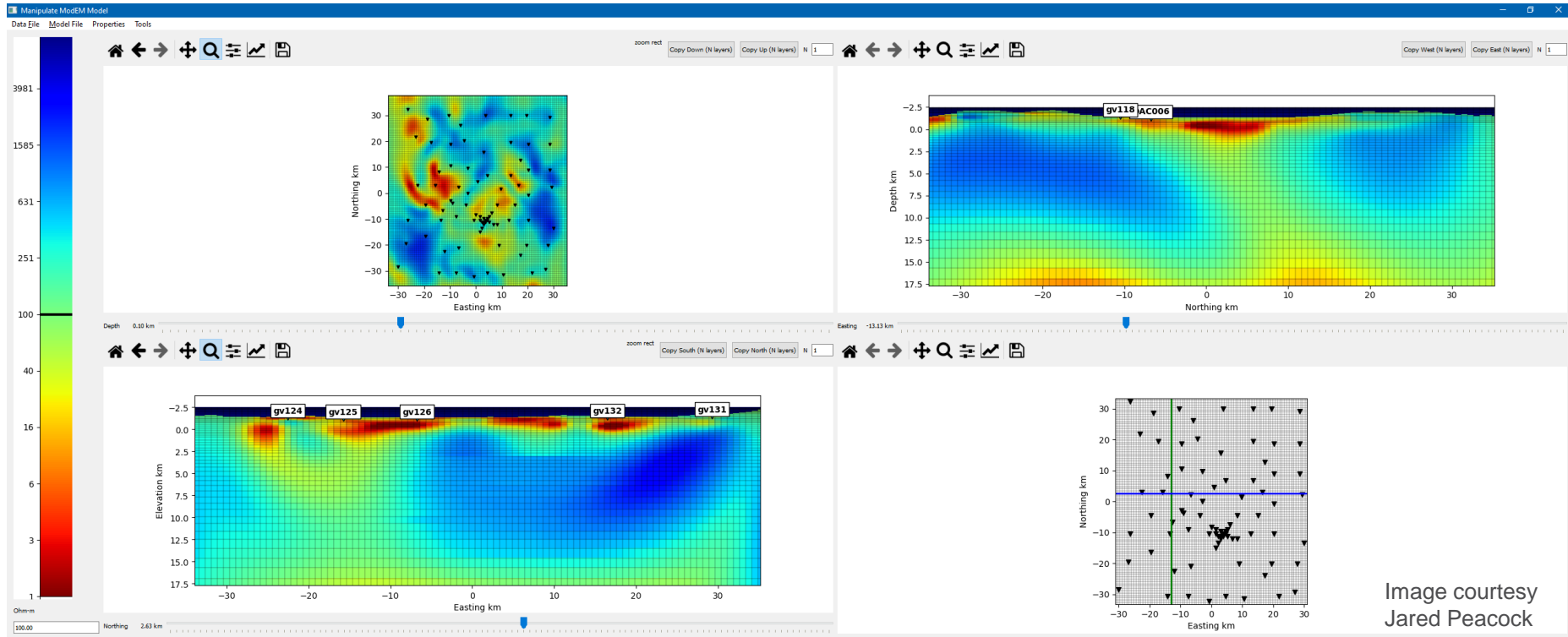
wd = r'C:\data\modemfiles'

model_fn = os.path.join(wd, 'Modular_MPI_1
data_fn = os.path.join(wd, 'ModEM_Data.da
|
ps = PlotSlices(model_fn,
                data_fn=data_fn,
                model_epsg = 28355, # ep
                cmap='bwr_r', # color maj
                climits=(0,4), # log10(c
                plot_stations = True, # j
                draw_colorbar=True, # wh
                fig_size=(6,5),
                plot_yn='n')

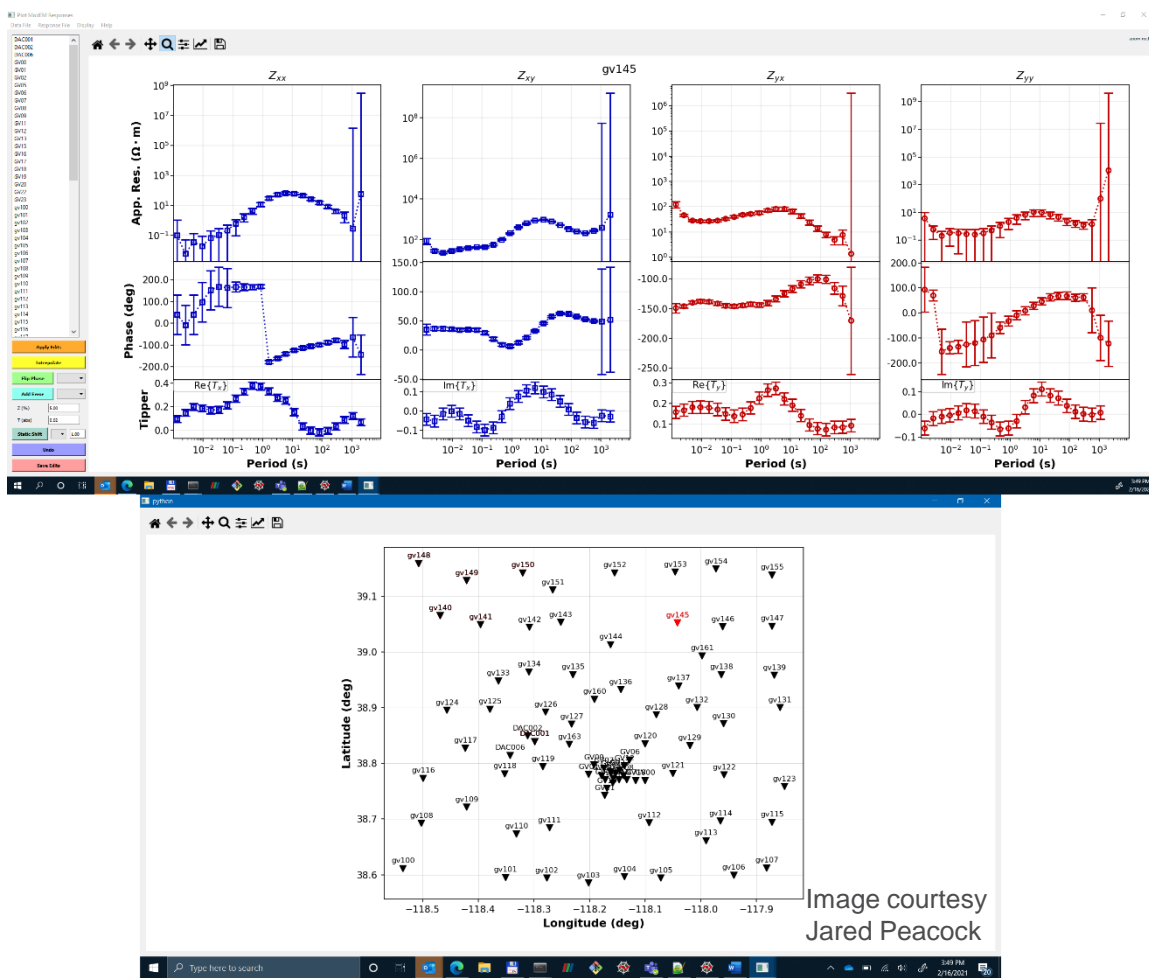
ps.basemap_plot(40e3, # depth to plot (c
                save=True,
                save_path=savepath) # sa
```



Also.. Model Manipulator (soon)



Plot Response (interactive)



Plot Response

```

import os
from mtpy.modeling.modem

wd = r'C:\data\modemfile:

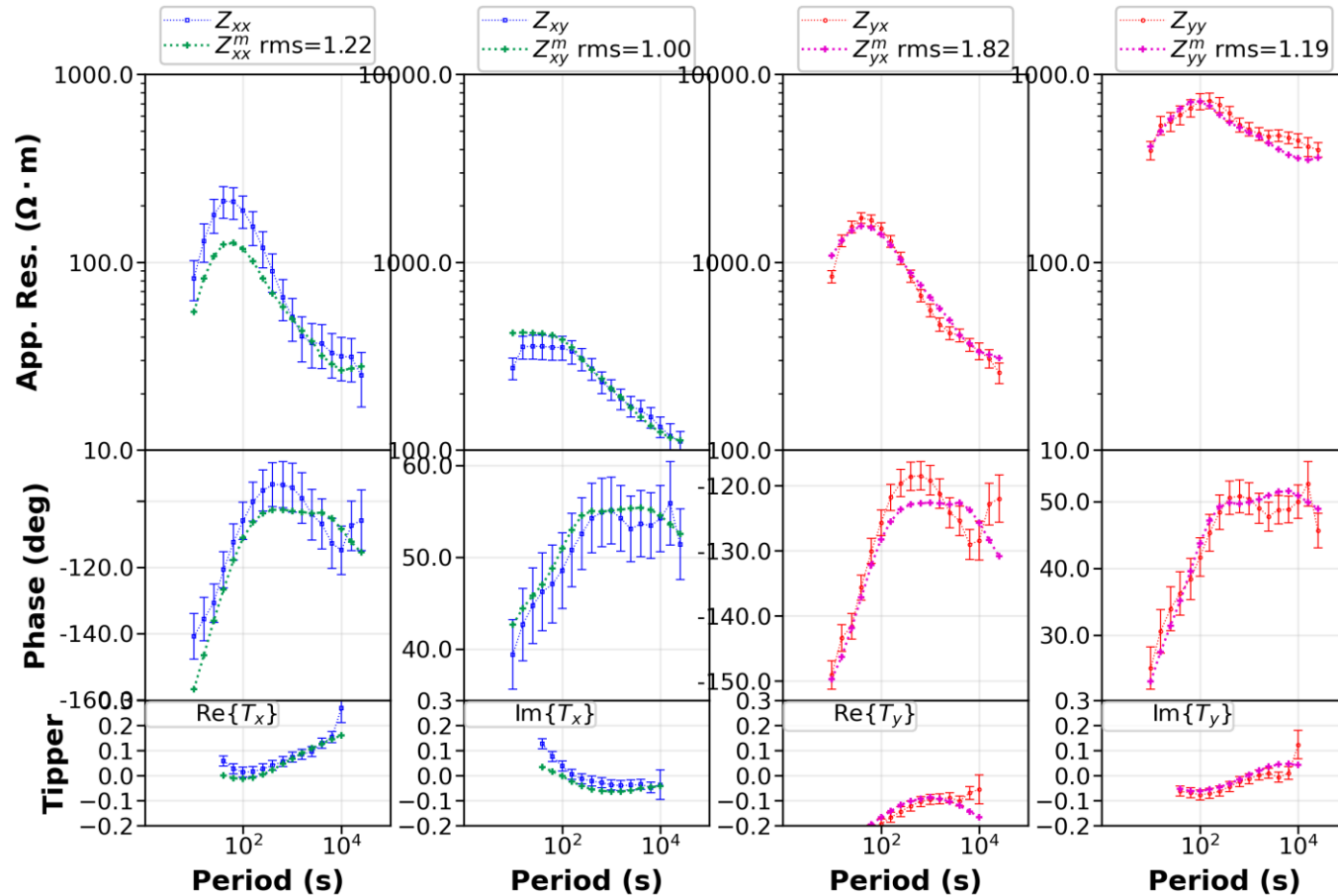
resp_fn = os.path.join(wd
data_fn = os.path.join(wd

# make a plot object
ro = PlotResponse(data_fn
                    resp_fn
                    plot_title

                    plot_style
                    fig_size
                    font_size
                    plot_zorder

                    )
ro.save_plots = True
ro.plot()

```



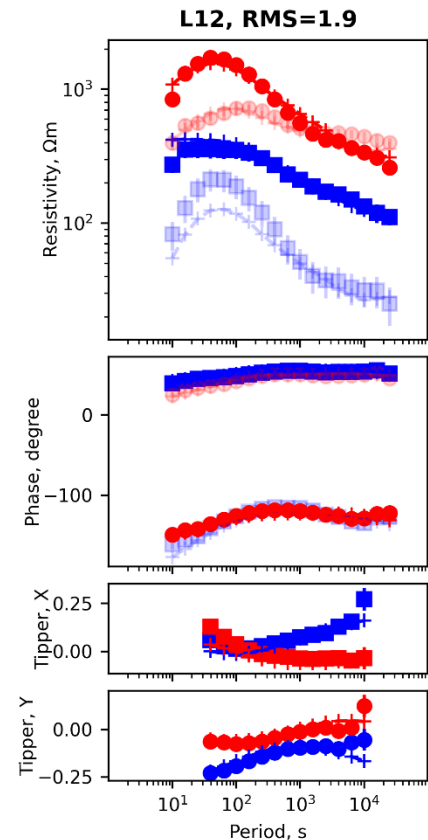
Plot Response (1 column)

```
import os
from mtpy.modeling.modem import PlotResponse

wd = r'C:\data\modemfiles'

model_fn = os.path.join(wd, 'Modular_MPI_NLCG_005.rho')
resp_fn = os.path.join(wd, 'Modular_MPI_NLCG_005.dat')
data_fn = os.path.join(wd, 'ModEM_Data.dat')

# make a plot object
ro = PlotResponse(data_fn=os.path.join(wd, data_fn),
                 resp_fn=os.path.join(wd, resp_fn),
                 plot_type=['L12'], # station names list
                               # or '1' to plot all stations
                 plot_style=3, # 1 for 4-columns; 2 for 2-columns
                 fig_size=(3,6),
                 font_size=8,
                 plot_z=False,
                 )
ro.save_plots = True
ro.plot()
```



RMS maps

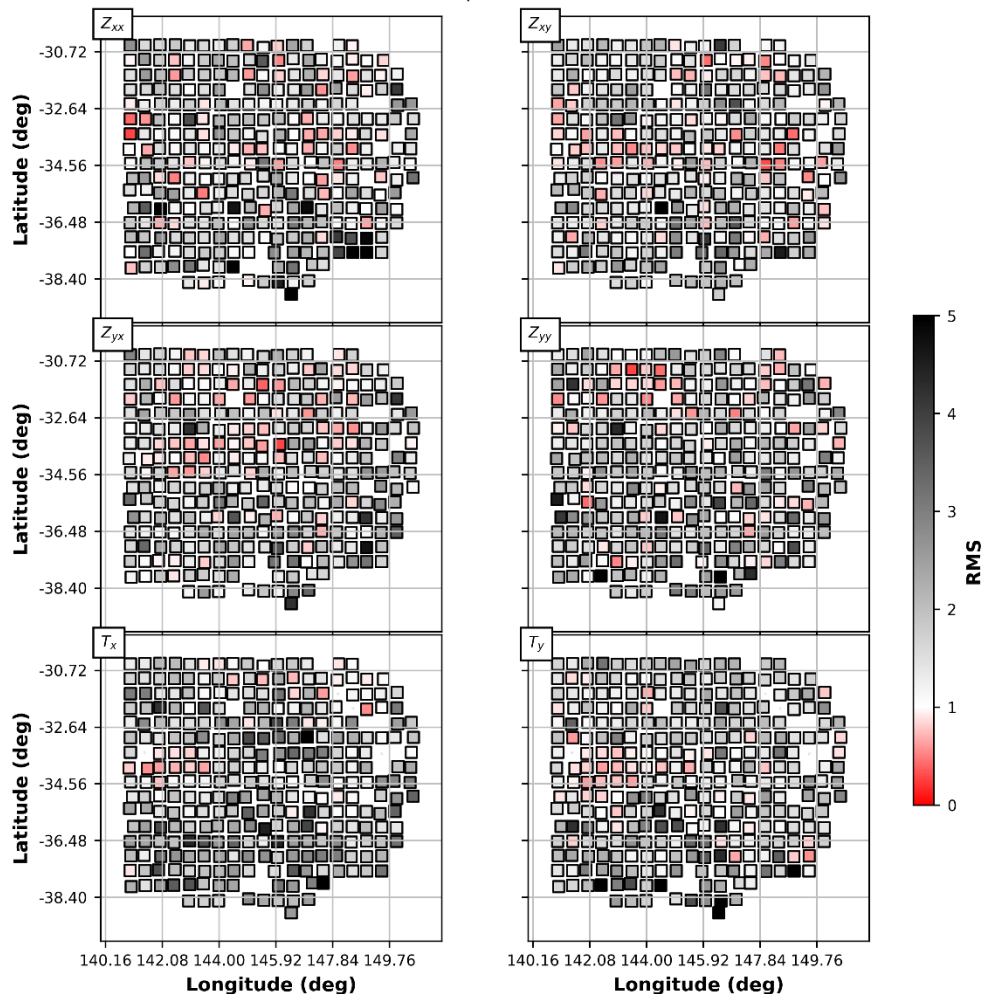
```
import os
import numpy as np
from mtpy.modeling.modem import PlotRMSMaps

wd = r'C:\data\modemfiles'
savepath = r'C:/tmp'

resid_fn = os.path.join(wd, 'Modular_MPI_NLCG')

# create a RMS plot object
projj = PlotRMSMaps(resid_fn,
                    rms_min=0, # min rms for
                    rms_max=5, # max rms for
                    period_index='all', # 'all'
                    fig_size=(7.5,8)
                    )

# save figure
projj.save_figure(save_path=savepath,
                 fig_close=False, # close figure
                 save_fig_dpi = 400 # resolution
                 )
```



Summary

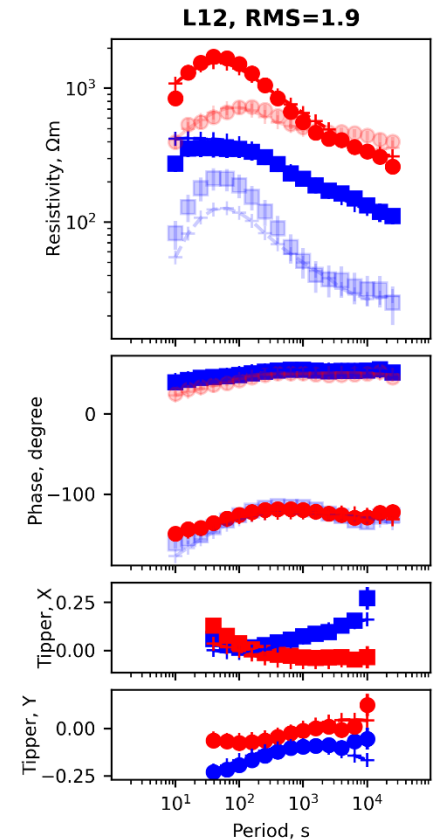
MTPy is a collection of tools to help with analysis and visualisation of MT data

Tools include:

- Reading, writing and visualisation of MT data
- Analysis tools
- Reading, writing and visualisation of commonly-used inversion inputs/outputs

Please report any bugs via issues page

v2.0 under scoping (so list any ideas in the “issues” page for v2.0)





Australian Government
Geoscience Australia



Thank you!

<https://github.com/MTgeophysics/mtpy>

EMinar, A Kirkby - MTPy, February 2021