pyCSAMT Documentation

Release v1.0.01

Kouadio K. Laurent

Mar 05, 2021

CONTENTS:

CHAPTER

ONE

PACKAGE CSAMT

1.1 Module AVG

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Wed Nov 18 16:45:35 2020

@author: KLaurent K. alias @Daniel03

class csamtpy.ff.core.avg.**Amps**(*amps_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)

Square-Wave current (amps)

amps_array [np.ndarray (ndarray, 1)] evolution of current data on the site

number_of_stations [int] number of survey_stations .

number_of_frequencies: int number of frequencies during the survey for each station location

Attributes	Type	Explanation
value	nd.array	data of Amp column on avgfile
max	float	maximum current enforce a that point
min	float	minimun current unit :amp (A)
loc	dict	main attribute location of station with the current enforce a that point .eg loc ['S07'] show the current Amps-data at that point

More attributes can be added by inputing a key word dictionary

Example

```
>>> from csamtpy.ff.core.avg import Avg
>>> path=os.path.join(os.environ["pyCSAMT"],
...       "csamtpy", "data", "K1.AVG")
>>> avg_obj=Avg(path)
>>> amps_obj =avg_obj.Data_section.Amps.loc['S00']
>>> print(amp_obj)
```

class csamtpy.ff.core.avg.**Avg**(*data_fn=None*, ***kwargs*)

A super class of all content of AVG Zonge file

Deal with Zonge Engeering Avg file .

Methods

avg_to_edifile([data_fn, profile_fn, ...])	Method to write avg file to SEG-EDIfile.Convert both files.Astatic or plainty avg file .
avg_to_jfile([avg_data_fn, station_fn, ...])	Method to write avg file to Jfile, convert both files , Astatic or plainty
avg_write_2_to_1([data_fn, savepath])	Method to rewrite avg Astatic file (F2) to main file F1 .

avg_to_edifile(*data_fn=None*, *profile_fn=None*, *savepath=None*, *utm_zone=None*, *apply_filter=None*)

Method to write avg file to SEG-EDIfile.Convert both files.Astatic or plainty avg file . if ASTATIC file is provided , will add the filter and filter values . if avg file is not astatic file , user an apply filter by setting filter to "tma, ama, or flma".Once apply , edifiles will exported by computing resistivities filtered

Parameters

*** data_fn** [str]

full path to avgfile

- **savepath** [str] outdir to store edifiles if None , is your current work directory
- **profile_fn: str** full path to station _profile file
- **apply_filter: str** add the name of filter to process the avg file exported in edifiles. can be ; [TMA | AMA| FLMA] TMA - Trimming Moving Average AMA - Adaptative Moving avarage , FLMA - Fixed dipoleLength moving average (number of point=7) Dipolelength willbe computed automatically

:Example:

```
>>> from csamtpy.pyCS.core import avg
>>> avg_obj= avg.Avg()
>>> avg_obj.avg_to_edifile(data_fn= os.path.join(path_to_avgfile,␣
↪avgfile) ,
...            profile_fn = os.path.join(path_to_avgfile, station_
↪profile_file),
...            savepath =save_edipath,
...            apply_filter=None )
```

avg_to_jfile(*avg_data_fn=None*, *station_fn=None*, *j_extension='dat'*, *utm_zone='49N'*, ***kws*)

Method to write avg file to Jfile, convert both files , Astatic or plainty avg file to A.G. Jones format.

Parameters

*** avg_data_fn:str**

pathLike , path to your avg file

- **station_fn: str** pathLike, path to your profile/station file .
- **j_extension: str** Extension type you want to export file . *Default* is ".dat"
- **utm_zone: str** add if station_profile are not referenced yet. later , it would be removed .
- **savepath: str** pathLike , path to save your outpufile
- **write_info: bool** write the informations of your input file , export informations into Jfile,*Default* is False.
- **survey_name: bool,** survey_area

avg_write_2_to_1(*data_fn=None*, *savepath=None*)
Method to rewrite avg Astatic file (F2) to main file F1 .

Parameters

- **data_fn** (*str*) – ASTATIC FILE
- **savepath** (*str*) – path to save your rewritten file .

class csamtpy.ff.core.avg.**Comp**(*comp_name=None*, ***kwargs*)
Components measured

Holds the following information:

class csamtpy.ff.core.avg.**Data**(*data_array=None*, ***kwargs*)
AVG Data informations , Container of all data infos .

class csamtpy.ff.core.avg.**Emag**(*e_mag_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)
E-field magnitude (microVolts/(kiloMeter*Amp))

class csamtpy.ff.core.avg.**Ephz**(*e_phz_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)
E-field phase (milliRadians)

class csamtpy.ff.core.avg.**Frequency**(*freq_array=None*, *normalize_freq_betw=None*, ***kwargs*)
Frequency informations - Frequency at which data was measured (Hertz). Frequency on Hz

Methods

normalize_frequency([normalize_freq_betw])	method to interpolate frequencies.

normalize_frequency(*normalize_freq_betw=None*)
method to interpolate frequencies.

Deprecated since version waist_method: will replaced on cs module.

Returns

array_like frequency normalized.

Raises

CSex raise Error if input arguments for frequency interpolating is wrong.

class csamtpy.ff.core.avg.**Header**(*header_infos=None*, ***kwargs*)
Read the info Header of AVG file and rewrite Avgfile (main type, Type1) .

Methods

write_header_log([data_fn, savepath])	Method to write your head log.

write_header_log(*data_fn=None*, *savepath=None*)
Method to write your head log. In fact just to see whether your Zonge Infos was set correctly.

Parameters

*** data_fn** [str]

path to Avgfile

- **savepath** [str] path to your destination file . if None , path is your current work directory .

:Example:

```
>>> from csamtpy.ff.core.avg.Avg import Header
>>> Header().write_header_log(data_fn=path,
...                )
>>> he=Header()
... he.write_header_log(data_fn=path,
...                     savepath=r'C:/Users\Administrator\
↪Desktop')
... avg_obj.Header.write_header_log(data_fn=path,
...                 savepath=r'C:/Users/Administrator/Desktop')
```

class csamtpy.ff.core.avg.**Hmag**(*h_mag_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)
H-field magnitude (picoTesla/Amp) (milliGammas/Amp)

class csamtpy.ff.core.avg.**Hphz**(*h_phz_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)
H-field phase (milliRadians)

class csamtpy.ff.core.avg.**Phase**(*phase_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)
Impedance phase on milliRadians can be calculate using Ephz object and Hphz object

- **Phase** = E-phase - H-phase

class csamtpy.ff.core.avg.**ReceiverProperties**(*Rx_data=None*, ***kwargs*)
Class for receiver properties.

Methods

set_receiver_properties([Rx_data])	Methods to set Receivers - properties infos from AVG files.

set_receiver_properties(*Rx_data=None*)
Methods to set Receivers - properties infos from AVG files.

Parameters

*** Rx_data** [list, optional] container infos of receivers. The default is None.

class csamtpy.ff.core.avg.**Resistivity**(*res_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)
based on Cagniard Resistivity (Ohm-Meters) calculation

class csamtpy.ff.core.avg.**Skip_flag**(*skip_flag=None*, ***kwargs*)
skip flag values

- 0. drop data
- 1. skip data
- 2. good quality of data

Methods

setandget_skip_flag([skip_flag])	simple method to set and get skip_flag.

setandget_skip_flag(*skip_flag=None*)
simple method to set and get skip_flag.

Parameters

*** skip_flag: str**

class csamtpy.ff.core.avg.**Station**(*station_data_array=None*, ***kwargs*)
Stations informations

class csamtpy.ff.core.avg.**SurveyAnnotation**(*survey_annotations_data=None*, ***kwargs*)
Class for survey annotations.

Methods

set_survey_annotations_infos([...])	Method to set _survey annotations informations .

set_survey_annotations_infos(*survey_annotations_data=None*)
Method to set _survey annotations informations .

Parameters

*** survey_annotation** [list] container of survey annotations infos.

class csamtpy.ff.core.avg.**SurveyConfiguration**(*survey_config_data=None*, ***kwargs*)
Class for survey Survey configuration.

Methods

set_survey_configuration_infos([...])	Method to set _survey configurations informations .

set_survey_configuration_infos(*survey_config_data=None*)
Method to set _survey configurations informations .

Parameters

* **survey_config_data** [list or pathLike str] container of survey configurations infos.

class csamtpy.ff.core.avg.**TransmitterProperties**(***kwargs*)
Class for transmitter properties.

Methods

set_transmitter_properties([Tx_data])	Method to set_Tx_properties.

set_transmitter_properties(*Tx_data=None*)
Method to set_Tx_properties.

Parameters **tx_data** – list of Tx-infos from AVG filename

:type tx_data:list

class csamtpy.ff.core.avg.**Z_Tensor**(*z_array=None*, *phase_array=None*, *freq=None*, *z_error=None*, ***kwargs*)

Impedance Tensor Z Calculation : class can recompute the apparent resistity rho base on impedance Tensor Z.

See also:

Zonge, K.L. and Hughes, L.J., 1991, Controlled source audio-frequency magnetotellurics,in Electromagnetic Methods in Applied Geophysics, ed. Nabighian, M.N., Vol. 2,Society of Exploration Geophysicists, pp. 713-809.

Attributes

freq

phase

rho

z

z_error

Methods

rhophi2rhoph_errors([res_array, ...])	compute the phase and resistivities error via res_array , phase_array and _z error.
z_and_zerr_2rhophi([z_array, freq])	Method to compute resistivity and phase phase using Z_values and Zerror _values

rhophi2rhoph_errors(*res_array=None*, *phase_array=None*, *z_error=None*, *freq=None*)
compute the phase and resistivities error via res_array , phase_array and _z error.

Parameters

*** res_array** [ndarray ,]

resistivity value in ohm.m

- **phase_array** [ndarray ,] phase angle value in mradians
- **z_error** [ndarray ,] impedance Tensor error
- **freq** [ndarray ,] frequency numbers in Hz

Returns

———

ndarrays resistivities error and phase errors values .

z_and_zerr_2rhophi(*z_array=None*, *freq=None*)
Method to compute resistivity and phase phase using Z_values and Zerror _values

Parameters

*** z_array** [complex]

Impedance tensor complex_number The *default* is None.

- **freq** [ndarray,] frequency value. The *default* is None.

Returns

ndarray(ndarray,1)

resistivity value computed in ohm.m

ndarray (ndarray, 1), phase angle value in degree.

class csamtpy.ff.core.avg.**ZongeHardware**(*zonge_hardw_infos=None*, ***kwargs*)
Some features of zonge AMTAVG software.

Methods

set_harware_infos([zonge_hardw_infos])	Method to set Harwares infos.

set_harware_infos(*zonge_hardw_infos=None*)
Method to set Harwares infos.

Parameters

*** zonge_hardw_infos** [list] Hardware informations collected

class csamtpy.ff.core.avg.**pcEmag**(*pc_e_mag_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)
Statistical variation of magnitude values from averaged data blocks.

Standard Deviation/Average Emag (percent)

class csamtpy.ff.core.avg.**pcHmag**(*pc_h_mag_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)

Statistical variation of magnitude values from averaged data blocks.

Standard Deviation / Average Hmag (percent)

pcHmag [ndarray] data array of statistical variataion of Hmag value on the field.

More attributes can be added by inputing a key word dictionary

Example

```
>>> from csamtpy.ff.core.avg import Avg
>>> path=os.path.join(os.environ["pyCSAMT"],
...       "csamtpy", "data", "K1.AVG")
>>> avg_obj=Avg(path)
>>> pchmag_obj =avg_obj.Data_section.pcHmag.loc['S05']
>>> print(pchmag_obj)
```

class csamtpy.ff.core.avg.**pcRho**(*pcRes_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)
Statistical variation of magnitude values from averaged data blocks. Standard Deviation / Average Rho (percent)

class csamtpy.ff.core.avg.**sEphz**(*sEphz_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)
Statistical variation of the data blocks averaged for this data point. 100 * Standard Deviation of Ephz values (milliradians)

class csamtpy.ff.core.avg.**sHphz**(*shphz_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)

Statistical variation of the data blocks averaged for this data point. 100 * Standard Deviation of Hphz values (milliradians)

class csamtpy.ff.core.avg.**sPhz**(*sPhase_array=None*, *number_of_frequencies=None*, *number_of_stations=None*, ***kwargs*)

Statistical variation of the data blocks averaged for this data point. 100 * Standard Deviation of Phase values (milliradians)

1.2 Module CS

Created on Wed Dec 2 11:29:32 2020

@author: KouaoLaurent alias @Daniel03

class csamtpy.ff.core.cs.**CSAMT**(*data_fn=None*, *profile_fn=None*, ***kwargs*)

CSAMT class is super class container of all the information of the other classes,

J, Avg and Edi. In fact, the CS object collect all the information of the other classes, once questioned for specific uses. The purpose of the construction of this object to avoid the repetition of scripts throughout the project. Objet CSAMT can recognize the typycal input obj of file and set attributes for use .

Attributes

dipolelength

doi

east

elev

fpath assert path and redirect to file either single file ,edifiles or Jfiles.

freq

lat

lon

north

phase

phase_err

resistivity

resistivity_err

skindepth

station

station_distance

station_separation

z_err

zxy

zyx

Methods

find_path([path, ptol])	Check path and return filepath , edipath or jpath .

static find_path(*path=None*, *ptol=0.7*)
Check path and return filepath , edipath or jpath .

Parameters

- **path** (*str or pathlike*) – full path to file or directory
- **ptol** (*float*) – tolerence given by the program , less or egal to 1

Returns specific path

Return type str

Note: tolerence param inspects the number of EDI or J file located on the path and determine the typical path of files either edipath or jpath.

property fpath
assert path and redirect to file either single file ,edifiles or Jfiles. find the specific path [EDI|J] path.

class csamtpy.ff.core.cs.**Location**(***kwargs*)
Details of sation location . Classe used to convert cordinnates and check values for lat/lon , east/north

Attributes	Type	Description
latitude	float/ndarray,1	sation latitude
longitude	float/ndarray,1	station longitude
elevation	float/ndarray	station elevantion in m or ft
easting	float/ndarray.1	station easting coordinate (m)
northing	float/ndarray,1	station northing coordinate (m)
azimuth	float/ndarray,1	station azimuth in meter
stn_pos	ndarray,1	sation dipoleposition
utm_zone	str	UTM location zone

Methods	Description
convert_location_2_utm	convert position location lon/lat in utm easting northing
convert_location_2_latlon	convert location postion from east/north to latitude/longitude

Attributes

azimuth

easting

elevation

latitude

longitude

northing

stn_pos

utm_zone

Methods

convert_location_2_latlon([utm_zone])	Project coodinate on longitude latitude once data are utm at given reference ellispoid constrained to WGS-84.
convert_location_2_utm([latitude, longitude])	Project coordinates to utm if coordinates are in degrees at given reference ellipsoid constrained to WGS84.
get_eastnorth_array_from_latlon(arr_lat, arr_lon)	Method to quicly convert array of latitude and northing into easting northing

convert_location_2_latlon(*utm_zone=None*)
Project coodinate on longitude latitude once data are utm at given reference ellispoid constrained to WGS-84.

convert_location_2_utm(*latitude=None*, *longitude=None*)
Project coordinates to utm if coordinates are in degrees at given reference ellipsoid constrained to WGS84.

Parameters

- **latitude** (*float*) – latitude number
- **longitude** (*float*) – longitude number

get_eastnorth_array_from_latlon(*arr_lat*, *arr_lon*)
Method to quicly convert array of latitude and northing into easting northing

Parameters

- **arr_lat** (*array_like*) – array of latitude value
- **array_lon** (*array_like*) – array of longitude value.

Returns easting array

:rtype : array_like

Returns northing array

Return type array_like

class csamtpy.ff.core.cs.**Profile**(*profile_fn=None*, ***kwargs*)
Profile class deal with AVG Zonge station file and statation locations coordinates could be find in *.stn* file or SEG-EDI file.

Parameters **profile_fn** (*str*) – Path to Zonge *STN file of SEG-EDI locations or Zonge station file

Note: When EDI file is called , EDI-collecton auto populated profile attributes and coordinates are automatically rescalled.

Attributes	Type	Explanation
Location	class	Location class for Easting Northing azimuth details.
profile_angle	float	If user doesnt Know the angle profile . He can use the method "get_profile_angle" to get the value of profile angle.
stn_interval	(ndarray,1)	Array of station separation.
dipole_length	float	Dipole length value computed automatically.
lat/lon	(ndarray,1)	latitude/longitude of stations points .
east/north	(ndarray,1)	Easting and northing of stations points.
azimuth	(ndarray,1)	Azimuth array stations .
ele	(ndarray,1)	Elevation array at each station points.
stn_position	(ndarray,1)	Station position occupied at each stations.

More attributes can be added by inputing a key word dictionary

Example

```
>>> from csamtpy.ff.core.cs import Profile
>>> file_stn = 'K1.stn'
>>> path =  os.path.join(os.environ["pyCSAMT"],
...                      'csamtpy','data', file_stn)
>>> profile =Profile (profile_fn=path)
>>> profile.straighten_profileline(
    X=profile.east, Y=profile.north,straight_type='n')
>>> profile.rewrite_station_profile(
...            easting=profile.east,
...            northing=profile.north,
...            elevation =profile.elev,
...            add_azimuth=True)
>>> separation = profile.stn_separation(
...    easting = profile.east,
...    northing =profile.north)
```

Attributes

east

elev

lat

lon

north

stn_position

Methods

compute_dipolelength_from_coords([easting, ...])	Fonction to compute dipole length from coordinates easting and northing values.
get_profile_angle([easting, northing])	Method to compute profile angle .
read_stnprofile([profile_fn, easting, ...])	Method to read profile station file.
reajust_coordinates_values([x, y, stn_fn, ...])	Simple staticmethod to readjut coordinates values and write new station file.
rewrite_station_profile([easting, northing, ...])	Mthod to rewrite station_profile or output new profile by straightening profile throught reajusting location coordinates values. User can use this method to create zonge *stn* file if coordinates are known.
stn_separation([easting, northing, interpolate])	Compute the station separation Distance between every two stations
straighten_profileline([X, Y, ...])	Method to straighten profile line and/or rescaled coordinates.

static compute_dipolelength_from_coords(*easting=None*, *northing=None*, ***kwargs*)

Fonction to compute dipole length from coordinates easting and northing values.

Parameters

- **easting** (*array_like*) – array of easting coordinate in meters
- **northing** (*array_like*) – array of northing coordinate in meters
- **lat** (*array_like (ndarray,1)*) – latitude coordinate in degree
- **lon** (*array_like (ndarray,1)*) – longitude coordinate in degree
- **reference_ellipsoid** (*int*) – id, Ellipsoid name, Equatorial Radius, square of eccentricity ,default is 23

Returns length of dipole during survey approximated .

Return type float

Returns position of dipole from reference station

Return type array_like(ndarray,1)

Note: the first electrode is located at 0 and second electrode to dipole length i.e [0, 50 , . . . , nn*50] where nn number of point -1. Data are relocated in center position of dipole.

get_profile_angle(*easting=None*, *northing=None*)

Method to compute profile angle .

Parameters

- **easting** (*array_like*) – easting corrdinate of the station point
- **northing** – northing coordinate of station point.

:type northing:array-like

Returns profile angle in degrees.

Return type float

read_stnprofile(*profile_fn=None*, *easting=None*, *northing=None*, *elevation=None*, *split_type=None*, ***kwargs*)

Method to read profile station file. user can use its special file .user can specify a head of its file. method will read and will parse easting , northing , elevation , or lon, lat, elev or station . User can also provided easting , northing and elevation value .

Parameters

*** profile_fn :str**

path to station profile file

- **split_type :str** How data is separed . Default is "".
- **easting** [array_like] easting coordinate (m),
- **northing** [array_like] northing coordinate value (m)
- **lat** [array_like] latitude coordinate in degree
- **lon** [array_like] longitude coordinate in degree
- **azim** [array_like ,] azimuth in degree If not provided can computed automatically
- **utm_zone :str** survey utm zone if not porvided and lat and lon is set , can compute automatically

static reajust_coordinates_values(*x=None*, *y=None*, *stn_fn=None*, *rewrite=False*, *savepath=None*, ***kwargs*)

Simple staticmethod to readjut coordinates values and write new station file. by default , the reajustment substract value. to add value to you old coordiantes , use negative X and Y method offer possibility of output new file by setting write to True. By convention we use X as EASTING correction and Y for NORTHING correction.

Parameters

*** x: float**

value for ajusting X coordinates _EASTING

- **y: float** value for ajustig Y coordinates ._NORTHING
- **stn_file: str** station profile file . it may be a STN file .
- **rewrite: bool** rewrite a new station file after reajust coordinates.
- **savepath** [str] outdir pathLike to save your new profile file.

Returns

- **array_like**

stations_pk , station profile pka value(m) . Electrode fixed point value.

  - **array_like** easting coordinate value (m)
  - **array_like** northing coordinate value (m)
  - **elevation** [array_like] evelation point at each station (m)

:Example :

```
>>> from csamtpy.ff.core.cs import Profile
>>> stn_file =K1.stn
>>> path =  os.path.join(os.environ["pyCSAMT"],
...                          'csamtpy','data',
...                          stn_file)
>>> profile =Profile.reajust_coordinates_values(
...           x=-300238.702 ,y=-2369.252  )
```

rewrite_station_profile(*easting=None*, *northing=None*, *elevation=None*, *area_name=None*, *username=None*, *add_azimuth=False*, ***kwargs*)

Mthod to rewrite station_profile or output new profile by straightening profile throught reajusting location coordinates values.

User can use this method to create zonge *stn* file if coordinates are known.

Parameters

- **easting** (*array_like*) – easting coordinates (m)
- **northing** (*array_like*) – northing coordinates (m)
- **elevation** (*array_like*) – elevation values (m)
- **username** (*str*) – name of user
- **add_azimuth** (*bool*) – compute azimuth positive down(clockwise)

stn_separation(*easting=None*, *northing=None*, *interpolate=False*)

Compute the station separation Distance between every two stations

Parameters

*** easting** [array_like (ndarray, 1)]

easting coordinates

- **northing** [array_like (ndarray,1)] northing coordinates
- **interpolate** [bool] if interpolate is True will extend to N+1 number to much excatly the number of electrode. If false , it match the number of dipole N. *Default* is False.

Returns

array_like

separation value array

float separation mean or average separation value

straighten_profileline(*X=None*, *Y=None*, *straight_type='classic'*, *reajust=(0, 0)*, *output=False*)

Method to straighten profile line and/or rescaled coordinates. User can readjust coordinateq of profile by adding coordinate of readjustation Method provides 3 type of straighten profile. Default is “classic”, it could be “‘natural or distorded’, equidistant” . “natural or distorded Type” is not to straight a profile like a straight line but , it keeps the equidistant point of the station at normal place that the survey must be. sometimes on the field , crew may get around some obstacle and despite the line is not straight , the distance between station is distorded. Using ‘distord or natural type ‘ , it will show the right place station must be.

Note: for easier approch we use X as easting and Y as northing.

Parameters

- **X** (*array_like (ndarry, array,1)*) – easting coordinates array.
- **Y** (*array_like (ndarray,1)*) – northing coordinates array
- **straight_type** (*str*) – type of straighten ,it could be "equistant or egal, natural or distord". *default* is "classic"
- **reajust** (*tuple*) – coordinates for reajustment (index 0 :x index 1 : y)

class csamtpy.ff.core.cs.**Site**(*data_fn=None*, ***kwargs*)

Specific site object Easy pack data :lat, lon, elev, azim, east, north, into dictionnary for easy access .

Attributes

azim

east

elev

lat

lon

north

stn_name

Methods

set_site_info([data_fn, easting, northing, ...])	Set-info from site file, can read zonge *stn* profile fine or set easting and northing coordinates.

set_site_info(*data_fn=None*, *easting=None*, *northing=None*, *utm_zone='49N'*)

Set-info from site file, can read zonge *stn* profile fine or set easting and northing coordinates.

Parameters

- **data_fn** (*str*) – path to site data file . may Use Stn or other file of coordinates infos
- **utm_zone** (*str*) – Utm zone WGS84

csamtpy.ff.core.cs.**round_dipole_length**(*value*)

small function to graduate dipole length 5 to 5. Goes to be reality and simple computation

1.3 Module EDI

Created on Mon Jan 11 11:37:51 2021

@author: KouaoLaurent alias @Daniel03

class csamtpy.ff.core.edi.**Copyright**(***kwargs*)

Information of copyright, mainly about how someone else can use these data. Be sure to read over the conditions_of_use.

Holds the following informations:

Attributes	Type	Explanation
references	References	citation of published work using these data
conditions_of_use	string	conditions of use of data used for testing program
release_status	string	release status [open | public |proprietary]

More attributes can be added by inputing a key word dictionary

Example

```
>>> from csamtpy.ff.core.edi import Copyright
>>> Copyright(**{'owner':'University of CSAMT',
...                  'contact':'Cagniard'})
```

class csamtpy.ff.core.edi.**DefineMeasurement**(*defineMeas_list=None*, ***kwargs*)

Begins Measurement definition data section . Defines Location of sensors and parameters pertainning to runs for each measurments .

Attributes

refelev

reflat

reflong

Methods

get_DefineMeasurement_info([edi_fn])	Class method to get definemeasurement list.
read_define_measurement([...])	readmeasurement inedilist and populate attributes .
write_define_measurement([...])	Write definemeasurement method ,intend to write and rewrite measurements infos into list. informations must be on list as possible if not may set attribute manually i.e [key01=value02 , ..., keynn=valuenn + dictXX]dictXX ={meas_e}, {meas_hx}, {meas_ey}{meas_hx}, {meas_hy} {meas_hz}.

classmethod get_DefineMeasurement_info(*edi_fn=None*)

Class method to get definemeasurement list.

Parameters **edi_fn** (*str*) – full path to edifiles.

Returns new class with infos list

Return type list

read_define_measurement(*define_measurement_list=None*)

readmeasurement inedilist and populate attributes .

Parameters **define_measurement_list** (*list*) – list of measurement data can be [key_01=value_01 , ..., key_xx = value_xx] Emeas and Hmeas will be set on dictionnary and call the class to populate attribute *default* is None

Example

```
>>> from csamtpy.ff.core.edi import DefineMeasurement
>>> file_edi= 'S00_ss.edi'
>>> path =  os.path.join(os.environ["pyCSAMT"],
...                          'csamtpy','data', file_edi_2)
>>> definemeas =DefineMeasurement.get_measurement_info(edi_
↪fn=path)
>>> print(definemeas.define_measurement)
>>> print(definemeas.meas_ex)
```

Note: to get measurement_hx or measurement_ex for instance get attribute <id> of Emeasurement , do the following script

Example

```
>>> from csamtpy.ff.core.edi import DefineMeasurement
>>> definemeas =DefineMeasurement.get_measurement_info(edi_
↪fn=path)
>>> print(definemeas.meas_ex.id)
... 1004.
```

write_define_measurement(*define_measurement_list=None*)

Write definemeasurement method ,intend to write and rewrite measurements infos into list. informations must be on list as possible if not may set attribute manually

i.e [key01=value02 , ..., keynn=valuenn + dictXX]dictXX ={meas_e}, {meas_hx}, {meas_ey}{meas_hx}, {meas_hy} {meas_hz}

Parameters **define_measurement_list** (*list*) – list of define measuremenent

Returns new list of define_measurement

Return type list

. .notes :: If no edifiles is provided , can write definemeasurement by creating dict of Eand H measurment.

Example

```
>>> ex_dict ={'id':1002, 'chtype':'Ex', 'x':0, 'y':0, 'z':0, 'x2
↪':-50,'y2':0, 'z2':0,
...                 'acqchan':0,'filter':'hanning', 'sensor':
↪'ex', 'gain':None}
>>> ey_dict ={'id':1003.1, 'chtype':'Ey', 'x':0, 'y':50, 'z':0,
↪'x2':0,'y2':0, 'z2':0,
...              'acqchan':0}
>>> hy_dict ={'id':1003, 'chtype':'Hy', 'x':0, 'y':50, 'z':90,
↪'azm':0.0,'dip':35,
...                 'acqchan':'hy','filter':'Hanning', 'sensor
↪':None, 'gain':0, 'measdate':''}
>>> definemeas =DefineMeasurement()
>>> ex =Emeasurement(**ex_dict)
>>> ey =Emeasurement(**ey_dict)
>>> hy=Hmeasurement(**hy_dict)
... definemeas.__setattr__('meas_ex', ex)
... definemeas.__setattr__('meas_ey', ey)
... definemeas.__setattr__('meas_hy', hy)
>>> print(definemeas.write_define_measurement())
```

class csamtpy.ff.core.edi.**Edi**(*edi_filename=None*, ***kwargs*)

Ediclass is for especialy dedicated to .edi files, mainly reading and writing which are meant to follow the archaic EDI format put forward by SEG Can read impedance, Tipper but not spectra. To read spectra format please consult MTpy documentation https://mtpy2.readthedocs.io/en/develop/ The Edi class contains a class for each major section of the .edi file.

Note: Frequency and components are ordered from highest to lowest frequency.

Attributes

elev

lat

lon

processingsoftware

station

Methods

read_edi([edifile])	Read edifile and populate attribute to each data section of edi.
write_edifile([edi_fn, new_edifilename, ...])	Method to write edifiles from data setting oin attribute of Edi or from existing file.

read_edi(*edifile=None*)

Read edifile and populate attribute to each data section of edi.

Parameters **edifile** (*str*) – full path to edifile

write_edifile(*edi_fn=None*, *new_edifilename=None*, *datatype=None*, *savepath=None*, *add_filter_array=None*)

Method to write edifiles from data setting oin attribute of Edi or from existing file. can write also EMAP data are filled attribute of EDI.

Parameters

- **edifile** (*str*) – new edifile name .If None , will write edi using station_name plus type of survey (MT of EMAP) plus year of writing as < S00_emap.2021.edi> or <S00_mt.2021.edi>
- **datatype** (*str*) – type of file , "mt" or "emap" if None , program will detect which file is provided . If datatype is set , program will be force to rewrite edi into given format.
- **savepath** (*str*) – path to save edifile. if None save to your current work directory
- **add_filter_array** – ndarray(nfreq, 2, 2), EDI edifile is EMAP section data , if add filter is provided , will recompute rho.

Returns new_edifile , full path to edifile

Return type str

class csamtpy.ff.core.edi.**Edi_collection**(*list_of_edifiles=None*, *edipath=None*, *survey_name=None*)

Super class to deal with Edifiles .Collect edifiles and set important properties form Controled Source audiofrequency magnetotelluRic ,two(2) components XY and YX will be set and calculated .

Attributes

elevation

latitude

longitude

phs_err_xy

phs_err_yx

phs_xy

phs_yx

res_err_xy

res_err_yx

res_xy

res_yx

stnames

z_err_xy

z_err_yx

z_xy

z_yx

class csamtpy.ff.core.edi.**Emeasurement**(***kws*)

Define the electrode location , and run parameters for electric field measurement. EMeasurement contains metadata for an electric field measurement.

Attributes	Description (Restriction)
id	Measurement ID , channel number (‘required ‘)
chtype	Type of Hmeasurement[Ex | Ey](required)
x	x (m) north offset from first electrode (reauired)
y	y (m) offest from ref for first electrode(reauired)
z	z offset from the ref for first electrode(reauired)
x2	x offset from the 2nd electrode (‘required’)
y2	y offset from the 2nd electrode (required)
z2	z offset from the 2nd electrode (“”)
acqchan	name of the channel acquired usually same as chtype
filter	description of sensor to run (“”)
gain	gain used for run (“”)
measdate	date of run (“”)

To Fill Metadata , let take this example

Example

```
>>> import csamtpy.ff.core.edi as csedi
>>> emeas_dict = {'id': '1000.4', 'chtype':'ex', 'x':0,
...                   'y':0, 'azm':0, 'acqchan':'ex', }
>>> emeas = csedi.Hmeasurement(**emeas_dict)
... print(emeas.chtype)
... print(emeas.azm)
... print(emeas.measdate)
```

class csamtpy.ff.core.edi.**Head**(*edi_header_list=None*, ***kwargs*)

The edi head block contains a series of options which (1) identity the data set, (2) describe whn , where and by whoom was acquired , and (3) describe when , how and by whom it was written.

Attributes

elev

lat

long

Methods

get_header_list_from_edi([edi_fn])	Class method to return edi_head_list .
read_head([edi_header_list])	read_header_list and set attributes values
write_head_info([head_list_infos])	Write list info .

classmethod get_header_list_from_edi(*edi_fn=None*)

Class method to return edi_head_list .

Paramedi_fn full path to edifile

read_head(*edi_header_list=None*)

read_header_list and set attributes values

Parameters **edi_header_list** (*list*) – list of edifile header infos

Example

```
>>> from csamtpy.ff.core.edi import Head
>>>> file_edi= 'S00_ss.edi'
>>>> path =  os.path.join(os.environ["pyCSAMT"],
...                       'csamtpy','data', file_edi)
>>> edihead= Head.get_header_list_from_edi(edi_fn=path)
>>> print(edihead.lat)
>>> print(edihead.long)
>>> print(edihead.elev)
>>> print(edihead.acqby)
```

write_head_info(*head_list_infos=None*)

Write list info . Can read edi and rewrite list or to provide input as ['key_01=value_01', 'key_02=value_02', . . . ,'key_nn=value_nn']

Note: If value is None , don't need to write the key .

Parameters **head_list_infos** (*list*) – list , list of head info

Returns write_info list , list ready to write to let EDI file more visible .

Return type list

Example

```
>>> from csamtpy.ff.core.edi import Head
>>> path =  os.path.join(os.environ["pyCSAMT"],
...                       'csamtpy','data', S00_ss.edi)
>>> edihead= Head.get_header_list_from_edi(edi_fn=path)
>>> print(edihead.write_head_info(head_list_infos = edihead.edi_
↪header))
```

class csamtpy.ff.core.edi.**Hmeasurement**(***kws*)

Define the sensor location and orientation , and run parameters for a magnetic field measurement.HMeasurement contains metadata for a magnetic field measurement.

Attributes	Description
id	Measurement ID , channel number
chtype	Type of Hmeasurment[HX | HY | HZ | RHX | RHY]
x	x (m) north offset from reference sensor
y	y (m) offest from ref sensor
azm	angle of sensor relative to north = 0
acqchan	name of the channel acquired usually same as chtype
dip	dip angle for sensor (“”)
filter	description of sensor to run (“”)
gain	gain used for run (“”)
measdate	date of run (“”)

To fill Metadata, let get a look of this example

Example

```
>>> import csamtpy.ff.core.edi as csedi
>>> hmeas_dict = {'id': '1000.3', 'chtype':'hx', 'x':0,
...                   'y':0, 'azm':0, 'sensor':'None'}
>>> hmeas = csedi.Hmeasurement(**hmeas_dict )
>>> print(hmeas.chtype)
>>> print(hmeas.azm)
>>> print(hmeas.measdate)
>>> print(hmeas.sensor)
```

class csamtpy.ff.core.edi.**Info**(*edi_info_list=None*, ***kwargs*)

Class EDI info class , collect information of the survey.

> INFO

- MAXINFO=999
- PROJECT=SAMTEX
- SURVEY=Kaapvaal 2003
- YEAR=2003
- PROCESSEDBY=SAMTEX team
- PROCESSINGSOFTWARE=JONES 2.3
- PROCESSINGTAG=
- SITENAME=South Africa
- RUNLIST=
- REMOTEREF=
- REMOTESITE=
- SIGNCONVENTION=exp(+ iomega t)

Attributes	Type	Explanation
maxrun	int>=1	maximum number of text lines in info text(maybe less)
Source	class obj	Porvenace of data to rewrite
Processing	Processing obj	How data where processed
Notes	Note class	info additions

Methods

get_info_list_from_edi([edi_fn])	Class to get edinfo from edifiles
read_info([edi_info_list])	readinformation and populate attaribute info can set other attributes once read and not present on the file.
write_edi_info([edi_info_list])	Write edi information info .

classmethod get_info_list_from_edi(*edi_fn=None*)
Class to get edinfo from edifiles

Parameters **edi_fn** (*str*) – full path to edifile

Returns edi_info_list

Return type list

read_info(*edi_info_list=None*)
readinformation and populate attaribute info can set other attributes once read and not present on the file.

Parameters **edi_info_list** (*list*) – list of infos files

write_edi_info(*edi_info_list=None*)
Write edi information info . Can read edi and rewrite list or to provide input as ['key_01=value_01', 'key_02=value_02', . . . ,'key_nn=value_nn'] Note : If value is absent i.e None , don't write the key . Info write method add some field notes informations from other softwares if exists.

Parameters **edi_info_list** (*list*) – list of infos contain in info sections

Returns list of info

Return type list

Example

```
>>> from csamtpy.ff.core.edi import Info
>>> file_edi_2='SAMTEX.edi_2.edi'
>>> file_edi= 'S00_ss.edi'
>>> path =  os.path.join(os.environ["pyCSAMT"],  'csamtpy','data
↪', file_edi_2)
>>> info_obj =Info.get_info_list_from_edi(edi_fn=path)
>>> print(info_obj.write_edi_info())
```

class csamtpy.ff.core.edi.**MTEMAP**(*mt_or_emap_section_list=None*, ***kwargs*)
Begins an MT and EMAP data section .Defines the default measurement for MT sounding and Defines the measurments which makeup an EMAP lines.

>=EMAPSECT	>=EMAPSECT
• SECTID=””	• SECTID=S00
• NFREQ=**	• NCHAN=4
• HX= 1001.001	• MAXBLKS=999
• HY= 1002.001	• NDIPOLE =47
• HZ= 1003.001	• NFREQ=17
• EX= 1004.001	• HX= 0.
• EY= 1005.001	• HY= .0 • HZ= NONE • CHKSUM =None

Parameters **mt_or_emap_section_list** (*list*) – mt and emap section can read ediflies by calling class method <’get_mtemap_section_list’>

Attributes	Description	Default	Restriction
sectid	Name of this section	None	str or””
nfreq	Number of frequencies	required	int >=1
maxblks	maximum number of blocks of this section	None	int>=1
ndipole	Number of dipoles in the EMAP line	required	
type	Descrip. of spatial filter type used	None	str or “”
hx	Meas ID for Hx measurement	None	Def Meas Id or “
hy	Meas ID for Hy measurement	None	Def Meas Id or “
hz	Meas ID for Hz measurement	None	Def Meas Id or “
ex	Meas ID for Ex measurement	None	Def Meas Id or “
ey	Meas ID for Ey measurement	None	Def Meas Id or “
rx	Meas ID for Rx measurement	None	Def Meas Id or “
ry	Meas ID for Ry measurement	None	Def Meas Id or “
chksum	checksum total for dvalues	None	Num of “”

Note: MTEMAP can recognize function to value provided with type of data acquired either MT or EMAP. More attributes can be added by inputing a key word dictionary. . . .

Example

```
>>> MTEMAP(**{'ex':'0011.001', 'hx':'0013.001','sectid':'Niable',
↪'nfreq':47,
...              'ey':'0012.001', 'hy':0014.001 , 'hz': 0015.001 })
>>> MTEMAP (**{'sectid':'yobkro','nfreq':18 , 'maxblks':100, 'hx':
↪"1003.1",
...            'ex':'1005.4','hz':'1006.3', 'ey':1002.3 , 'hy':'1000.
↪2','chksum':47,
```

(continues on next page)

(continued from previous page)

```
...             'ndipole':47, 'type':'hann'})
```

Attributes

start_data_lines_num

temp_sectid

Methods

get_mtemap_section_list([edi_fn])	MT or EMAP section_classmethod to get special MT info or EMAP info in edifile
read_mtemap_section([mt_or_emap_section_list])	Read mtsection and set attribute . values can be set as key
write_mtemap_section([...])	Method to write MT or EMAP section into list by providing list as ['key01= value01', ..., keyxx=valuexx]. Method can recognize whether edifile provided is MT or EMAP then can read file according to.

classmethod get_mtemap_section_list(*edi_fn=None*)

MT or EMAP section_classmethod to get special MT info or EMAP info in edifile

Parameters **edi_fn** (*str*) – full path to edifile

Returns newclass contained a list of mtemap infos

Return type list

read_mtemap_section(*mt_or_emap_section_list=None*)

Read mtsection and set attribute . values can be set as key [key01=value01, ..., keynn=valuenn]

Parameters **mt_or_emap_section_list** (*list*) – *mt* or *emap* section list

Example

```
>>> import csamtpy.ff.core.edi as csedi
>>> mtsection_obj= csedi.MTEMAP.get_mtemap_section_list(edi_fn␣
↪=path)
>>> info = mtsection_obj.read_mtemap_section()
>>> print(mtsection_obj.sectid)
```

write_mtemap_section(*mt_or_emap_section_list=None*, *nfreq=None*)

Method to write MT or EMAP section into list by providing list as ['key01= value01', ..., keyxx=valuexx]. Method can recognize whether edifile

provided is MT or EMAP then can read file according to.

Parameters **mt_or_emap_section_list** (*list*) – list of mt or eamp sectiionvalidate by egal ('=')

Example

```
>>> from csamtpy.ff.core.edi  import MTEMAP
>>> mtemapinfo ={'sectid':'yobkro','nfreq':18 , 'maxblks':100,
↪'hx':"1003.1",
...          'ex':'1005.4','hz':'1006.3', 'ey':1002.3 ,
...          'hy':'1000.2'}#'chksum':18, 'ndipole':47, 'type':
↪'hann'
>>> mtemapsection_obj = MTEMAP(**mtinfo)
... writeinfomtemapsect = mtemapsection_obj.write_mtemap_
↪section()
... print(writeinfomtemapsect)
```

class csamtpy.ff.core.edi.**Person**(***kwargs*)

Information for a person

Holds the following information:

Attributes	Type	Explanation
email	string	email of person
name	string	name of person
organization	string	name of person's organization
organization_url	string	organizations web address

More attributes can be added by inputing a key word dictionary

Example

```
>>> from csamtpy.ff.core.edi import Person
>>> person =Person(**{'name':'ABA', 'email':'aba@cagniard.res.org',
...                   'phone':'00225-0769980706',
...           'organization':'CagniadRES'})
>>> person.name
... ABA
>>> person.organisation
... CagniardRES
```

class csamtpy.ff.core.edi.**Processing**(***kwargs*)

Information for a Edi processing

Holds the following information:

Attributes	Type	Explanation
ProcessingSoftware	class	Software obj : Input software info.
processedby	str	name handler of dataprocessing
processingtag	str	specifictag
runlist	list	—
remoteref	str	reference point for remoting
remotesite	str	reference site name
signconvention	str	convention sign provide default

More attributes can be added by inputing a key word dictionary

class csamtpy.ff.core.edi.**References**(***kwargs*)

References information for a citation.

Holds the following information:

Attributes	Type	Explanation
author	string	Author names
title	string	Title of article, or publication
journal	string	Name of journal
doi	string	DOI number
year	int	year published

More attributes can be added by inputing a key word dictionary

Example

```
>>> from csamtpy.ff.core.edi import References
>>> refobj = References(**{'volume':18, 'pages':'234--214',
...                        'title':'pyCSAMT :python toolbox for␣
↪CSAMT' ,
...                 'journal':'Computers and Geosciences',
...                 'year':'2021', 'author':'DMaryE'})
>>> print(refobj.journal)
```

class csamtpy.ff.core.edi.**Software**(***kwargs*)

software info

Holds the following information:

Attributes	Type	Explanation
name	string	name of software
version	string	version of sotware
Author	string	Author of software
release	string	latest version release

More attributes can be added by inputing a key word dictionary

Example

```
>>> from csamtpy.ff.core.edi import Software
>>> Software(**{'release':'0.11.23'})
```

class csamtpy.ff.core.edi.**Source**(***kwargs*)

Information of the file history, how it was made

Holds the following information:

Attributes	Type	Explanation
project	string	where the project have been done
sitename	string	where the survey have been taken place
creationdate	string	creation time of file YYYY-MM-DD,hh:mm:ss
creatingsoftware	string	name of program creating the file
author	Person	person whom created the file
submitter	Person	person whom is submitting file for archiving

More attributes can be added by inputing a key word dictionary

Example

```
>>> from csamtpy.ff.core.edi import Source
>>> Source(**{'archive':'IRIS',
...           'reprocessed_by':'grad_student'})
```

class csamtpy.ff.core.edi.**Spectra**

class csamtpy.ff.core.edi.**TSeries**

csamtpy.ff.core.edi.**gather_measurement_key_value_with_str_parser**(*old_measurement_list*, *parser=None*)

fonction to rebuild xmeasurement list , to solder list with egal. In the case where no value is found at the last item, we will add "None" .

Parameters

- **old_measurement_list** (*list*) – measurement list to solder
- **parser** (*str*) – can be egal or all you want, *Default* is None mean parser '='.

Returns list solded with egal like <key=value>

Return type list

Example

```
>>> from csamtpy.ff.core.edi import gather_measurement_key_value_
↪with_str_parser
>>> measm = [ ['>HMEAS', 'ID=1001.001', 'CHTYPE=HX', 'X=', '0.0',
↪'Y=',
...                '0.0', 'Z=', '0.0', 'AZM=', '0.0', 'TS='],
...                  ['>HMEAS', 'ID=1002.001', 'CHTYPE=HY', 'X=', '0.
↪0',
...                      'Y=', '0.0', 'Z=', '0.0', 'AZM=', '90.0'],
...                  ['>HMEAS', 'ID=1003.001', 'CHTYPE=HZ', 'X=', '0.
↪0', 'Y=',
...                      '0.0', 'Z=', '0.0', 'AZM=', '0.0', 'TS=', '
↪']]
>>> for item in measm:
...     print(gather_measurement_key_value_with_str_parser(old_
↪measurement_list=item) )
... ['ID=1001.001', 'CHTYPE=HX', 'X=0.0', 'Y=0.0', 'Z=0.0', 'AZM=0.0
↪', 'TS=None']
... ['ID=1002.001', 'CHTYPE=HY', 'X=0.0', 'Y=0.0', 'Z=0.0', 'AZM=90.
↪0']
... ['ID=1003.001', 'CHTYPE=HZ', 'X=0.0', 'Y=0.0', 'Z=0.0', 'AZM=0.0
↪', 'TS=None']
```

csamtpy.ff.core.edi.**minimum_parser_to_write_edi**(*edilines*, *parser=None*)

This fonction validate edifile for writing , string with egal. We assume that dictionnary in list will be for definemeasurment E and H fieds.

Parameters

- **edilines** (*list*) – list of items to parse
- **parser** (*str*) – parser edifile section DefineMeasurement, can be change. *default* is egal (=)

Example

```
>>> from csamtpy.ff.core.edi  import DefineMeasurement
>>> file_edi= 'S00_ss.edi'
>>> path =  os.path.join(os.environ["pyCSAMT"],
...                          'csamtpy','data', file_edi_2)
>>> definemeas =DefineMeasurement.get_measurement_info(edi_fn=path)
```

(continues on next page)

(continued from previous page)

```
>>> minimparser = minimum_parser_to_write_edi(edilines =definemeas.
↪define_measurement)
>>> print(minimparser)
```

1.4 Module J

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Thu Dec 3 22:31:16 2020

@author: @Daniel03

class csamtpy.ff.core.j.**J**(*j_fn=None*, ***kws*)

Class deal with A.G. Jones j-file. see : http://mtnet.info/docs/jformat.txt

Attributes

japp_rho

jerror

jimag

jmode

jperiod

jpha

jphamax

jphamin

jreal

jrhomax

jrhomin

jweight

jwpha

jwrho

Methods

jMode([polarization_type])	Jmode is conformited a different set of H-Polarization ans E-Polarization.
jname(number_of_sites[, survey_name])	Staticmethod for generate alphanumeric *station name* (case sensitive when reading A.G Jones files) survey XXX, station 001.
read_j([j_fn])	Method to read Jformat file.

jMode (*polarization_type='RXY'*)

Jmode is conformited a different set of H-Polarization ans E-Polarization. for more detail :

See also:

http://mtnet.info/docs/jformat.txt

:Convention:The convention used is for RXY to represent the E-polarization (TE) mode, and for RYX to represent the B-polarization mode.

static jname (*number_of_sites*, *survey_name=None*)

Staticmethod for generate alphanumeric *station name* (case sensitive when reading A.G Jones files) survey XXX, station 001.

Parameters

- **number_of_sites** (*int*) – number of stations .
- **survey_name** (*str*) – place location name .

Returns list of alphanumeric station names

Return type list

read_j (*j_fn=None*)

Method to read Jformat file.

Parameters **j_fn** (*str*) – path to jfile

class csamtpy.ff.core.j.**J_collection** (*list_of_jfiles=None*, *survey_name=None*, ***kwargs*)

J collection Class . Collect jfiles

Attributes

azimuth

elevation

latitude

longitude

stnames

Methods

collect_jfiles([list_of_jfiles, jpath])	Collect the Jfile from jlist from jpath.Read and populate attributes .
plot_Topo_Stn_Azim([list_of_jfiles, plot, ...])	plot Topography , Stn _Azim from Jfiles
rewrite_jfiles([list_of_jfiles, savepath, ...])	Method to rewrite A.G.

collect_jfiles(*list_of_jfiles=None*, *jpath=None*)
Collect the Jfile from jlist from jpath.Read and populate attributes .

Parameters **list_of_jfiles** (*list*) – list of jfiles

plot_Topo_Stn_Azim(*list_of_jfiles=None*, *plot='*'*, *show_stations=False*, *compute_azimuth=True*, *savefig=None*)
plot Topography , Stn _Azim from Jfiles

Parameters

- **list_of_jfiles** (*list*) – list of jfiles
- **plot** (*str or int*) –

  type of plot '*' means all the 03 plots or use *topo*, *stn*, *azim* to plot individually. *default* is '*'
- **show_stations** (*bool*) – see the stations names as xlabel .
- **compute_azimuth** (*bool*) – if add azimuth, set azimuth to False *Default* is True
- **savefig** (*str*) – PathLike - path to save your figure

Note: Work but not stable ...

rewrite_jfiles(*list_of_jfiles=None*, *savepath=None*, *survey_name=None*, *j_extension='.dat'*)
Method to rewrite A.G. johnson file (J-Format file).

See also:

http://mtnet.info/docs/jformat.txt

Parameters

- **list_of_files** (*list*) – collection of Jfiles
- **survey_name** (*str*) – name of exploration area
- **j_extension** (*str*) – output format, *Default* is '.dat'
- **savepath** (*str*) – output directory .If None , file will be store in current work directory

class csamtpy.ff.core.j.**J_infos**(***kwargs*)
J_infos class - set the information of J_file

1.5 Module z

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

---

exception csamtpy.ff.core.z.**MT_Z_Error**

class csamtpy.ff.core.z.**ResPhase**(*z_array=None*, *z_err_array=None*, *freq=None*, ***kwargs*)
resistivity and phase container .. module:: Z .. moduleauthor:: Jared Peacock <jpeacock@usgs.gov> .. moduleauthor:: Lars Krieger

Attributes

phase

phase_det

phase_det_err

phase_err

phase_err_xx

phase_err_xy

phase_err_yx

phase_err_yy

phase_xx

phase_xy

phase_yx

phase_yy

res_det

res_det_err

res_err_xx

res_err_xy

res_err_yx

res_err_yy

res_xx

res_xy

res_yx

res_yy

resistivity

resistivity_err

Methods

compute_resistivity_phase([z_array, ...])	compute resistivity and phase from z and z_err
set_res_phase(res_array, phase_array, freq)	Set values for resistivity (res - in Ohm m) and phase (phase - in degrees), including error propagation.

compute_resistivity_phase(*z_array=None*, *z_err_array=None*, *freq=None*)
compute resistivity and phase from z and z_err

set_res_phase(*res_array*, *phase_array*, *freq*, *res_err_array=None*, *phase_err_array=None*)
Set values for resistivity (res - in Ohm m) and phase (phase - in degrees), including error propagation.

Parameters

- **res_array** (*np.ndarray(num_freq, 2, 2)*) – resistivity array in Ohm-m
- **phase_array** (*np.ndarray(num_freq, 2, 2)*) – phase array in degrees
- **freq** (*np.ndarray(num_freq)*) – frequency array in Hz
- **res_err_array** (*np.ndarray(num_freq, 2, 2)*) – resistivity error array in Ohm-m
- **phase_err_array** (*np.ndarray(num_freq, 2, 2)*) – phase error array in degrees

class csamtpy.ff.core.z.**Tipper**(*tipper_array=None*, *tipper_err_array=None*, *freq=None*)
Tipper class –> generates a Tipper-object.

Errors are given as standard deviations (sqrt(VAR))

Parameters

- **tipper_array** (*np.ndarray((nf, 1, 2), dtype='complex')*) – tipper array in the shape of [Tx, Ty] *default* is None
- **tipper_err_array** (*np.ndarray((nf, 1, 2))*) – array of estimated tipper errors in the shape of [Tx, Ty]. Must be the same shape as tipper_array. *default* is None
- **freq** (*np.ndarray(nf)*) – array of frequencies corresponding to the tipper elements. Must be same length as tipper_array. *default* is None

Attributes	Description
freq	array of frequencies corresponding to elements of z
rotation_angle	angle of which data is rotated by
tipper	tipper array
tipper_err	tipper error array

Methods	Description
mag_direction	computes magnitude and direction of real and imaginary induction arrows.
amp_phase	computes amplitude and phase of Tx and Ty.
rotate	rotates the data by the given angle

Attributes

amplitude

amplitude_err

angle_err

angle_imag

angle_real

freq

mag_err

mag_imag

mag_real

phase

phase_err

tipper

tipper_err

Methods

compute_amp_phase()	Sets attributes:
compute_mag_direction()	computes the magnitude and direction of the real and imaginary induction vectors.
rotate(alpha)	Rotate Tipper array.
set_amp_phase(r_array, phi_array)	Set values for amplitude(r) and argument (phi - in degrees).
set_mag_direction(mag_real, ang_real, ...)	computes the tipper from the magnitude and direction of the real and imaginary components.

compute_amp_phase()

Sets attributes:

- *amplitude*
- *phase*
- *amplitude_err*
- *phase_err*

values for resistivity are in in Ohm m and phase in degrees.

compute_mag_direction()
computes the magnitude and direction of the real and imaginary induction vectors.

rotate(*alpha*)
Rotate Tipper array.

Rotation angle must be given in degrees. All angles are referenced to geographic North=0, positive in clockwise direction. (Mathematically negative!)

In non-rotated state, 'X' refs to North and 'Y' to East direction.

Updates the attributes:

- *tipper*
- *tipper_err*
- *rotation_angle*

set_amp_phase(*r_array*, *phi_array*)
Set values for amplitude(r) and argument (phi - in degrees).

Updates the attributes:

- tipper
- tipper_err

set_mag_direction(*mag_real*, *ang_real*, *mag_imag*, *ang_imag*)
computes the tipper from the magnitude and direction of the real and imaginary components.

Updates tipper

No error propagation yet

class csamtpy.ff.core.z.**Z**(*z_array=None*, *z_err_array=None*, *freq=None*)
Z class - generates an impedance tensor (Z) object.

Z is a complex array of the form (n_freq, 2, 2), with indices in the following order:
- Zxx: (0,0)
- Zxy: (0,1)
- Zyx: (1,0)
- Zyy: (1,1)

All errors are given as standard deviations (sqrt(VAR))

Parameters

- **z_array** (*numpy.ndarray(n_freq, 2, 2)*) – array containing complex impedance values
- **z_err_array** (*numpy.ndarray(n_freq, 2, 2)*) – array containing error values (standard deviation) of impedance tensor elements
- **freq** (*np.ndarray(n_freq)*) – array of frequency values corresponding to impedance tensor elements.

Attributes	Description
freq	array of frequencies corresponding to elements of z
rotation_angle	angle of which data is rotated by
z	impedance tensor
z_err	estimated errors of impedance tensor
resistivity	apparent resisitivity estimated from z in Ohm-m
resistivity_err	apparent resisitivity error
phase	impedance phase (deg)
phase_err	error in impedance phase

Methods	Description
det	calculates determinant of z with errors
invariants	calculates the invariants of z
inverse	calculates the inverse of z
remove_distortion	removes distortion given a distortion matrix
remove_ss	removes static shift by assumin Z = S * Z_0
norm	calculates the norm of Z
only1d	zeros diagonal components and computes the absolute valued mean of the off-diagonal components.
only2d	zeros diagonal components
res_phase	computes resistivity and phase
rotate	rotates z positive clockwise, angle assumes North is 0.
set_res_phase	recalculates z and z_err, needs attribute freq
skew	calculates the invariant skew (off diagonal trace)
trace	calculates the trace of z

Example

```
>>> import mtpy.core.z as mtz
>>> import numpy as np
>>> z_test = np.array([[0+0j, 1+1j], [-1-1j, 0+0j]])
>>> z_object = mtz.Z(z_array=z_test, freq=[1])
>>> z_object.rotate(45)
>>> z_object.resistivity
```

Attributes

det Return the determinant of Z

det_err Return the determinant of Z error

freq Frequencies for each impedance tensor element

invariants Return a dictionary of Z-invariants.

inverse Return the inverse of Z.

norm Return the 2-/Frobenius-norm of Z

norm_err Return the 2-/Frobenius-norm of Z error

only_1d Return Z in 1D form.

only_2d Return Z in 2D form.

phase

phase_det

phase_det_err

phase_err

phase_err_xx

phase_err_xy

phase_err_yx

phase_err_yy

phase_xx

phase_xy

phase_yx

phase_yy

res_det

res_det_err

res_err_xx

res_err_xy

res_err_yx

res_err_yy

res_xx

res_xy

res_yx

res_yy

resistivity

resistivity_err

skew Returns the skew of Z as defined by Z[0, 1] + Z[1, 0]

skew_err Returns the skew error of Z as defined by Z_err[0, 1] + Z_err[1, 0]

trace Return the trace of Z

trace_err Return the trace of Z

z Impedance tensor

z_err

Methods

compute_resistivity_phase([z_array, ...])	compute resistivity and phase from z and z_err
remove_distortion(distortion_tensor[, ...])	Remove distortion D form an observed impedance tensor Z to obtain the uperturbed "correct" Z0: Z = D * Z0
remove_ss([reduce_res_factor_x, ...])	Remove the static shift by providing the respective correction factors for the resistivity in the x and y components.
rotate(alpha)	Rotate Z array by angle alpha.
set_res_phase(res_array, phase_array, freq)	Set values for resistivity (res - in Ohm m) and phase (phase - in degrees), including error propagation.

property det
Return the determinant of Z

Returns det_Z

Return type np.ndarray(nfreq)

property det_err
Return the determinant of Z error

Returns det_Z_err

Return type np.ndarray(nfreq)

property freq
Frequencies for each impedance tensor element

Units are Hz.

property invariants
Return a dictionary of Z-invariants.

property inverse
Return the inverse of Z.

(no error propagation included yet)

property norm
Return the 2-/Frobenius-norm of Z

Returns norm

Return type np.ndarray(nfreq)

property norm_err
Return the 2-/Frobenius-norm of Z error

Returns norm_err

Return type np.ndarray(nfreq)

property only_1d
Return Z in 1D form.

If Z is not 1D per se, the diagonal elements are set to zero, the off-diagonal elements keep their signs, but their absolute is set to the mean of the original Z off-diagonal absolutes.

property only_2d
Return Z in 2D form.

If Z is not 2D per se, the diagonal elements are set to zero.

remove_distortion(*distortion_tensor*, *distortion_err_tensor=None*)
Remove distortion D form an observed impedance tensor Z to obtain the uperturbed "correct" Z0: Z = D * Z0

Propagation of errors/uncertainties included

Parameters

- **distortion_tensor** (*np.ndarray(2, 2, dtype=real)*) – real distortion tensor as a 2x2
- **distortion_err_tensor** – default is None

Return type

np.ndarray(2, 2, dtype='real')

returns impedance tensor with distorion removed

Return type

np.ndarray(num_freq, 2, 2, dtype='complex')

returns impedance tensor error after distortion is removed

Return type

np.ndarray(num_freq, 2, 2, dtype='complex')

Example

```
>>> import mtpy.core.z as mtz
>>> distortion = np.array([[1.2, .5],[.35, 2.1]])
>>> d, new_z, new_z_err = z_obj.remove_
↪distortion(distortion)
```

remove_ss(*reduce_res_factor_x=1.0*, *reduce_res_factor_y=1.0*)

Remove the static shift by providing the respective correction factors for the resistivity in the x and y components. (Factors can be determined by using the "Analysis" module for the impedance tensor)

Assume the original observed tensor Z is built by a static shift S and an unperturbated "correct" Z0 :

- Z = S * Z0

therefore the correct Z will be :

- Z0 = S^(-1) * Z

Parameters

- **reduce_res_factor_x** (*float or iterable list or array*) – static shift factor to be applied to x components (ie z[:, 0, :]). This is assumed to be in resistivity scale
- **reduce_res_factor_y** (*float or iterable list or array*) – static shift factor to be applied to y components (ie z[:, 1, :]). This is assumed to be in resistivity scale

Returns static shift matrix,

Return type np.ndarray ((2, 2))

Returns corrected Z

Return type mtpy.core.z.Z

Note: The factors are in resistivity scale, so the entries of the matrix "S" need to be given by their square-roots!

---

rotate(*alpha*)

Rotate Z array by angle alpha.

Rotation angle must be given in degrees. All angles are referenced to geographic North, positive in clockwise direction. (Mathematically negative!)

In non-rotated state, X refs to North and Y to East direction.

Updates the attributes

- *z*
- *z_err*
- *zrot*
- *resistivity*

- *phase*
- *resistivity_err*
- *phase_err*

property skew
Returns the skew of Z as defined by Z[0, 1] + Z[1, 0]

Note: This is not the MT skew, but simply the linear algebra skew

Returns skew

Return type np.ndarray(nfreq, 2, 2)

property skew_err
Returns the skew error of Z as defined by Z_err[0, 1] + Z_err[1, 0]

Note: This is not the MT skew, but simply the linear algebra skew

Returns skew_err

Return type np.ndarray(nfreq, 2, 2)

property trace
Return the trace of Z

Returns Trace(z)

Return type np.ndarray(nfreq, 2, 2)

property trace_err
Return the trace of Z

Returns Trace(z)

Return type np.ndarray(nfreq, 2, 2)

property z
Impedance tensor

np.ndarray(nfreq, 2, 2)

csamtpy.ff.core.z.**correct4sensor_orientation**(*Z_prime*, *Bx=0*, *By=90*, *Ex=0*, *Ey=90*, *Z_prime_error=None*)
Correct a Z-array for wrong orientation of the sensors.
Assume, E' is measured by sensors orientated with the angles E'x: a E'y: b
Assume, B' is measured by sensors orientated with the angles B'x: c B'y: d
With those data, one obtained the impedance tensor Z': E' = Z' * B'
Now we define change-of-basis matrices T,U so that E = T * E' B = U * B'
=> T contains the expression of the E'-basis in terms of E (the standard basis) and U contains the expression of the B'-basis in terms of B (the standard basis) The respective expressions for E'x-basis vector and E'y-basis vector are the columns of T. The respective expressions for B'x-basis vector and B'y-basis vector are the columns of U.

We obtain the impedance tensor in default coordinates as:
E' = Z' * B' => T^(-1) * E = Z' * U^(-1) * B => E = T * Z' * U^(-1) * B => Z = T * Z' * U^(-1)

Parameters

- **Z_prime** – impedance tensor to be adjusted
- **Bx** (*float (angle in degrees)*) – orientation of Bx relative to geographic north (0) *default* is 0
- **By** –
- **Ex** (*float (angle in degrees)*) – orientation of Ex relative to geographic north (0) *default* is 0
- **Ey** (*float (angle in degrees)*) – orientation of Ey relative to geographic north (0) *default* is 90
- **Z_prime_error** (*np.ndarray(Z_prime.shape)*) – impedance tensor error (std) *default* is None

Dtype Z_prime np.ndarray(num_freq, 2, 2, dtype='complex')

Returns adjusted impedance tensor

Return type np.ndarray(Z_prime.shape, dtype='complex')

Returns impedance tensor standard deviation in default orientation

Return type np.ndarray(Z_prime.shape, dtype='real')

CHAPTER

TWO

PACKAGE PROCESSING

2.1 Module Dispatcher

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Thu Nov 26 20:55:39 2020

@author: @Daniel03

csamtpy.ff.processing.callffunc.**agso_data**()
Geological data codes processing

Deprecated since version function: will later deprecated

csamtpy.ff.processing.callffunc.**dipole_center_position**(*dipole_position=None*)
Generaly positions are taken at each electrode of dipole to that to easy correct data for ploting and for noise correction , we adjust coordinate by taking the center position that means , the number of points will be substract to one.

Parameters **dipole_position** (*array_like*) – postion array at each electrodes.

Returns centered position value array

Return type array_like

csamtpy.ff.processing.callffunc.**get_array_from_reffreq**(*array_loc*, *freq_array*, *reffreq_value*, *stnNames=None*)
Get array value at special frequency :param * array_loc: dictionnary of stations , array_value e.g: S00:(ndarray,1) rho_values :type * array_loc: dict , :param * freq_array: frequency array for CSAMT survey :type * freq_array: (ndarray,1) :param * reffreq_value: the value of frequency user want to get the value :type * reffreq_value: int or float :param * stnNames: list of stations names . :type * stnNames: list

Returns

array_like an array of all station with reffreq_value . e.g reffreq_value =1024. it return all value of the array at 1024Hz frequency .

csamtpy.ff.processing.callffunc.**relocate_on_dict_arrays**(*data_array*, *number_of_frequency*, *station_names=None*)

Put data arrays on dictionnary where keys is each station and value the array of that station. if station_names is None , program will create name of station. if station_names is given , function will sorted stations names . please make sure to provide correctly station according the disposal you want .

Parameters

- **number_of_frequency** (*array_like*) – array of frequency during survey
- **station_names** (*list of array_like*) – list of station

Returns infos at data stations

Return type dict

csamtpy.ff.processing.callffunc.**set_stratum_on_dict**()

Process to put geocodes_strata and geocodes_structures into dictionnaries better way to go on metaclasses merely. Thus each keys of dictionary will be its own object.

Returns

strata_dict [dict]

Disctionnary of geostrata

structures_dict [dict] Dictionnary of geostructures.

csamtpy.ff.processing.callffunc.**straighten_cac2CSfile**(*data_array*, *component_column_section=None*)

Sometimes head_sections of file _F2(CAC2CSAMT) provided is little different colunms section name according to different version . it 's better to filter and to check before returning the correct informations we need.

Parameters

*** data_array** [ndarray]

data from AVG astatic file

- **component_column_section** [list] astactic file column comps provided

Returns

array_we_need :ndarray

same infos present in the plainty /1 Avg file

array_other_comp [pd.Core.DataFrame] infos include through Astatic softwares very usefull therefore we keep it .

csamtpy.ff.processing.callffunc.**truncated_data**(*data*, *number_of_reccurence*, ***kwargs*)

Function to truncate all data according to number of frequency.

Parameters

*** data** [list, or nd.array]

data must be truncate.

- **number_of_freq** [int] number of frequency imaged.

Returns

list loc_list , data truncated on list.

csamtpy.ff.processing.callffunc.**zstar_array_to_nan**(*zstar_array*, *nan_value=nan*, *keep_str=False*)

Parameters

*** zstar_array** [ndarray]

array contain unfloat converter value. the unconverter value can be a '*'

- **nan_value** [float or np.nan type] the nan_value could be any value either int, float or str. The *default* is np.nan.
- **keep_str** [bool, optional] keep the str item on your array. f keep_str is set to false and the type nan_value is str , the program will force '**keep_str_**' to True to allow converter . The *default* is False.

Returns

ndarray zstrar_array converted .

2.2 Module Shifting

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Sat Dec 12 13:55:47 2020

@author: @Daniel03

csamtpy.ff.processing.corr.**interp_to_reference_freq**(*freq_array*, *rho_array*, *reference_freq*, *plot=False*)

Interpolate frequencies to the reference frequencies.

Parameters

- **freq_array** (*array_like*) – frequency array
- **reference_freq** (*float*) – frequency at clean data

class csamtpy.ff.processing.corr.**shifting**(*data_fn=None*, *freq_array=None*, *res_array=None*, *phase_array=None*, ***kwargs*)

processing class [shifting processing workflow] coorection class deal with AVG Zonge station file "*.stn" or SEG-EDI file.

Attributes

app_rho

frequency

phase

referencefreq

Methods

AMA()	** future plan ****
FLMA()	***Future plan **
TMA([data_fn, freq_array, res_array, ...])	Trimmed-moving-average filter to estimate average apparent resistivities at a single static-correction-reference frequency.

AMA()
** future plan ****

FLMA()
Future plan *

TMA(*data_fn=None*, *freq_array=None*, *res_array=None*, *phase_array=None*, *stnVSrho_loc=None*, *reference_freq=None*, *number_of_TMA_points=5*)
Trimmed-moving-average filter to estimate average apparent resistivities at a single static-correction-reference frequency. User can compute TMA by inputing only the data file . the program vill find automaticall other parameters . If not may provide all the parameters excepth the data file .

Parameters

*** data_fn** [str]

path to avg file or edi file .

- **freq_array** [array_like (ndarray,1)] frequency array of at normalization frequency (reference value) of all stations. station j to n .(units = Hz)
- **res_array** [dict of array_like (ndarra,1)] dict of array of app.resistivity at reffreq. from station j to n.
- **phase_array** [dict of array_lie(ndarray,1), dict of array of phase at reffreq.] from station j to n. (unit=rad)value of frequency with clean data . (unit=Hz)
- **stnVSrho_loc** [dict] set of dictionnary of all app.resistivity data from station j to n . (optional)
- **num_of_TMA_point :int** window to apply filter .

Returns

dict rho_corrected , value corrected with TMA filter from station j to n.

1. corrected data from [AVG] ..

Example

```
>>> from csamtpy.ff.processing.corr import Shifting
>>> path =  os.path.join(os.environ["pyCSAMT"],
...          'csamtpy','data', LCS01.AVG)
... static_cor =shifting().TMA (data_fn=path,
...                             reference_freq=1024.,
↪number_of_TMA_points =5 )
```

2. **corrected from edifiles [EDI]**

Example

```
>>> from csamtpy.ff.core.cs import CSAMT
>>> from csamtpy.ff.processing.corr import Shifting
>>> edipath = r'C:/Users\Administrator\Desktop
↪est\edirewrite'
>>> csamt_obj =CSAMT(edipath =edipath)
>>> static_cor =shifting().TMA( reference_freq =256. ,
...                                freq_array = csamt_
↪obj.freq ,
...                                res_array = csamt_
↪obj.resistivity ,
...                             phase_array =csamt_obj.
↪phase ,
...                             number_of_TMA_points=5)
... print(static_cor)
```

2.3 Module ZCalculator

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Thu Dec 3 16:44:29 2020

@author: @Daniel03

csamtpy.ff.processing.zcalculator.**comp_phz**(*comphz_array*, *units='deg'*)

PHZc are from each data block, units in rad

Parameters **comphz_array** (*float*) – average parameters phase for data blocs.

Returns component phase averaged.

Return type component phase averaged.

Example

```
>>> path =  os.path.join(os.environ["pyCSAMT"],
...              'csamtpy','data', 'K1.AVG')
>>> from csamtpy.core import avg
>>> phs_obj =avg.Phase(path)
>>> phs_obj.loc['S00']
>>> value, ss = comp_phz(comphz_array=phs_obj.loc[
...            'S00'], to_degree=True)
... print(value)
```

csamtpy.ff.processing.zcalculator.**comp_rho**(*mag_E_field*, *mag_H_field*, *freq_array*, *A_spacing*, *Txcurr*)

Function to compute component average unit in in ohm.m

Parameters

- **mag_E_field** (*np.ndarray (ndarray,1)*) – magnitude of E-filed , averaged
- **mag_H_field** (*np.ndarray (ndarray,1)*) – magnitude of H-Field , averaged
- **freq_array** (*np.ndarray (ndarray,1)*) – frequency of station field
- **A_spacing** (*np.float*) – step_between station
- **Txcurr** (*np.float,*) – distance of coil in meter

Returns comp_rho , component averaged rho.

Return type np.ndarray

csamtpy.ff.processing.zcalculator.**compute_FLMA**(*z_array=None*, *weighted_window=None*, *dipole_length=None*, *number_of_points=None*)

Note: We will add this filter later !

***future plan ***

csamtpy.ff.processing.zcalculator.**compute_TMA**(*data_array=None*, *number_of_TMApoints=None*)

function to compute a trimmed-moving-average filter to estimate average apparent resistivities.

Parameters

- **data_array** (*array_like(ndarray,1)*) – content of value to be trimmed
- **points** (*number_of_TMA*) – number of filter points .

Returns value corrected with TMA

Return type array_like (ndarray, 1)

csamtpy.ff.processing.zcalculator.**compute_components_Z_Phz**(*magn_E_field*, *magn_H_field*, *phz_E_field*, *phz_H_field*, *freq_value*, ***kwargs*)

Function to compute all components derived from Impedance Z. user can enter specifik units in kwargs arguments . program will compute and converts value automatically.

Parameters

* **magn_E_field** [np.ndarray]

E_.field magnitude (ndarray,1) in microV/KM*A

- **magn_H_field** [np.ndarray] H_.field magnitude (ndarray,1)in mGammas/A or picoTesla/A
- **phz_E_field** [np.ndarray] E_field phase (ndarray, 1) in mrad
- **phz_H_field** [np.ndarray] H_field phase (ndarray,1) in mrad.
- **freq_value** [np.ndarray] Frequency at which data was measured(ndarray,1)in Hz
- **kwargs** [str] units conversion.

Returns

rho: ndarray

Cagnard resistivity calculation. ohm.m

phz: ndarray Impedance phase value.

Zij: ndarray Impedance Tensor value.

Zreal: ndarray Value of Real part of impedance Tensor.

Zimag: float Value of Imaginary part of impedance Tensor.

Zreal_imag: ndarray , complex Complex value of impedance Tensor.

Example

```
>>> from csamtpy.core import avg
>>> path =  os.path.join(os.environ["pyCSAMT"],
...              data', 'avg', 'K1.AVG')
>>> emag_ob = avg.Emag(path)
>>> hmag_obj = avg.Hmag(path)
>>> ephz_obj = avg.Ephz(path)
>>> hphz_obj = avg.Hphz(path)
>>> freq_obj =avg.Frequency(path)
>>> station_name ='S00'
>>> rho, phz, Z, real, imag, comp =compute_components_Z_
↪Phz(
...    magn_E_field=emag_ob.loc[station_name],
...                              magn_H_field =hmag_obj.
↪loc[station_name],
...                              phz_E_field =ephz_obj.
↪loc[station_name],
...                              phz_H_field=hphz_obj.
↪loc[station_name],
...                              freq_value=freq_obj.
↪loc[station_name])
... rho
```

Raises

CSex :pyCSAMT exception , Exceptions if units entered by the user doesnt match or are messy.

csamtpy.ff.processing.zcalculator.**compute_sigmas_e_h_and_sigma_rho**(*pc_emag*, *pc_hmag*, *pc_app_rho*, *app_rho*, *emag*, *hmag*)

function to compute Standard Deviation for E-field (sigma_e), Standard Deviation for H-Field (sigma_h) , & Standard Deviation for Component RHO (sigma_rho)

Parameters

*** pc_emag** [float]

Statistical variation of magnitude values from averaged data blocks. Standard Deviation/Average Emag (%)

- **pc_hmag :float** Statistical variation of magnitude values from averaged data blocks. Standard Deviation / Average Hmag (%)
- **pc_app_rho: float** Statistical variation of magnitude values from averaged data blocks. Standard Deviation / Average Rho (%)
- **app_rho :float** resistivity calculated from averaged component (ohm.m)
- **Emag** [float] average E - field magnitude(microVolt/Km *amp)
- **Hmag** [float] average H - field magnitude(pTesta/amp) or (milliGammas/Amp)

Returns

sigma_rho [float]

srhoC (Standard Deviation for Component RHO)

c_var_Rho [float] C-varrhoC(Coefficient of Variation for Component RHO)

csamtpy.ff.processing.zcalculator.**find_reference_frequency**(*freq_array=None*, *reffreq_value=None*, *sharp=False*, *etching=True*)

Method to find and interpolate reference value if it is not present on the frequency range.

Parameters

- **freq_array** (*array_like*) – array_like frequency range
- **reffreq_value** (*float or int*) – reference frequency value
- **sharp** (*bool*) – if set to True , it forces the program to find mainly a value closest inside the frequency range.
- **etching** (*bool*) – bool , if set to True , it will print in your stdout.

Returns reference frequency

Return type float

csamtpy.ff.processing.zcalculator.**get_data_from_reference_frequency**(*array_loc*, *freq_array*, *reffreq_value*)

Function to get reference frequency without call especially stations array. The function is profitable but . It's less expensive However if something wrong happened by using the first step to get a reference array , it will try the traditionnally function to get it. If none value is found , an Error will occurs.

Parameters

- **array_loc** (*dict*) – assume to be a dictionnary of stations_data_values.
- **freq_array** (*array_like*) – frequency array
- **reffreq_value** (*float or int*) – float or int , reffrence value If the reference value is not in frequency array , function will force to interpolate value amd find the correlative array.

Returns an array of reference value at specific index .

Return type array_like

Example

```
>>> get_data_from_reference_frequency(array_loc=rho,freq_array=freq_
↪array,
...                                   reffreq_value=1023.)
... Input reference frequency has been interpolated to < 1024.0 > Hz
```

csamtpy.ff.processing.zcalculator.**get_reffreq_index**(*freq_array*, *reffreq_value*)

Get the index of reference index. From this index ,All array will filter data at this reffreq value . :param freq_array: array of frequency values :type freq_array: array_like

Parameters **reffreq_value** (*float, int*) – value of frequency at clean data

csamtpy.ff.processing.zcalculator.**hanning**(*dipole_length=50.0*, *number_of_points=7*, *large_band=False*)

Function to compute hanning window .

Parameters

- **dipole_length** (*float*) – the length of dipole , xk is centered between dipole
- **number_of_points** (*int*) – number of filter points

Returns windowed hanning

Return type array_like

csamtpy.ff.processing.zcalculator.**hanning_x**(*x_point_value*, *dipole_length=50.0*, *number_of_points=7*, *bandwidth_values=False*, *on_half=True*)

Function to compute point on window width . Use discrete computing . Function show the value at center point of window assume that the point is center locate on the window width . It intergrates value between dipole length. User can use see_extraband to see the values on the total bandwith. If half is False the value of greater than center point will be computed and not be 0 as the normal definition of Hanning window filter.

Parameters

- **x_point_value** (*float*) – value to intergrate.
- **dipole_length** (*float*) – length of dipole on survey
- **number_of_point** (*int*) – survey point or number point to apply.
- **bandwidth_values** – see all value on the bandwith , value greater than x_center point will be computed .

:type bandwidth_values:bool

Parameters **on_half** (*bool*) – value on the bandwith; value greater that x_center point = 0.

Returns hannx integrated X_point_value or array of window bandwidth .

Return type array_like

csamtpy.ff.processing.zcalculator.**hanning_xk**(*dipole_length=50.0*, *number_of_points=7*)

compute _hanning window on a wtdth of number of point : integrate value on all the window_bandwidth discrete and continue. if value is greater than Hald of the width value == 0 .

Parameters

- **dipole_length** (*float*) – length of dipole
- **number_of_points** (*int*) – value of points or survey stations .

Returns han_xk , continue value on half bandwidth x0– xk (center point)

Return type array_like

Returns windowed hanning, discrete _value ,SUM(han(x0, xk))

Return type array_like

csamtpy.ff.processing.zcalculator.**interpolate_sets**(*array_to*, *fill_value=None*, *array_size=None*)

Function to interpolate data contain of multiple nan values.

csamtpy.ff.processing.zcalculator.**mag_avg**(*mag_array*, *A_spacing*, *Txcur*)

RAW, E-, or H-field magnitude values)for each frequency units : mV/Km*

Parameters

- **mag_array** (*np.array (ndarray, 1),*) – magnnitude value for each data block
- **a_spacing** (*float*) – dipole length
- **txcur** – curv coil, transmitter length

Returns averaged data of magnitude data Block

Rtype mag_avg np.ndarray

csamtpy.ff.processing.zcalculator.**param_phz**(*pphz_array*, *to_degree=False*)

PHZn are from each data block , units in mrad

Parameters

- **pphz_array** (*array_like*) – average parameters phase for data blocs.
- **to_degree** (*bool*) – ascertain convertion to degree

csamtpy.ff.processing.zcalculator.**param_rho**(*rho_array*)

Parameter Average RHO (RHOp), RHO are from each data block, unit (ohm.m)

Parameters **rho_array** (*array_like*) – array of resistivity values

csamtpy.ff.processing.zcalculator.**perforce_reference_freq**(*dataset*, *frequency_array=None*)

Function to get automatically the reference frequency. If user doesnt provide the value , function will find automatically value .

Parameters

- **data** (*array_like*) – array of avg DATA, ndim>1
- **frequency_array** (*array_like*) – array of frequency

Returns reffreq_value float , reference frequency value

Return type float

Returns uncover_index, index of reference value on frequency array

Return type int

Returns nan_ratio , the ratio or the prevalence of nan in the data_set

Return type float

csamtpy.ff.processing.zcalculator.**phz_avg**(*phz_array*, *to_degree*)

E-, H-field, or Impedance Phase values unit in mrad

Parameters

- **phz_array** (*array_like*) – array of phase values in mrad
- **to_degree** (*bool*) – ascertain convertion to degree

csamtpy.ff.processing.zcalculator.**rhophi2z**(*phase*, *freq*, *resistivity=None*, *z_array=None*)

Function to compute z , real part and imag part .

Parameters

- **phase** (*ndarray*) – phase angles array in radians

- **freq** (*array_like*) – frequencies array in Hz
- **resistivity** (*array_like*) – rho array in ohm.m
- **z_array** (*array_like*) – impedance z array in V/m

Returns z_abs , absolute value of zz

Return type float

Returns z_real, real part of complex number

Return type float

Returns z_imag, imaginary part of zz

Return type complex

Returns ndarray

Return type zz, array of z_abs, z_imag, z_real

csamtpy.ff.processing.zcalculator.**wbetaX**(*Xpos*, *dipole_length=50.0*, *number_of_points=7*)

weight Beta is computed following the paper of Torres-verdfn, C., and F. X. Bostick, 1992, Principles of spatial surface electric field filtering in magnetotellurics- Electromagnetic array profiling (EMAP)- Geophysics, 57(4), 25–34.

Parameters

- **Xpos** (*str*) – reference position on the field
- **dipole_length** (*float*) – length of dipole measurement
- **number_of_points** (*int*) – point to stand filters , window width

csamtpy.ff.processing.zcalculator.**weight_beta**(*dipole_length=50.0*, *number_of_points=7*, *window_width=None*)

WeightBeta function is weight Hanning window . if window width is not provide , function will compute the width of window.

See also:

Torres-Verdin and Bostick, 1992, Principles of spatial surface electric field filtering in magnetotellurics: electromagnetic array profiling (EMAP), Geophysics, v57, p603-622. ...

Parameters

- **dipole_length** (*float*) – length of dipole in meter (m)
- **number_of_points** (*int*) – number of station points to filter
- **window_width** (*float*) – the width of window filter

Returns beta_array at each station

Return type array_like

csamtpy.ff.processing.zcalculator.**z_error2r_phi_error**(*z_real*, *z_imag*, *error*)

Error estimation from rectangular to polar coordinates. By standard error propagation, relative error in resistivity is 2*relative error in z amplitude. Uncertainty in phase (in degrees) is computed by defining a circle around the z vector in the complex plane. The uncertainty is the absolute angle between the vector to (x,y) and the vector between the origin and the tangent to the circle.

Parameters

- **z_real** (*float*) – real component of z (real number or array)
- **z_imag** (*complex*) – imaginary component of z (real number or array)

- **error** (*float*) – absolute error in z (real number or array)

Returns containers of relative error in resistivity, absolute error in phase

Rtupe tuple

CHAPTER

THREE

PACKAGE MODELING

3.1 Module Occam2D

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

See also:

- http://marineemlab.ucsd.edu/Projects/Occam/sharp/index.html
- **https://marineemlab.ucsd.edu/~kkey/Software.php** ...

Note: Actually this module is not able to write Occam2D meshes file or build OccamInputfiles. You can import MTpy and use it to build Occam2Dinputfiles. for next realease , we will call mtpy directly.

@author: KouaoLaurent alias @Daniel03 Created on Fri Jan 22 20:31:14 2021

class csamtpy.modeling.occam2d.**Data**(*data_fn=None*, ***kwargs*)
Read and write Occam data

Methods

read_occam2d_datafile([data_fn])	read_occam_data file.
write_occam2d_data()	

read_occam2d_datafile(*data_fn=None*)
read_occam_data file. and populates attributes

Parameters **fn** (*str*) – full path to occam data file .

Example

```
>>> path =os.path.join(os.environ ['pyCSAMT'],
...                        'csamtpy', 'data', 'occam2D',
...                        'OccamDataFile.dat')
>>> occamdata_obj =Data(fn = path)
>>> print(occamdata_obj.occam_data_format)
>>> print(occamdata_obj.occam_data_blocks)
>>> print(occamdata_obj.occam_data_title)
>>> print(occamdata_obj.occam_data_sites)
>>> print(occamdata_obj.occam_data_frequencies)
>>> print(occamdata_obj.occam_data_offsets)
>>> occamDATA = occamdata_obj.occam_data
```

write_occam2d_data()

class csamtpy.modeling.occam2d.**DataBlock**(***kwargs*)

Read OccamDataBlock aand set corresponding attributes

Attributes	Description (Restriction)
site	number of the site from the site list that this data belongs to.
freq	number of the frequency from the frequency list.
type	type of data
datum	data value
error	size of the error for this measurement For log10 resistivity this value can look a little strange. It is derived from the calculation d(log_10(x))/dx = 1/[x ln(10)]. So for 10% error, dx = 0.10x thus d(log_10(x)) = 0.10x / x ln(10) =.1/ln(10) = 0.0434

Example

```
>>> form csamtpy.ff.modeling import DataBlock
>>> np.random.seed(1983)
>>> dblocks_dict = {'site' : ['S{0:02}'.format(ii)  for ii in range
↪(10)],
...                 'freq': np.linspace(1, 8912, 10),
...                 'type' :(5,6),
...                 'data' : np.power(10, np.random.randn(20*2)),
...                 'error' : np.random.randn(20)
...                 }
>>> dblock_obj = DataBlock(**dblocks_dict)
>>> print(dblock_obj.freq )
>>> print(dblock_obj.error)
>>> print(dblock_obj.site)
```

Methods

decode_each_site_data(data_blocks, . . .)	Decode each site data and get differents values array :param datablocks: datafrom differents blocks :type datablocks: ndarray

static decode_each_site_data(*data_blocks*, *data_type_index*)

Decode each site data and get differents values array :param datablocks: datafrom differents blocks :type datablocks: ndarray

Parameters **data_type_index** (*int*) – specify the data type index

class csamtpy.modeling.occam2d.**Iter**(*iter_fn=None*, ***kwargs*)

Occam iteration file, inherets from startup obj , in fact two object a similar the same

Methods

read_occam2d_iterfile([iter_fn])	read_occam iteration file and populate attributes
read_occam2d_startupfile([startup_fn])	read occam2d_startupfile

read_occam2d_iterfile(*iter_fn=None*)

read_occam iteration file and populate attributes

Parameters **iter_fn** (*str*) – full path to iteration file

Example

```
>>> from csamtpy.ff.modeling.occam2d import Iter
>>> path =os.path.join(os.environ ['pyCSAMT'], 'csamtpy',
...                    'data', 'occam2D', 'ITER17.iter')
>>> iter_obj=Iter(iter_fn= path)
... print(iter_obj.occam_iter_param_count)
... iterDATA= iter_obj.occam_iter_data
```

class csamtpy.modeling.occam2d.**Iter2Dat**(*iter2dat_fn=None*, *model_fn=None*, *data_fn=None*, *iter_fn=None*, *mesh_fn=None*, ***kwargs*)

Iter2Dat is a format converter which convert *.iter file and related mesh files to so called 'x,y,z' *.data file for post-processing. The class was inpired from the Bo Yang matlab script . reading functions come from 'plotOccam2DMT.m' routine

See also:

Occam routine <http://marineemlab.ucsd.edu/Projects/Occam/sharp/index.html > of SIO, UCSD.

Bo Yang matlab-script is on *add.info* sub-packages of pyCSAMT. If your are familiar with matlab and you want to rewrite the script please contact author at:

Methods

read_iter2dat([iter2dat_fn, bln_fn, scale, ...])	Read YangBo iter2data file or provided Occam 2D specific files .
read_occamfiles([path_to_occamfiles])	getffiles and readfiles to populates speciales attributes
write_iter2dat_file([filename, model_fn, ...])	write 'x,y,z' *.data file for post-processing can read and rewrite iter2dat file

read_iter2dat(*iter2dat_fn=None*, *bln_fn=None*, *scale='km'*, *model_fn=None*, *iter_fn=None*, *mesh_fn=None*, *doi='1km'*, *data_fn=None*, *occam_model_obj=None*, ***kws*)

Read YangBo iter2data file or provided Occam 2D specific files .

Parameters

- **iter2dat_fn** (*str*) – full path to iter2dat file
- **bln_file** (*str*) – full path to bln file
- **scale** – str , scale of output data . Most of time , Bo yang iter2Dat file is converted in kilometer . If not turn

  the scale to *m*.

Example

```
>>> from csamtpy.modeling.occam2d import Iter2Dat
>>> i2d='iter17.2412.dat'
>>> i2d_2='K1.iter.dat'
>>> bln='iter17.2412.bln'
>>> bln_2 = 'K1.bln'
>>> pathiter = os.path.join(os.path.dirname(os.environ ['pyCSAMT
↪']),
...                              '_iter2dat_', i2d_2)
>>> pathbln = os.path.join(os.path.dirname(os.environ ['pyCSAMT
↪']),
...                             '_iter2dat_', bln_2)
>>> occam_iter2dat_obj =Iter2Dat(iter2dat_fn=pathiter,
...                                  bln_fn =pathbln )
>>> i2d_data = occam_iter2dat_obj.read_iter2datfile()
>>> i2d_sta, i2d_depth = occam_iter2dat_obj.model_x_nodes,
>>> occam_iter2dat_obj.model_z_nodes
>>> i2d_model_res = occam_iter2dat_obj.model_res
```

read_occamfiles(*path_to_occamfiles=None*, ***kws*)

getffiles and readfiles to populates speciales attributes

Parameters **path_to_occamfiles** (*str*) – full path to occamfiles [ITER|DATA|RESP] files.

Example

```
>>> path =os.path.join(os.environ ['pyCSAMT'], 'csamtpy', 'data
↪', 'occam2D')
>>> iter2_obj=Iter2Dat(path_to_occamfiles= path)
>>> sites= iter2_obj.OccamData.occam_data_sites
>>> freq =iter2_obj.OccamData.occam_data_frequencies
>>> iter_roughn =iter2_obj.OccamIter.occam_iter_roughness_value
>>> iter_misfit=iter2_obj.OccamIter.occam_iter_misfit_reached
```

(continues on next page)

(continued from previous page)

```
>>> forward_data = iter2_obj.OccamResponse.forward_data
>>> residual_data = iter2_obj.OccamResponse.residual
```

write_iter2dat_file(*filename=None*, *model_fn=None*, *iter_fn=None*, *mesh_fn=None*, *data_fn=None*, *doi='1km'*, *savepath=None*, *occam_model_obj=None*, *negative_depth=True*, *scale='km'*, ***kws*)

write 'x,y,z' *.data file for post-processing can read and rewrite iter2dat file

Parameters

- **x** (*array_like*) – offset
- **y** (*array_like*) – depth
- **z** (*array_like*) – log10 resistivities

params	Type	Description
model_fn	str	full path to occam model file
iter_fn	str	full path to iteration file
mesh_fn	str	full path to occam mesh file
data_fn	str	fll path to occam data file
filename	str	output of iter2dat file
doi	str of float	depth of investigation. default is 1km if your povide value on float, default unit is "meter" eg : 2000=2000m
negative_depth	bool	if True , will provide file with negative depth value
scale	str	scaled the offset value and elevation . might be [m|km]
elevation	ndarray|list	can be provided if usefull

1. Write with Occam 2D files

Examples

```
>>> from csamtpy.modeling.occam2d import Iter2Dat as i2d
>>> data='OccamDataFile.dat'
>>> mesh = 'Occam2DMesh'
>>> model = 'Occam2DModel'
>>> path =os.path.join(os.environ ['pyCSAMT'], 'csamtpy',
...                        'data', 'occam2D')
...                    #,'OccamDataFile.dat')
>>> pathi2d =os.path.join(os.environ ['pyCSAMT'], 'csamtpy',
↪'data', '_iter2dat_')
>>> occam_iter2dat_obj =i2d(mesh_fn=os.path.join(path, mesh),
...                     iter_fn = os.path.join(path, iter_),
...                     model_fn =os.path.join(path, model),
...                     data_fn =os.path.join(path, data))
>>> occam_iter2dat_obj.write_iter2dat_file()
```

2. Rewrite the with iter2dat file

Example

```
>>> from csamtpy.modeling.occam2d import Iter2Dat as i2d
>>> iter_='ITER17.iter'
>>> idat = K1.iter.20142.dat
>>> bln = K1.iter.20142.bln
>>> occam_iter2dat_obj =i2d(iter2dat_fn=os.path.join(pathi2d,␣
↪idat),
                            bln_fn = os.path.join(pathi2d, bln)
>>> occam_iter2dat_obj.write_iter2dat_file()
```

class csamtpy.modeling.occam2d.**Mesh**(*mesh_fn=None*, ***kwargs*)

Read Occam read mesh file

Methods

read_occam2d_mesh([mesh_fn])	Read mesh and get mesh values data and populates attributes

read_occam2d_mesh(*mesh_fn=None*)

Read mesh and get mesh values data and populates attributes

Parameters **mesh_fn** (*str*) – full path to mesh file

class csamtpy.modeling.occam2d.**Model**(*model_fn=None*, *iter_fn=None*, *mesh_fn=None*, ***kwargs*)

Occam Model , Actually read model file

Methods

read_occam2d_modelfile([model_fn, mesh_fn, ...])	read and ascertain modelfile.

read_occam2d_modelfile(*model_fn=None*, *mesh_fn=None*, *iter_fn=None*)

read and ascertain modelfile.

Parameters

- **model_fn** (*str*) – full path to OCCAM2D model file
- **mesh_fn** (*str*) – full path to MESH model file
- **iter_fn** (*str*) – full path to ITER model file

Example

```
>>> path =os.path.join(os.environ ['pyCSAMT'], 'csamtpy',
...                    'data', 'occam2D', 'Occam2DModel')
>>> occammodel_obj =Model(model_fn = path)
... print(occammodel_obj.model_name)
... print(occammodel_obj.model_mesh_file)
... print(occammodel_obj.model_num_layers)
```

class csamtpy.modeling.occam2d.**Response**(*response_fn=None*, *data_fn=None*, ***kwargs*)

Occam response file . Response is paired with the iteration file above. One is output at the end of each successful iteration. Inherets from Occam 2D data file .

Methods

read_occam2d_datafile([data_fn])	read_occam_data file.
read_occam2d_responsefile([response_fn])	Read Occam2D response file.
write_occam2d_data()	

read_occam2d_responsefile(*response_fn=None*)

Read Occam2D response file.

Parameters **response_fn** – full path to response file

:type response_fn:str

Example

```
>>> from csamtpy.modeling.occam2d import Response
>>> pathresp =os.path.join(os.environ ['pyCSAMT'], 'csamtpy',
...                            'data', 'occam2D',RESP17.resp )
>>> path_data =os.path.join(os.environ ['pyCSAMT'], 'csamtpy',
...                             'data', 'occam2D',OccamDataFile.
↪dat )
>>> resp_obj = Response(response_fn=pathresp, data_fn = path_
↪data )
>>> respDATA= resp_obj.resp_data_value
>>> resp_obj.occam_mode
>>> resp_obj.occam_dtype
>>> forward_data = resp_obj.resp_forward_value
>>> residual_data = resp_obj.resp_residual_value
>>> tm_log10= resp_obj.resp_tm_log10
>>> tm_phase=resp_obj.resp_tm_phase
>>> tm_forward = resp_obj.resp_tm_log10_forward
>>> tm_residual = resp_obj.resp_tm_log10_forward
>>> tm_forward = resp_obj.resp_tm_log10_residual
>>> tm_phase_error = resp_obj.resp_tm_phase_err
>>> tm_phase_err = occam_resp_obj.resp_tm_log10_forward_phase
>>> tm_residual_phase_err = occam_resp_obj.resp_tm_log10_
↪residual_phase_err
```

class csamtpy.modeling.occam2d.**Startup**(*startup_fn=None*, ***kwargs*)

Occam startup file Actually read startup file. for more detail:

See also:

http://marineemlab.ucsd.edu/Projects/Occam/sharp/index.html

Methods

read_occam2d_startupfile([startup_fn])	read occam2d_startupfile

read_occam2d_startupfile(*startup_fn=None*)

read occam2d_startupfile

Parameters **startup_fn** (*str*) – full path to startup_file

Example

```
>>> from csamtpy.ff.modeling.occam2d import Startup
>>> path =os.path.join(os.environ ['pyCSAMT'],
...                        'csamtpy', 'data', 'occam2D', 'Startup
↪')
>>> startup_obj=Startup(startup_fn = path)
... print(startup_obj.occam_startup_data_file)
```

class csamtpy.modeling.occam2d.**occamLog**(*fn=None*, ***kwargs*)

Class to deal with occcam 2d logfile . File output after inverted data.

Attributes

fn

Methods

read_occam2d_logfile([fn])	Read occam file and populate attributes.

read_occam2d_logfile(*fn=None*)

Read occam file and populate attributes.

Parameters **fn** (*str*) – full path to occam2d log file

Example

```
>>> from csamtpy.ff.modeling import occam2d
>>> path =os.path.join(os.environ ['pyCSAMT'], 'csamtpy', 'data
↪',
...                        'occam2D', 'logFile.logfile')
>>> occamlog_obj =occam2d.Log(fn = path)
>>> print(occamlog_obj.rms.shape)
>>> print(occamlog_obj.roughness.shape)
```

CHAPTER

FOUR

PACKAGE GEOCORE

4.1 Module Geodrill

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Sat Sep 19 12:37:42 2020 @author: Daniel03

class geodrill.geoCore.geodrill.**Drill**(*well_filename=None*, *build_manually_welldata=False*, ***kwargs*)

This class is focus on well logs . How to generate well Log for Oasis:

Methods

dhGeology([dh_geomask])	Method to build geology drillhole log.
dhSample([path_to_agso_codefile, dh_sampmask])	Method to build Sample log.
dhSurveyElevAz([add_elevation, add_azimuth])	Method to build Elevation & Azimuth DH logs. if add_elevation and . add_azimuth are set . The programm will ignore the computated azimuth, and it will replace to the new azimuth provided . all elevation will be ignore and set by the new elevation . *kwargs arguments {add_elevation , add-azimuth } must match the same size like the number of Drillholes . Each one must be on ndarray(num_of_holes, 1).

continues on next page

Table 1 – continued from previous page

writeDHData([data2write])	Method to write allDH logs. It depends to the users to sort which data want to export and which format. the program support only two format (.xlsx and .csv) if one is set , it will ouptput the convenience format. Users can give a list of the name of log he want to export. Program is dynamic and flexible. It tolerates quite symbols number to extract data logs.

dhGeology(*dh_geomask=None*)

Method to build geology drillhole log. The name of input rock must feell exaction accordinag to a convention AGSO file . If not sure for the name of rock and Description and label . you may consult the geocode folder before building the well_filename. If the entirely rock name is given , program will search on the AGSO file the corresponding Label and code . If the rock name is founc then it will take its CODE else it will generate exception.

Parameters

*** dh_geomask** [np.ndarray, optional] geology mask. send mask value can take exactly the np.ndarray(num_of_geology set ,). The better way to set geology maskis to fill on the wellfilename. if not , programm will take the general mask value. The *default* is None.

Returns

pd.DataFrame geology drillhole log.

dhSample(*path_to_agso_codefile=None*, *dh_sampmask=None*)

Method to build Sample log. This method focuses on the sample obtained during the DH trip.it may georeferenced as the well_filename needed. A main thing is to set the AGSO_STCODES file. AGSO_STCODES is the conventional code of structurals sample. If you have an own AGSO_STCODES , you may provide the path * kwargs=path_to_ags_codefile * . the program will read and generate logs according to the DESCRIPTION and STCODES figured. if None, the program will take it STCODES and set the samplelogs. When you set the Sample code aor sample name , make sur that the name match the same name on STCODES. If not , program will raises an error.

Parameters

*** path_to_agso_codefile** [str, optional]

path to conventional AGSO_STRUCTURAL CODES. The *default* is None.

- **dh_sampmask** [np.ndarray, optional] Structural mask. The default is None.

Returns

pd.DataFrame Sample DH log.

dhSurveyElevAz(*add_elevation=None*, *add_azimuth=None*, ***kwargs*)

Method to build Elevation & Azimuth DH logs. if add_elevation and . add_azimuth are set . The programm will ignore the computated azimuth, and it will replace to the new azimuth provided . all elevation will be ignore and set by the new elevation . *kwargs arguments {add_elevation , add-azimuth } must match the same size like the number of Drillholes . Each one must be on ndarray(num_of_holes, 1).

Parameters

*** add_elevation** [np.nadarray , optional]

elevation data (num_of_holes, 1) The *default* is None.

- **add_azimuth** [np.ndarray , optional] azimuth data (num_of_holes,1). The *default* is None.
- **DH_RL :np.float or np.ndarray(num_of_hole,1),** if not provided , it's set to 0. means No topography is added'.

Returns

pd.Dataframe

Elevation DH log .

pd.DataFrame Azimuth DH log.

writeDHData(*data2write=None*, ***kwargs*)

Method to write allDH logs. It depends to the users to sort which data want to export and which format. the program support only two format (.xlsx and .csv) if one is set , it will ouptput the convenience format. Users can give a list of the name of log he want to export. Program is dynamic and flexible. It tolerates quite symbols number to

extract data logs.

Parameters

*** data2write** [str or list , optional]

the search key. The default is None.

- **datafn :str** savepath to exported file *Default* is current work directory.
- **write_index_on_sheet** [bool,] choice to write the sheet with pandas.Dataframe index.
- **writeType** [str ,] file type . its may *.csv or *.xlsx* . **Default* is *.xlsx
- **add_header** [bool,] add head on exported sheet. set False to mask heads. *Default* is True.
- **csv_separateType** [str ,] Indicated for csv exported files , the type of comma delimited . defaut is ',' .

class geodrill.geoCore.geodrill.**Geodrill**(*iter2dat_fn=None*, *model_fn=None*, *data_fn=None*, *iter_fn=None*, *mesh_fn=None*, *bln_fn=None*, ***kwargs*)

Class to manage data from Occam2D and Model so to create section of each sites and it depth.

Each station constitues an attribute framed by two closest point of station offsets from model resistivities. Deal with True resistivities get on the survey or with others firms. In fact , input truth resistivities values into our model , produce an accuracy underground map.The challenge to build pseudolog allow to know how layers are disposal in underground so to emphasize the large conductive zone especially in the case of groundwater exploration. Program works in combinaison with geophysic data especially Occam 2D inversion data, and geological data. Actually the program deal only with Occam 2D inversion files or Bo Yang (x,y,z) files. We intend to extend later with other external softares but can generate output directly see use with Golder sofware('surfer'). If user

has a golder software installed on its computer , It can use output files generated here to produce 2D map so to compare both maps to see how far

is the difference between Model map and detail-sequences map) “pseudosequences model” could match better the reality of underground). Details sequences map is most closest to the reality When {step descent} parameter is not too small at all. Indeed True geological data allow to harmonize the value of resistivity produced by Occam2D model so to force the pogramm to make a correlation between data fromtruth layers and the model values.

Methods

geo_build_strata_logs([input_resistivities, ...])	”
geo_replace_rho([input_resistivity_range, ...])	Allow ro replace the calculated resistivities from model to real resistivites obtained on survey area with other companies.
get_average_rho(data_array[, transpose])	Function to average rho to one point to onother . It show the lowest point and the maximum point averaged . Function averaged rho value between local maximum and local minima values . if data values of station are located on columnlines , set transpose to True then rotate the matrix to find minima and maxima locals value then calculated averaged rho after will return matrix transpose as the same shape as inputted . .Defaut is **False** .
get_geo_formation_properties(...[, ...])	Get the list of stuctures and their names of after replaced , flexible tools.
get_structure(resistivities_range)	function to get according the range of resistivities values , the corresponding associated geological rocks The list of electrical properties of rocks is not exhaustive , can be fill by others
set_geodata([iter2dat_fn, model_fn, ...])	Readmodel data and collected from each site its value from surface to depth
to_golden_software([input_resistivities, ...])	Output average files, rehoreplaced files, spseudosequennces files and station locations files will generate 3 outputs for Golden software plots
to_oasis_montaj([model_fn, iter_fn, ...])	write output to oasis montaj when station loocation and profile coordinates are provided . We assune before using this method , you are already the coordinates files at disposal(*stn), if not use the method *to_golden_software*.Coordinated files are Easting Northing value not in degree decimals . It uses occam 2D outputfiles or Bo Yang iter2Dat file also add profile XY coordinates (utm_zone).

geo_build_strata_logs(*input_resistivities=None*, *input_layers=None*, *step_descent=None*, ***kwargs*)

” Read resistivits data got on survey area and build geological strata block. If constrained_electrical_properties_of_rocks is False , will build a log by considering the conventional electrical property of rocks can be find through this link :

See also:

https://www.eoas.ubc.ca/ubcgif/iag/foundations/properties/resistivity.htm list is not exhaustive and depend of the geological formations of survey area.

Parameters

*** input_resistivity :array_like**

an array of resistivity on the site

- **step_descent** [float] depth value to averaged rho . Must be smaller as possible if None , it take the 2% times the investigation depth
- **input_layers** [list or array] layers_names , eg *granite*, *fault*, *river*

A Sample of electrical_properties_of_rocks:

========== ========== ==========

Rocks Max Rho Min Rho (ohm-m)

========== ========== ==========

igneous rocks 10^6 10^3

duricrust 5.10^3 5.10^2

gravel and sand 10^4 10^2.90(800)

conglomerate 10^4 10^1.95(90)

dolomite/limestone 10^5 10^3

permafrost 10^5 10^2.62(750)

metamorphic rocks 5.10^2 10.^1

tills 8.10^2 10^1.93(85)

standstone conglomerate 10^4 10^1.92 (80)

lignite/coal 10^2.89(790) 10^1

shales 10^1.7(50) 10^1.48(30)

clays 10^2 10^1.7(50)

saprolite 10^2.08(120) 10^1.48(30)

sedimentary rocks 10^4 10^0

fresh water 3.10^2 10^0

salt water 10^0 10^-0.15

massive sulfide 10^0 10^-2

sea water 10^-0.09(0.8) 10^-1

ore minerals 10^0 10^-4

Graphite 10^-2.5 10^-3.5

========== ========== ==========

.. note :: list is not Exhaustive, use the data base script to populate most of goeological electrical properties.

geo_replace_rho(*input_resistivity_range=None*, *input_layers=None*, ***kwargs*)

Allow ro replace the calculated resistivities from model to real resistivites obtained on survey area with other companies. The accuracy depend How many number of resistivites are. More resitivities ,more accuracy in the designed of underground model geostratigrapyhy model.

Parameters

- **input_resistivity_range** (*array_like*) – an array of resistivity on the site
- **input_layers** – layers_names , eg .`granite`, *fault*, `river `
- **input_layers** – list or arrays
- **depth_range** – array _of depth of specific layer

static get_average_rho(*data_array*, *transpose=False*)

Function to average rho to one point to onother . It show the lowest point and the maximum point averaged . Function averaged rho value between local maximum and local minima values . if data values of station are located on columnlines , set transpose to True then rotate the matrix to find minima and maxima locals value then calculated averaged rho after will return matrix transpose

as the same shape as inputted . .Defaut is **False** .

Parameters **data_array** (*ndarray*) – data of resistivities collected at the site point

static get_geo_formation_properties(*structures_resistivities*, *real_layer_names=None*, *constrained_electrical_properties_of_rocks=True*, ***kwargs*)

Get the list of stuctures and their names of after replaced , flexible tools. wherever structures provided, the name, color , as well as the pattern . if constrained electrical properties if True , will keep the resistivities with their corresponding layers as reference. If the layer names if found on the data Base then , will return its pattern and color else defaultcolor is black and patter is "+.-". if constrained_electrical_properties_of_rocks is False , will check under data base to find the resistivities that match better the layers

Parameters

*** structures resistivities** [array_like,]

resistivities of structures

- **real_layer_names** [array_like |list] names of layer of survey area if not provided , will use resistivities to find the closet layer that match the best the resistivities
- **constrained_electrical_properties_of_rocks: bool** set to True mean the realy_layer is provided. if not program will enforce to False , will use default conventional layers Default is False, assume to povided layer names for accuracy

Returns

f_name: array_like

names of formations find with their corresponding rho

f_pattern: array_like patter of differents geological formations

f_color: array_like color of differents geological formations

static get_structure(*resistivities_range*)

function to get according the range of resistivities values , the corresponding associated geological rocks The list of electrical properties of rocks is not exhaustive , can be fill by others

Parameters **resistivities_range** (*array_like,*) – array of input_resistivities

Returns the list of geological structures form go_electrical _rocks properties

Return type list

set_geodata(*iter2dat_fn=None*, *model_fn=None*, *data_fn=None*, *iter_fn=None*, *mesh_fn=None*, ***kwargs*)

Readmodel data and collected from each site its value from surface to depth

1. Read with Occam 2D outputs files

Example

```
>>> form csamtpy.geodrill import Geodrill
>>> path_occam2d = os.path.join(os.environ ['pyCSAMT'],
...                                'csamtpy', 'data', 'occam2D')
>>> path_i2d =os.path.join(os.environ ['pyCSAMT'],
...                            'csamtpy', 'data', '_iter2dat_2')
>>> geo_obj =Geodrill(model_fn  = os.path.join(path_occam2d,
↪'Occam2DModel'),
...                   mesh_fn = os.path.join(path_occam2d,
↪'Occam2DMesh'),
...                   iter_fn = os.path.join(path_occam2d,
↪'ITER17.iter'),
...                   data_fn = os.path.join(path_occam2d,
↪'OccamDataFile.dat'))
```

2. Read with only iter2dat file and bln file

Example

```
>>> form csamtpy.geodrill import Geodrill
>>> geo_obj =Geodrill(iter2dat_fn  = os.path.join(path_i2d, 'K1.
↪iter.dat'),
...                   bln_fn = os.path.join(path_i2d, 'K1.bln'),)
>>> geo_model_off = geo_obj.model_x_nodes
>>> geo_model_res= geo_obj.model_res
>>> geoS01 = geo_obj.geo_name_S01
>>> geoS00= geo_obj.geo_name_S00
>>> geoS46=geo_obj.geo_name_S46
```

to_golden_software(*input_resistivities=None*, *input_layers=None*, *step_descent=None*, *filename=None*, *savepath=None*, ***kwargs*)

Output average files, rehoreplaced files, spseudosequennces files and station locations files will generate 3 outputs for Golden software plots

1. **One model for rho averaged (*_aver), transitory data betwen** the calculated rho and true rho.
2. **second for rho value replaced . Replaced calcualted model** structures resistivities

to their closest resistivities as resistivities reference from input resistivities.

3. **the most important files: the cut out resistivities value** with step descent (._sd) show most dominant stratigraphy sequences .

4. the station location file (.bln)

Plot the 3 files in Golden software to see transition from model calculation to truthsequence detail models which is most closest to reality.

Parameters

*** input_resistivities** [array_like]

Truth values of resistivities

- **filename :str** name of output file
- **step_descent** [float,]

  Step to cut out data and to force resistivites calcualted to match the reference data as input resistivities if not provided the step will be 20% of D0I.
- **input_layers** [array_like]

  True input_layers names (geological informations of encountered layers)
- **savepath** [str,] full path to the savepath , if None , will create folder name to savepath

========== ========== ==========

Other params Type Explanation

========== ========== ==========

elevation array_like elevation of survey area

to_negative_depth bool export deth in negative value or positive default is "negative"

scale str scale to export data .Must be *m* or *km*. *default* is **m**

========== ========== ==========

to_oasis_montaj(*model_fn=None*, *iter_fn=None*, *profile_fn=None*, *mesh_fn=None*, *data_fn=None*, *filename=None*, *savepath=None*, ***kwargs*)

write output to oasis montaj when station loocation and profile coordinates are provided . We assune before using this method , you are already the coordinates files at disposal(*stn), if not use the method *to_golden_software*.Coordinated files are Easting Northing value not in degree decimals . It uses occam 2D outputfiles or Bo Yang iter2Dat file also add profile XY coordinates (utm_zone).

Parameters

*** model_fn** [str]

full path to Occam2D model file

- **mesh_fn** [str] full path to Occam2D mesh file
- **data_fn** [str] full path to Occam2D data file
- **iter_fn** [str] full path to Occam2D iteration file OR Bo Yang (x, y, z) files

- **iter2dat_fn** [str] full path to iter2dat file (see _occam2d module to see which file is it. or call

  occam2d.Iter2Dat.__doc__)

- **bln** [str] full path to satation location file profile files (Easting , Northing Coordinates) , "see :ref:module-cs"
- **profile_fn: str** full path to station profile file . You can useProfile module to rewrite _coordinate files
- **write_negative_depth: bool** output negative depth. *Default* is True
- **scalled_east_north: tuple** scalled the easting and northing. Substract or add value to easting or northing values. first *index1* equal easting and *index2* equal Nothing.*Default* is (0,0).

========== ========== ==========

Other params Type Explanation

========== ========== ==========

filename str New name of output file . *Default* is None

normalize_depth bool Set the depth ega spacing depth . In fact for oasis montaj ,depth must be equidistant the same spacing when image in deeper. *default* is True. if false , will generate depth as Occam2D mesh z nodes.

easting array_like Easting UTM coordinates .Profile *Stn* file is provided, no need to input. *default* is None

northing array_like Northing coordinates . If station coordinates is provided from *stn file .It will detect automatically. *default* is None.

elevation array_like Elevation area . *Default* is None.

output_s_XY bool Scalled coordinates output files. *Default* is True

input_rho array_like input resistivities of True geological formations (optional) if input resistivities is provided will generated output step descent files (_sd), roughness_rho (_rr), rho_averaged (_aver). *Default* is output the main resistivty model.

input_layers array_like list of true names of layers(opt.)

step_descent float Step to cut out data and to force resistivites calcualted to match the reference data as input resistivities if not provided the step will be 20% of D0I.(opt)

writeType str Writer format *.csv or *.xlsx . *Default* is *.xlsx

add_header bool Add head on exported sheet. set False to mask heads. *Default* is True.

csv_separateType str Indicated for csv exported files. The type of comma delimited. *Defaut* is ','

to_log10 bool if True [will ouput all] resistivities data to log10 values

========== ========== ==========

class geodrill.geoCore.geodrill.**Geosurface**(*path=None*, ***kwargs*)

Read Multidata from oasis montaj output files generated by *geodrill* module . Class to Build a depth surface map for imaging .

Attributes

extension_file

Methods

get_depth_surfaces([path, depth_values])	get the depth surfaces for multi-lines and build the numpy corresponding array at that depth .
read_oasis_files([path])	Method to get depth spacing , station info data infos, Each line becomes it own attributes components of info values are *Stations*, *Easting_X_m*,`Northing_Y_m`, *v_H_m*,`x_m` *Norm_h_m*,`sets_m` *DOI_max_m*.
write_file([path, depth_values, fileformat])	Write output files.

get_depth_surfaces(*path=None*, *depth_values=None*)

get the depth surfaces for multi-lines and build the numpy corresponding array at that depth .

Parameters

- **depth_values** (*float or array_like*) – array of depth
- **path** (*str*) – full path to oasis outputfiles .

read_oasis_files(*path=None*)

Method to get depth spacing , station info data infos, Each line becomes it own attributes components of info values are *Stations*, *Easting_X_m*,`Northing_Y_m`, *v_H_m*,`x_m` *Norm_h_m*,`sets_m` *DOI_max_m*.

Parameters **path** (*str*) – full path to the oasis files .

Example

```
>>> from geodrill.geoCore.geodrill  import Geosurface
>>> path =  r'F:/__main__csamt__\oasis data\OASISWORKSll_data'
>>> geo_surface_obj = Geosurface( path =path )
>>> geo_surface_obj.read_oasis_files()
>>> geofilenames = geo_surface_obj.filenames
```

... note::To get the values of line K1_cor_oas "K1_cor_oas.csv do `k1_obj = geo_surface_obj.K1_cor_oas`

write_file(*path=None*, *depth_values=None*, *fileformat='.csv'*, ***kwargs*)

Write output files. Output files are *.xlsx* or *.csv* .

Parameters

- **path** (*str*) – full path to *geodrill* ouput files
- **depth_value** (*float or array_like*) – depth values for imaging

:param fileformat:*xlsx* or *csv* are actually the acceptable formats. :type fileformat: str

geodrill.geoCore.geodrill.**ascertain_layers_with_its_resistivities**(*real_layer_names*, *real_layer_resistivities*)

Assert the length of of real resistivites with their corresponding layers. If length of resistivities of larger that the layer's names, then will add layer that match the best the remained resistivities. If the length of layer is larger than resistivities , to avoid miscomputation , will cut out this more layer and work only the length of resistivities provided.

Parameters

*** real_layer_names: array_like , list**

list of input layer names as real layers names encountered in area

- **real_layer_resistivities :array_like , list** list of resistivities get on survey area

Returns

list real_layer_names, new list of input layers

geodrill.geoCore.geodrill.**geo_length_checker**(*main_param*, *optional_param*, *force=False*, *param_names=('input_resistivities', 'input_layers')*, ***kws*)

Geo checker is a function to check differents length of different params.

the length of optional params depend of the length of main params . if the length of optional params is larger than the length of main params, the length of optional params will be reduce to the length of main params . if the optional params length is shorther than the length of main params, will filled it either with "None" if dtype param is string of 0.if dtype params is float or 0 if integer.if Force is set True , it will absolutely check if the main params and the optional params have the same length. if not the case , will generate an error occurs.

Parameters

*** main_param** [array_like, list]

main parameter that must took its length as reference length

- **optional params** [array_like, list] optional params, whom length depend to the length of main params
- **param_names** [tuple or str] names of main params and optional params so to generate error if exits.
- **fill_value: str, float, optional** Default value to fill thearray in the case where the length of optional param is less than the length of the main param .If None , will fill according to array dtype

Returns

array_like optional param truncated according to the man params

class geodrill.geoCore.geodrill.**geostrike**

Class to deal with computation with profile angle and geo_electrical strike. Compute Profile angle and strike angle

Need to import scipy.stats as one module . Sometimes import scipy

differently with stats may not work . either *import scipy.stats* rather than *import scipy as sp* to use : *sp.stats.linregress* .

Methods

compute_geoelectric_strike([profile_angle, ...])	Compute geoelectric strike
compute_profile_angle([easting, northing])	Essentially dedicated to compute geoprofile angle.

static compute_geoelectric_strike(*profile_angle=None*, *easting=None*, *northing=None*, ***kws*)

Compute geoelectric strike

Parameters

*** profile_angle** [float]

If not provided , will comput with easting and northing coordinates

- **easting** [array_like] Easting coordiantes values
- **northing** [array_like] Northing coordinates values
- **geo_strike** [float] strike value , if provided, will recomputed geo_electric strike .

Returns

———

float profile_angle in degree E of N

float geo_electric_strike in degrees E of N

str message of return

static compute_profile_angle(*easting=None*, *northing=None*)

Essentially dedicated to compute geoprofile angle.

Parameters

*** easting** [array_like]

easting coordiantes values

- **northing** [array_like] northing coordinates values

Returns

float

profile_angle

float geo_electric_strike

str message of return

geodrill.geoCore.geodrill.**get_closest_value**(*values_range*, *input_value*)

Fonction to get closest values when input values is not in the values range we assume that values are single on array. if the same value is repeated will take the first index and the value at that index

Parameters

- **values_range** (*array_like*) – values to get
- **input_value** (*float,*) – specific value

Returns the closest value and its index

Rtye float

4.2 Module Structural

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

:synopsis:class for geological structural analysis contains some conventional structure can populate the data

Created on Sat Nov 28 21:19:13 2020

@author: @Daniel03

class `geodrill.geoCore.structural.`**Geo_formation**(*agso_file=None*, ***kwargs*)

This class is an axilliary class to supplement geodatabase , if the GeodataBase doesnt reply to SQL request , then use this class to secah information about structures . If SQL is done as well , program won't call this class as rescure . Containers of more than 150 geological strutures.

Note: replace in attributes param "**" by the *name of struture*

Attributes	Type	Explanation
names	array_like	names of all geological strutures
codes	array_like	names of all geological codes
**code	str	code of specific geological structure
**label	str	label of specific structure
**name	str	label of specific structure
**pattern	str	pattern of specific structure
**pat_size	str	pattern size of specific structure
**pat_density	str	pattern density l of specific structure
**pat_thickness	str	pttern thickess of specific structure
**color	str	color of specific structure

1. To see the names of strutures , write the script below

   Example

```
>>> from geodrill.geoCore.strutural import Geo_formation as gf
>>> geo_structure = gf()
>>> geo_structure.names
```

2. To extract color and to get the code of structure like amphibolite

   Example

```
>>> from geodrill.geoCore.strutural import Geo_formation as gf
>>> geo_structure = gf()
>>> geo_structure.amphibolite['color']
>>> geo_structure.amphibolite['code']
>>> geoformation_obj.AMP['color']
... 'R128G128'
```

Attributes

agso_fn

class geodrill.geoCore.structural.**Structure**(***kwargs*)

Class for typical geological strutural conventions for AGSO_STCODES . All geological structural informations are geostructral object.

Holds the following information:

More attributes can be added by inputing a key word dictionary

Example

```
>>> from geodrill.geoCore.structural import Structure
>>> structural=Structure()
>>> boudin=boudin_axis()
>>> print(boudin.code)
>>> print(structural.boudin_axis.name)
>>> print(structural.boudin_axis.color)
```

class geodrill.geoCore.structural.**banding_gneissosity**(***kwargs*)

Special class for banding_gneissosity

Holds the following information:

Attributes	Type	Explanation
code	str	conventional code
label	str	named label
size	str	drawing size
pattern	str	drawing pattern
density	str	elmts density
thickness	str	drawing thickness
color	str	color set

More attributes can be added by inputing a key word dictionary

class geodrill.geoCore.structural.**boudin_axis**(***kwargs*)

Special class for boudins_axis

Holds the following information:

Attributes	Type	Explanation
code	str	conventional code
label	str	named label
size	str	drawing size
pattern	str	drawing pattern
density	str	elmts density
thickness	str	drawing thickness
color	str	color set

More attributes can be added by inputing a key word dictionary

class geodrill.geoCore.structural.**fault_plane**(***kwargs*)
Special class for fault_plane

Holds the following information:

Attributes	Type	Explanation
code	str	conventional code
label	str	named label
size	str	drawing size
pattern	str	drawing pattern
density	str	elmts density
thickness	str	drawing thickness
color	str	color set

More attributes can be added by inputing a key word dictionary

class geodrill.geoCore.structural.**fold_axial_plane**(***kwargs*)
Special class for fold_axial_plane

Holds the following information:

Attributes	Type	Explanation
code	str	conventional code
label	str	named label
size	str	drawing size
pattern	str	drawing pattern
density	str	elmts density
thickness	str	drawing thickness
color	str	color set

More attributes can be added by inputing a key word dictionary

class geodrill.geoCore.structural.**fracture_joint_set**(***kwargs*)
Special class for fracture_joint_set

Holds the following information:

Attributes	Type	Explanation
code	str	conventional code
label	str	named label
size	str	drawing size
pattern	str	drawing pattern
density	str	elmts density
thickness	str	drawing thickness
color	str	color set

More attributes can be added by inputing a key word dictionary

class geodrill.geoCore.structural.**geo_pattern**
Singleton class to deal with geopattern with other modules. It is and exhaustive pattern dict, can be add and change. This pattern will be depreacted later , to create for pyCSAMT, its owwn geological pattern in coformity with the conventional geological swatches .

deal with USGS(US Geological Survey) swatches- references and FGDC (Digital cartographic

Standard for Geological Map Symbolisation -FGDCgeostdTM11A2_A-37-01cs2.eps)

make _pattern:{'/', '', 'l', '-', '+', 'x', 'o', 'O', '.', '*'} / - diagonal hatching - back diagonal | - vertical - - horizontal + - crossed x - crossed diagonal o - small circle O - large circle . - dots * - stars

geodrill.geoCore.structural.**get_color_palette**(*RGB_color_palette*)

Convert RGB color into matplotlib color palette. In the RGB color system two bits of data are used for each color, red, green, and blue. That means that each color runs on a scale from 0 to 255. Black would be 00,00,00, while white would be 255,255,255. Matplotlib has lots of pre-defined colormaps for us . They are all normalized to 255, so they run from 0 to 1. So you need only normalize data, then we can manually select colors from a color map

Parameters **RGB_color_palette** (*str*) – str value of RGB value

Returns rgba, tuple of (R, G, B)

Return type tuple

Example

```
>>> from geodrill.geoCore.structural import get_color_palette
>>> get_color_palette (RGB_color_palette ='R128B128')
```

class geodrill.geoCore.structural.**s_fabric**(***kwargs*)

Special class for s_fabric

Holds the following information:

Attributes	Type	Explanation
code	str	conventional code
label	str	named label
size	str	drawing size
pattern	str	drawing pattern
density	str	elmts density
thickness	str	drawing thickness
color	str	color set

More attributes can be added by inputing a key word dictionary

class geodrill.geoCore.structural.**sharp_contact**(***kwargs*)

Special class for sharp_contact

Holds the following information:

More attributes can be added by inputing a key word dictionary

class geodrill.geoCore.structural.**undifferentiated_plane**(***kwargs*)

Special class for undifferentiated_plane

Holds the following information:

Attributes	Type	Explanation
code	str	conventional code
label	str	named label
size	str	drawing size
pattern	str	drawing pattern
density	str	elmts density
thickness	str	drawing thickness
color	str	color set

More attributes can be added by inputing a key word dictionary

4.3 Module Strata

CHAPTER

FIVE

PACKAGE GEODATABASE

5.1 Module Recorder

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Wed Oct 14 13:38:13 2020

@author:kkouaoLaurent alias @Daniel03

class geodrill.geoDB.sql_recorder.**GeoDataBase**(*geo_structure_name=None*)

Geodatabase class . Currently we do not create the specific pattern for each geostructures. DataBase is built is built following the

codef *code*, *label*,`__description`,`pattern`, *pat_size*,`pat_density`, *pat_thickness*,`RGBA`, *electrical_props*, *hatch*, *colorMPL*, *FGDC* .

Attributes

colorMPL return geocolorMPL

electrical_props return electrical property

hatch

pattern return geopattern

rgb return georgb

static _add_geo_structure(*new_geological_rock_name=None*, ***kws*)

Add new _geological information into geodatabase .

_avoid_injection()

For secure , we do not firstly introduce directly the request. We will check whether the object *request* exists effectively in our GeoDatabase . if not , request will be redirect to structural and strata class issue from module *structural*

_get_geo_structure(*structure_name=None*)
After checking wether the name of structures exists , let find the geoformation properties from geodatabase .

Parameters **struture_name** (*str*) – name of geological rock or layer

_reminder_geo_recorder(*geo_structure_name*)
To have reminder of geological formation into the geodatabase , this method allow to output information if the structure doesnt not exist, An error will occurs.

Parameters **geo_structure_name** (*str*) – name of geological formation

property _setGeoDatabase

Note: property of GeoDataBase -create the GeoDataBase Setting geoDataBase table No Need to reset the DataBase at least you dropped the table, avoid to do that if you are not sure of what you are doing.

_update_geo_structure(*geo_formation_name=None*, ***kws*)
Update _indormation into geoDataBase .

Remember that the geodatabase is build following this table codef 'code','label','__description','pattern', 'pat_size', 'pat_density','pat_thickness','rgb','electrical_props', 'hatch', 'colorMPL', 'FGDC'.

Parameters **geo_formation_name** (*str*) – name of formation be sure the formation already exists in the geoDataBase if not an error occurs

- Update the electrical property of basement rocks = [1e99, 1e6]

Example

```
>>> from geodrill.geoDB.sql_recorder import GeoDataBase
>>> GeoDataBase()._update_geo_structure(**{'__description':
↪'basement rocks',
                                        'electrical_props':[1e99, 1e6 ]}
↪)
```

property colorMPL
return geocolorMPL

property electrical_props
return electrical property

property pattern
return geopattern

property rgb
return georgb

class geodrill.geoDB.sql_recorder.**Recorder_sql**(*database*, *table=None*, ***kwargs*)
Class to record data from file or pd.core.DataFrame and to tranfer into SQL database.

Methods

arrangeData_for_dictapp(datalist, **kwargs)	Function overwritten from "set_on_dictapp func".
keepDataInfos(data[, new_tablename])	Function to KeepData from file infos .
recordData(data[, new_tablename])	Function to KeepData from file infos .
set_on_dict_app(datalist, **kwargs)	Function overwritten from "set_on_dictapp func".
transferdata_to_sqlDB([record_list, ...])	Function to transfer Data from Dict_app to SQL DataBase.

static arrangeData_for_dictapp(*datalist*, ***kwargs*)

Function overwritten from "set_on_dictapp func". Reorganise data to dict_app model.

Parameters

*** datalist** [list, dict] list of value providen for fill the dict_app.

Returns

dict datalist, Data arranged according to dict_app arrangement.

Raises

pyCSAMTError_SQL_manager None dataname detected

static keepDataInfos(*data*, *new_tablename=None*, ***kwargs*)

Function to KeepData from file infos . the function is otherwritten fo RecordData. The difference between two function is that function organise data from each row of columns

Parameters

*** data** [str, np.array, list, or pd.core.DataFrame object]

Data ca, be on the format above or filename of data if the argument "data" is a filename, we must be convert on ".csv" format.

- **new_tablename** [str, optional] Name of database. if name is not given , the function return only list . The default is None.

Returns

list list of value in the case of no name is providen for tablename. else return dict if name of datatable is providen.

Raises

IndexError.

if lengh of number of columns like heads of data

does not match the data.shape[0], then errors will occurs.

static recordData(*data*, *new_tablename=None*, ***kwargs*)

Function to KeepData from file infos . the function is otherwritten fo RecordData. The difference between two function is that function organise data from each row of columns

Parameters

*** data** [str, np.array, list, or pd.core.DataFrame object]

Data ca, be on the format above or filename of data if the argument "data" is a filename, we must be convert on ".csv" format.

- **new_tablename** [str, optional] Name of database. if name is not given , the function return only list . The default is None.

Returns

list list of value in the case of no name is providen for tablename. else return dict if name of datatable is providen.

Raises

IndexError.

if lengh of number of columns like heads of data

does not match the data.shape[0], then errors will occurs.

static set_on_dict_app(*datalist*, ***kwargs*)

Function overwritten from "set_on_dictapp func". Reorganise data to dict_app model.

Parameters

*** datalist** [list, dict] list of value providen for fill the dict_app.

Returns

dict datalist, Data arranged according to dict_app arrangement.

Raises

pyCSAMTError_SQL_manager None dataname detected

transferdata_to_sqlDB(*record_list=None*, *filename=None*, *table_name=None*, ***kwargs*)

Function to transfer Data from Dict_app to SQL DataBase. Users can use this function by including several arguments. The function will build the data , arrange it and put it in the dataBase by commit the dataBase. Use only this func is benefit. It is better to revise arguments of that function.

Parameters

*** record_list** [dict, optional]

Dictionnay build according the dict_app model. The *default* is None.

- **filename** [str, optional] file must be on ".csv" format. The default is None.
- **table_name** [str, optional] Name of DataBase Table. The default is None.
- **comments** [str] little comment to identify your database table.
- **path_to_sqlDataBase** [str] path where the SQL dataBase is located .
- **visualize_table_creating_query** [bool] If the connexion to server is unlikable set to True to see whether query entered is right or wrong.
- **Drop_DB_Tables** [str,]

  way to drop table in SQL Database . set litteral arguments like

  the name of database user want to drop or [no "*" or all to drop all tables.
- **Ready_to_transfer** [str] process to commit Database , the curso tranfered the dataBase to SQL connexion. set litteral 'no' or 'yes' to do.
- **close_connexion** [bool ,] set True when transfer is done . it seems connexion.close()

Raises

Exception occurs when Table Name is not set on dict_app.

Note: The process of organization is full request of PostgreSQL ..

5.2 Module Interface

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

synopsis Specially dedicate to Manage SQL

Created on Tue Oct 13 15:28:57 2020

@author: @Daniel03

class geodrill.geoDB.interfaceDB.**ManageDB**(*db_name=None*, *db_host=None*)

build a datable postgre Sql from dict_app.py simple way to make a transit between two objects One object dict_app to populate DataBase

Methods

closeDB()	simple method to close Database.
commit()	special commit method for the database when cursor and connexion are still open.
connect_DB([db_host, db_name])	Create sqqlite Database
dicT_sqlDB(dicTables, **kwargs)	Method to create Table for sqlDataBase .
drop_TableDB(dicTables[, drop_table_name, ...])	Drop the name of table on dataBase or all databases.
executeReq(query[, param])	Execute request of dataBase with detection of error.
export_req([query, export_type])	method to export data from DataBase
print_query([column_name])	return the result of the previous query.

closeDB()

simple method to close Database.

commit()

special commit method for the database when cursor and connexion are still open.

connect_DB(*db_host=None*, *db_name=None*)

Create sqqlite Database

Parameters

- **db_host** (*str*) – DataBase location path

- **db_name** (*str*) – str , DataBase name

dicT_sqlDB(*dicTables*, ***kwargs*)

Method to create Table for sqlDataBase . Enter Data to DataBase from dictionnary. Interface objet : Database _Dictionnary to see how dicTable is arranged , may consult dict_app module

Parameters

*** dictTables** [dict] Rely on dict_app.py module. it populates the datababse from dictionnay app

Returns

str execute queries from dict_app

:Example :

```
>>>  mDB=GestionDB (dbname='memory.sq3,
...                    db_host =os.getcwd()')
>>> mDB.dicT_sqlDB(dicTables=Glob.dicoT,
...                  visualize_request=False)
>>> ss=mB.print_last_query()
>>> print(ss)
```

drop_TableDB(*dicTables*, *drop_table_name=None*, *drop_all=False*)

Drop the name of table on dataBase or all databases.

Parameters

*** dicTables** [dict] application dictionnary. Normally provide from dict_app.py module

*** drop_table_name** [str, optional] field name of dictionnay (Table Name). The default is None.

*** drop_all** [Bool, optional] Must select if you need to drop all table. The default is False.

Raises

Exception [Errors occurs ! .]

executeReq(*query*, *param=None*)

Execute request of dataBase with detection of error.

Parameters

*** query** [str] sql_query

*** param** [str] Default is None .

export_req(*query=None*, *export_type='.csv'*, ***kwargs*)

method to export data from DataBase

Parameters

*** query** [str, optional] Sql requests. You may consult sql_request files. The default is None.

*** export_type** [Str, optional] file extension. if None , it will export on simple file. The default is '.csv'.

*** kwargs** [str] Others parameters.

Raises

Exception Print wrong sqlrequests.

Example

```
>>> from sqlrequests import SqlQ
>>> manageDB.executeReq(SqlQ.sql_req[2])
>>> ss=manageDB.print_last_Query()
>>> print(ss)
>>> manageDB.export_req(SqlQ.sql_req[-1],
                        export_type='.csv',
                        )
```

print_query(*column_name=None*)
return the result of the previous query.

5.3 Module Request

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

synopsis Deal with Boreholes and well data , requests to DataBase

Created on Tue Oct 13 17:36:35 2020

@author: @Daniel03

class geodrill.geoDB.sqlrequests.**SqlQ**
Build sql _requests - Not use for others you may populate request for your purpose. the request in not general , the user must change the request according its will .

5.4 Module DictApp

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Tue Oct 13 14:52:08 2020

@author: Daniel03

use : for sql dataBase works dico to sql_database .

class geodrill.geoDB.dict_app.**Glob**

Spaces of variables and fonctions pseudo-globales . dictionnary can be set outside the container class Glob, following the dicoT datastructuration .

Example

```
>>> value_DB =[('id_new', 'i', 'new_vision'),
...    ('infoTab','k','no comment'),
...    ('collar','d','collarDH')]
>>> Glob.dicoT.__setitem__('TableDB_set',value_DB )
```

CHAPTER

SIX

PACKAGE UTILS

6.1 Module Agso

Created on Sat Sep 26 20:30:41 2020

@author: KouaoLaurent alias @Daniel03

Class :

**Agso ** . Data of Geological Welllogs

class csamtpy.utils.agso.**Agso**
Read Agso as pandas Series Geological conventional rocks and structurals handling.

6.2 Module Decorator

class csamtpy.utils.decorator.**deprecated**(*reason*)
Description: used to mark functions, methods and classes deprecated, and prints warning message when it called decorators based on https://stackoverflow.com/a/40301488
Usage: todo: write usage
Author: YingzhiGou Date: 20/06/2017

Methods

__call__(cls_or_func)	Call self as a function.

class csamtpy.utils.decorator.**redirect_cls_or_func**(**args*, ***kwargs*)
Description: used to redirected functions or classes. Deprecated functions or class can call others use functions or classes.
Usage:
Author: @Daniel03 Date: 18/10/2020

Methods

`__call__`(cls_or_func)	Call self as a function.

6.3 Module Func-utils

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

...

Created on Sun Sep 13 09:24:00 2020 @author: @Daniel03

utils

- averageData
- concat_array_from_list
- sort_array_data
- **transfer_array_** (deprecated)
- interpol_scipy
- _set_depth_to_coeff
- **broke_array_to_**
- _OlDFUNCNOUSEsearch_fill_data (deprecated)
- _search_ToFill_Data
- straighten_out_list
- take_firstValue_offDepth
- dump_comma
- build_wellData
- compute_azimuth
- build_geochemistry_sample
- _nonelist_checker
- _order_well
- intell_index
- _nonevalue_checker
- _clean_space

*_cross_eraser * _remove_str_word * stn_check_split_type * minimum_parser_to_write_edi

csamtpy.utils.func_utils.**averageData**(*np_array*, *filter_order=0*, *axis_average=0*, *astype='float32'*)

Parameters

*** np_array** [numpy array]

must be an array data

- **filter_order** [int] must be the index of the column you want to sort
- **axis average** [int] axis you want to see data averaged, also , it is the concatenate axis default is axis=0
- **astype*: str ,** is the ndarray dtype array . change to have an outup arry dtype , you want .

Returns

numpy array Data averaged array

:Example :

```
>>> import numpy as np
>>> list8=[[4,2,0.1],[8,2,0.7],[10,1,0.18],[4,3,0.1],
...                [7,2,1.2],[10,3,0.5],[10,1,0.5],[8.2,0,1.9],
...                [10,7,0.5],[10,1,0.5],
...                [2,0,1.4],[5,4,0.5],[10,2,0.7],[7,2,1.078],
...                [10,2,3.5],[10,8,1.9]]
>>> np_test=np.array(list8)
>>> ss=averageData(np_array=np_test,filter_order=1,
...                axis_average=0, astype="int")
>>> ss
```

csamtpy.utils.func_utils.**broke_array_to_**(*arrayData*, *keyIndex=0*, *broken_type='dict'*)

broke data array into different value with their same key

Parameters

*** arrayData :np.array**

data array .

- **keyIndex** [int] index of column to create dict key

Returns

dict dico_brok ,dictionnary of array.

csamtpy.utils.func_utils.**build_geochemistry_sample**()

Build geochemistry_sample_data

Returns

np.ndarray Sample ,Geochemistry sample Data.

Example

```
>>> geoch=build_geochemistry_sample()
>>> print(geoch)
... sampleData
... [['S0X4' '0' '254.0' 'PUP']
...      ['S0X4' '254' '521.0' 'mg']
...      ['S0X4' '521' '625.0' 'tut']
...      ['S0X4' '625' '984.0' 'suj']
...      ['S0X2' '0' '19.0' 'pup']
...      ['S0X2' '19' '425.0' 'hut']
...      ['S0X2' '425' '510.0' 'mgt']
...      ['S0X2' '510' '923.2' 'pyt']]
```

Raises

Process to build geochemistry sample data manually .

csamtpy.utils.func_utils.**build_wellData**(*add_azimuth=False*, *utm_zone='49N'*, *report_path=None*, *add_geochemistry_sample=False*)

Parameters

*** add_azimuth** [Bool, optional]

compute azimuth if add_azimut is set to True. The default is False.

- **utm_zone** [Str, optional] WGS84 utm_projection. set your zone if add_azimuth is turn to True. The default is "49N".
- **report_path** [str, optional] path to save your _well_report. The default is None. its match the current work directory
- **add_geochemistry_sample: bool** add_sample_data.Set to True if you want to add_mannually Geochimistry data. default is False.

Returns

str

name of location of well .

np.ndarray WellData , data of build Wells .

np.ndarray GeolData , data of build geology.

Example

```
>>>  import numpy as np
>>>  import os, shutil
>>>  import warnings,
>>>  form _utils.avgpylog import AvgPyLog
>>>  well=build_wellData (add_azimuth=True, utm_zone=
↪"49N")
>>>  print("nameof locations
```

:",well[0])

```
>>>  print("CollarData
```

:",well[1])

```
>>>  print("GeolData
```

:", well[2]) ... nameof locations ... Shimen ... CollarData ... [['S01' '477205.6935' '2830978.218' '987.25' '-90' '0.0' 'Shi01' ... 'Wdanxl0'] ... ['S18' '477915.4355' '2830555.927' '974.4' '-90' '2.111' 'Shi18' ... 'Wdanxl0']] ... GeolData ... [['S01' '0.0' '240.2' 'granite'] ... ['S01' '240.2' '256.4' ' basalte'] ... ['S01' '256.4' '580.0' ' granite'] ... ['S01' '580.0' '987.25' 'rock'] ... ['S18' '0.0' '110.3' 'sand'] ... ['S18' '110.3' '520.2' 'agrilite'] ... ['S18' '520.2' '631.3' ' granite'] ... ['S18' '631.3' '974.4' ' rock']] ... **Shimen_wellReports_**

Raises

Exception

manage the dimentionaly of ndarrays .

OSError when report_path is not found in your O.S.

csamtpy.utils.func_utils.**compute_azimuth**(*easting*, *northing*, *utm_zone='49N'*, *extrapolate=False*)

Parameters

*** easting** [np.ndarray]

Easting value of coordinates _UTM_WGS84

- **northing** [np.ndarray] Northing value of coordinates._UTM_WGS84
- **utm_zone** [str, optional] the utm_zone . if None try to get is through gis.get_utm_zone(latitude, longitude). latitude and longitude must be on degree decimals. The default is "49N".
- **extrapolate** [bool ,] for other purpose , user can extrapolate azimuth value , in order to get the sizesize as the easting and northing size. The the value will repositionate at each point data were collected.

  Default is False as originally azimuth computation .

Returns

np.ndarray azimuth.

Example

```
>>> import numpy as np
>>> import gis_tools as gis
>>>  easting=[477205.6935,477261.7258,477336.4355,
↪477373.7903,477448.5,
...  477532.5484,477588.5806,477616.5968]
>>>  northing=[2830978.218, 2830944.879,2830900.427,␣
↪2830878.202,2830833.75,
...                 2830783.742,2830750.403,2830733.
↪734]
>>>  test=compute_azimuth(easting=np.array(easting),
...                      northing=np.array(northing),␣
↪utm_zone="49N")
>>>  print(test)
```

csamtpy.utils.func_utils.**concat_array_from_list**(*list_of_array*, *concat_axis=0*)

Small function to concatenate a list with array contents

Parameters

*** list_of_array** [list] contains a list for array data. the concatenation is possible if an index array have the same size

Returns

array_like numpy concatenated data

Example

```
>>> import numpy as np
>>>  np.random.seed(0)
>>> ass=np.random.randn(10)
>>> ass2=np.linspace(0,15,12)
>>>  ass=ass.reshape((ass.shape[0],1))
>>>  ass2=ass2.reshape((ass2.shape[0],1))
>>> or_list=[ass,ass2]
>>> ss_check_error=concat_array_from_list(list_of_
↪array=or_list,
...                                        concat_
↪axis=0)
>>>  secont test :
>>>  ass=np.linspace(0,15,14)
>>> ass2=np.random.randn(14)
>>> ass=ass.reshape((ass.shape[0],1))
>>> ass2=ass2.reshape((ass2.shape[0],1))
>>> or_list=[ass,ass2]
>>>  ss=concat_array_from_list(list_of_array=or_list,␣
↪concat_axis=0)
>>> ss=concat_array_from_list(list_of_array=or_list,␣
↪concat_axis=1)
>>> ss
>>> ss.shape
```

csamtpy.utils.func_utils.**dump_comma**(*input_car*, *max_value=2*, *carType='mixed'*)

Parameters

*** input_car** [str,]

Input character.

- **max_value** [int, optional] The default is 2.
- **carType: str** Type of character , you want to entry

Returns

Tuple of input character must be return tuple of float value, or string value

---

Note: carType may be as arguments parameters like ['value','val',"numeric", "num", "num","float","int"] or for pure character like

["car","character","ch","char","str", "mix", "mixed","merge","mer", "both","num&val","val&num&"] if not , can not possible to convert to float or integer. the *defaut* is mixed

---

Example

```
>>> import numpy as np
>>>  ss=dump_comma(input_car=",car,box", max_value=3,
...       carType="str")
>>>  print(ss)
... ('0', 'car', 'box')
```

csamtpy.utils.func_utils.**intell_index**(*datalist*, *assembly_dials=False*)

function to search index to difference value to string element like geological rocks and geologicals samples. It check that value are sorted in ascending order.

Parameters

*** datalist** [list]

list of element : may contain value and rocks or sample .

- **assembly_dials** [list, optional] separate on two list : values and rocks or samples. The default is False.

Returns

index: int

index of breaking up.

first_dial: list , first sclice of value part

secund_dial: list , second slice of rocks or sample part.

assembly [list] list of first_dial and second_dial

Example

```
>>> import numpy as np
>>> listtest =[['DH_Hole', 'Thick01', 'Thick02',
↪'Thick03',
...           'Thick04','Rock01', 'Rock02', 'Rock03',
↪'Rock04'],
...           ['S01', '0.0', '98.62776918', '204.
↪7500461','420.0266651','520', 'GRT',
...            'ATRK', 'GRT', 'ROCK','GRANODIORITE'],
...           ['S02', '174.4293956', '313.9043882',
↪'974.8945704', 'GRT', 'ATRK', 'GRT']]
>>> listtest2=[listtest[1][1:],listtest[2][1:]]
>>> for ii in listtest2 :
>>> op=intell_index(datalist=ii)
>>> print("index:
```

“,op [0])

```
>>> print('firstDials :
```

‘,op [1])

```
>>> print('secondDials:
```

‘,op [2])

csamtpy.utils.func_utils.**interpol_scipy**(*x_value*, *y_value*, *x_new*, *kind='linear'*, *plot=False*, *fill='extrapolate'*)

function to interpolate data

Parameters

*** x_value** [np.ndarray]

value on array data : original absciss

- **y_value** [np.ndarray] value on array data : original coordinates (slope)
- **x_new** [np.ndarray] new value of absciss you want to interpolate data
- **kind** [str]

  projection kind : maybe : "linear", "cubic"

- **fill** [str] kind of extraolation, if None , *spi will use constraint interpolation can be "extrapolate" to fill_value.
- **plot** [Boolean] Set to True to see a wiewer graph

Returns

np.ndarray y_new ,new function interplolate values .

Example

```
>>> import numpy as np
>>>  fill="extrapolate"
>>>  x=np.linspace(0,15,10)
>>>  y=np.random.randn(10)
>>>  x_=np.linspace(0,20,15)
>>>  ss=interpol_Scipy(x_value=x, y_value=y, x_new=x_,␣
↪kind="linear")
>>>  ss
```

csamtpy.utils.func_utils.**minimum_parser_to_write_edi**(*edilines*, *parser='='*)

This fonction validates edifile for writing , string with egal.we assume that dictionnary in list will be for define-measurment E and H fied.

Parameters

- **edilines** (*list*) – list of item to parse
- **parser** (*str*) – the egal is use to parser edifile . can be changed, default is =

csamtpy.utils.func_utils.**parse_wellData**(*filename=None*, *include_azimuth=False*, *utm_zone='49N'*)

Function to parse well information in*csv file

Parameters

*** filename** [str, optional]

full path to parser file, The default is None.

- **include_azimuth: bool ,** Way to compute azimuth automatically
- **utm_zone** [str,] set coordinate _utm_WGS84. Defaut is 49N

Returns

location: str

Name of location .

WellData [np.ndarray]

Specificy the collar Data .

GeoData [np.ndarray] specify the geology data .

SampleData [TYPE] geochemistry sample Data.

Example

```
>>> import numpy as np
>>> dir_=r"F:\OneDrive\Python\CodesExercices\ex_
↪avgfiles\modules"
>>> parse_=parse_wellData(filename='Drill&
↪GeologydataT.csv')
>>> print("NameOflocation:
```

",parse_[0])

```
>>> print("WellData:
```

",parse_[1])

```
>>> print("GeoData:
```

",parse_[2])

```
>>> print("Sample:
```

",parse_[3])

Raises

FileNotFoundError if typical file deoesnt match the *csv file.

csamtpy.utils.func_utils.**round_dipole_length**(*value*)

small function to graduate dipole length 5 to 5. Goes to be reality and simple computation .

Parameters **value** (*float*) – value of dipole length

Returns value of dipole length rounded 5 to 5

Return type float

csamtpy.utils.func_utils.**sort_array_data**(*data*, *sort_order=0*, *concatenate=False*, *concat_axis_order=0*)

Function to sort array data and concatenate numpy.ndarray

Parameters

*** data** [numpy.ndarray]

must be in simple array , list of array and dictionary whom the value is numpy.ndarray

- **sort_order** [int, optional] index of colum to sort data. The default is 0.
- **concatenate** [Boolean , optional] concatenate all array in the object. Must be the same dimentional if concatenate is set to True. The *default* is False.

- **concat_axis_order** [int, optional] must the axis of concatenation . The default is axis=0.

Returns

numpy.ndarray data , Either the simple sort data or array sorted and concatenated .

csamtpy.utils.func_utils.**stn_check_split_type**(*data_lines*)

Read data_line and check for data line the presence of split_type < ',' or ' ', or any other marks.> Threshold is assume to be third of total data length.

Params data_lines list of data to parse .

Returns The split _type

Return type str

Example

```
>>> from csamtpy.utils  import func_utils as func
>>> path =  os.path.join(os.environ["pyCSAMT"],
                'csamtpy','data', K6.stn)
>>> with open (path, 'r', encoding='utf8') as f :
...                     data= f.readlines()
>>>  print(func.stn_check_split_type(data_lines=data))
```

csamtpy.utils.func_utils.**straighten_out_list**(*main_list*, *list_to_straigh*)

Parameters

*** main_list** [list]

list of which the data must absolutely appear into the straighen list. in our case , it is the station list : a list of offset

- **list_to_straigh** [list] list contain the data (offset calculated , the depth and the resistivity (log10)),

Returns

- **list**

the straighen list. some offset have been replaced by the offsets which are not in the main_list whithout change the lengh of the straighen list.

Example

```
>>>  import numpy as np
>>>  np.random.seed(14)
>>>  ss=np.random.randn(10)*12
>>>  ss=ss.tolist()
>>>  ss=[round(float(jj),4) for jj in ss]
>>>  ss.sort()
>>>  red=np.random.randn(7)*12
>>>  red=red.tolist()
>>>  test=[19, 15.012, 5.5821, 0.7234,3.1,
...       0.7919, 3.445, 4.7398, 5.1, 10.8, 15.51,21]
>>>  main=[20., 0.7234, 5, 3.445, 15.51,10.7, 3,5.1]
>>>  test.sort()
>>>  main.sort()
>>>  red=[round(float(ss),1) for ss in red]
>>>  print(test)
```

(continues on next page)

(continued from previous page)

```
>>>  print(main)
>>>  sos=straighten_out_list (main_list=main ,
...                           list_to_straigh=test)
>>>  print("sos:
```

“,sos)

csamtpy.utils.func_utils.**take_firstValue_offDepth**(*data_array*, *filter_order=1*)

Parameters

*** data_array** [np.array]

array of the data .

- **filter_order** [int , optional] the column you want to filter. The default is 1.

Returns

array_like return array of the data filtered.

Example

```
>>>  import numpy as np
>>>  list8=[[4,2,0.1],[8,2,0.7],[10,1,0.18],[4,3,0.1],
...        [7,2,1.2],[10,3,0.5],[10,1,0.5],[8.2,0,1.9],
...        [10,7,0.5],[10,1,0.5],[2,0,1.4],[5,4,0.5],
...        [10,2,0.7],[7,2,1.078],[10,2,3.5],[10,8,1.9]]
>>>  test=np.array(list8)
>>>   print(np_test)
>>>  ss=take_firstValue_offDepth(data_array =np_test,␣
↪filter_order=1)
>>>   ss=averageData(np_array=np_test,filter_order=1,
>>>                  axis_average=0, astype="int")
>>> print(ss)
```

csamtpy.utils.func_utils.**transfer_array_**(*data*, *index_key*, *start_value_depth*, *end_value_depth*, *column_order_selection=0*, *axis=0*)

Parameters

*** data** [dict]

Dictionnary of numpy ndarray .

- **index_key** [float] key of the dictionnary . Must be a number of the first column of offset .
- **start_value_depth** [float] If the depth is not reach must add depth of the closest point. give the start value which match to the maxi depth of the data : The *default* is -214.
- **end_value_depth** [float] Maximum depth of the survey. The default is -904.
- **column_order_selection** [int,] the index of depth column. The default is 0.
- **axis** [int , optional] numpy.ndarray axis . The default is 0.

Returns

numpy.ndarray return the array data we want to top to .

Example

```
>>> import numpy as np
>>> sos=abs(np.random.randn(4,3)*4)
>>> sos2=abs(np.random.randn(4,3)*10.8)
>>>  print(sos2)
>>> sis1=sort_array_data(data=sos,sort_order =1,
...                      concatenate=False, concat_axis_
↪order=0)
>>> sis2=sort_array_data(data=sos2,sort_order =1,
...                      concatenate=False, concat_axis_
↪order=0)
>>> dico={"18.4":sis1,
...       "21.4":sis2}
>>> test=transfer_array_(data=dico, index_key=11.4,
...                        start_value_depth=-14, end_
↪value_depth=23,
...                        column_order_selection=1)
>>> print("sis1:",sis1)
>>> print("sis2:",sis2)
>>> print("Finaltest", test)
```

6.4 Module Gis-tools

6.5 Module Infos

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

synopsis Module contains various parameters of files handling and glossary of some technical words.

Created on Sat Dec 12 16:21:10 2020

@author:KouaoLaurent alias @Daniel03

class csamtpy.etc.infos.**_sensitive**

Note: *sensitive class* . Please keep carefully the indices like it's ranged. Better Ways to avoid redundancy in code during the program design. Core of parser of each files except files from module-geodrill. Aims are: 1. to check file . it is was the right file provided or not 2. to write file . to Zonge Avg_format or J-Format or EDI -format and else. 3. to compute value . Indice are used for computation , set and get specific value.

Methods

validate_avg(avg_data_lines)	Core function to validate avg file .
which_file([filename, deep])	Which file is class method .

static validate_avg(*avg_data_lines*)
Core function to validate avg file .

Parameters **avg_data_lines** (*list*) – list of avgfile

Returns 'yesAST' or 'yes' where 'yesAST' is Astatic file and 'yes' is plainty avg file (the main file)

Return type str

Returns item, spliting the headAvg components strutured by file.

Return type list

Example

```
>>> from csamtpy.etc.infos.Infos inmport _sensistive as SB
>>> path =  os.path.join(os.environ["pyCSAMT"],
...                                  'csamtpy','data', LCS01_2_
↪to_1.avg)
... with open (path, 'r', encoding ='utf8') as f :
...   datalines = f.readlines()
... ss =SB._sensitive.validate_avg(avg_data_lines=datalines)
```

classmethod which_file(*filename=None*, *deep=True*)
Which file is class method . List of files are the typical files able to read by pyCSAMT softwares. Sensitive class method.

Parameters

****filename :str****

corresponding file to read , pathLike

deep [bool ,] control reading : False for just control the extension file , not opening file . True control in deeper file and

find which file were inputted.

Returns

str

FileType could be [*avg* | *j* | *edi* | *resp* | *mesh* | *occamdat* | *stn* | *model* | *iter* | *logfile* | *startup*]

List of files read by pyCSAMT :

Example

```
>>> files = ['K1_exp.bln','LCS01.avg' ,'LCS01_2_to_1.
↪avg', 'K1.stn',
...             'csi000.dat','csa250.edi','LogFile.
↪logfile',
...                   'Occam2DMesh','Occam2DModel',
↪'OccamDataFile.dat',
...             'S00_ss.edi', 'Startup','RESP13.resp',
...                   'ITER02.iter']
>>>  for ii in files :
>>>      path =  os.path.join(os.environ["pyCSAMT"],
...                                      'csamtpy','data',
↪ ii)
...       try :
...         print(_sensitive.which_file(path,
↪deep=True))
...       except :pass
```

class csamtpy.etc.infos.**notion**
Singular class to explain CSAMT technical word in details. Also usefull for user to have info about any scientific context is does Know. Just call the word directly in warnings to give an overview of why error occurs. It like a short documentation using pyCSAMT software. Used everywhere in the script .

class csamtpy.etc.infos.**suit**
Singular class to easy manipulate word. used everywhere in the script to avoid redondancy.

6.6 Module Plot-utils

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Tue Dec 29 19:18:44 2020

@author: @Daniel03

csamtpy.utils.plot_utils.**annotate_tip**(*layer_thickness*, *layer_names*)
A tip to group text with the same resistivities one layer when the layer are successively the same .

Parameters

- **layer_thickness** (array_like llist) – thickness of layer
- **layer_names** (*list or array_like*) – names of layers , geological structures names

Returns v , layer thickness in depth

Return type array_like

Returns ni , list : name of layer

Return type array_like

Example

```
>>> from csamtpy.utils import plot_utils as punc
>>> rocks = ['Massive sulfide', 'Igneous rocks', 'Igneous rocks',
↪'Igneous rocks',
    'Igneous rocks', 'Igneous rocks', 'Igneous rocks', 'Igneous␣
↪rocks',
    'Massive sulfide', 'Igneous rocks', 'Igneous rocks', 'Igneous␣
↪rocks']
>>> resprops = [0.0, 49.0,69.0,89.0,109.0,129.0,
                149.0,179.0,249.0,699.0,799.0,899.0]
>>> thickness, lnames = punc.annotate_tip(layer_thickness=resprops,␣
↪layer_names=rocks)
>>>print(thickness)
>>> print(lnames)
>>> v, ni
...  [24.5, 149.0, 474.0, 799.0]
...  ['Massive sulfide', 'Igneous rocks', 'Massive sulfide',
↪'Igneous rocks']
```

csamtpy.utils.plot_utils.**average_rho_in_deep**(*dep_array*, *rho_array*, *step_descent*)

function to average rho in deep according to the value provided . In fact averaged rho in shorter depth distance allow us to understand the conductive zone. in approximately. The most conductive zone is detected as the zone with lower resistivities values . But fixing values as averaged rho , can build a specific strata that could match this zone .

Parameters

*** dep_array** [array_like]

the imaged depth (doi)

- **rho_array: array_like** resistivity array
- **step_descent** [float] value to step descent

Returns

array_like rho average for each station # dep_averaged for each station

Example

```
>>> import numpy as np
>>> from csamtpy.utils import plot_utils as punc
>>> pseudo_depth=np.array([  0. ,  6. , 13. , 20.  ,29.␣
↪, 39.,  49. ,
...                         59. , 69. , 89., 109. ,129.,␣
↪149. ,179.,
...                    209. ,249. ,289., 339., 399., 459.␣
↪,529. ,609.,
...                    699., 799., 899., 999.])
>>> pseudo_depth = np.arange(0, 1220, 20)
>>> rho = np.random.randn(len(pseudo_depth))
>>> rho_aver, dep_aver= average_rho_in_deep(dep_
↪array=pseudo_depth,
...                                         rho_
↪array=rho,
```

(continues on next page)

(continued from previous page)

```
...                                                 step_
↪descent=20.)
>>> rho_2,dep_2 =   average_rho_in_deeper (dep_array=␣
↪pseudo_depth,
...                                                  rho_
↪array=rho,
...                                                 step_
↪descent=1000)
>>>  print(pseudo_depth)
```

csamtpy.utils.plot_utils.**average_rho_in_deeper**(*dep_array*, *rho_array*, *step_descent*)

function to average rho in deep according to the value provided . In fact averaged rho in shorter depth distance allow us to understand the conductive zone. in approximately. The most conductive zone is detected as the zone with lower resistivities values . But fixing values as averaged rho , can build a specific strata that could match this zone .

Parameters

*** dep_array** [array_like]

the imaged depth (doi)

- **rho_array: array_like** resistivity array
- **step_descent** [float] value to step descent

Returns

array_like rho average for each station # dep_averaged for each station

Example

```
>>> import numpy as np
>>> from csamtpy.utils import plot_utils as punc
>>> pseudo_depth=np.array([  0. ,   6. , 13. , 20.  ,29.␣
↪, 39.,  49. ,
...                         59. , 69. , 89., 109. ,129.,␣
↪149. ,179.,
...                     209. ,249. ,289., 339., 399., 459.␣
↪,529. ,609.,
...                     699., 799., 899., 999.])
>>> pseudo_depth = np.arange(0, 1220, 20)
>>> rho = np.random.randn(len(pseudo_depth))
>>> rho_aver, dep_aver= average_rho_in_deep(dep_
↪array=pseudo_depth,
...                                                  rho_
↪array=rho,
...                                                 step_
↪descent=20.)
>>> rho_2,dep_2 =   average_rho_in_deeper (dep_array=␣
↪pseudo_depth,
...                                                  rho_
↪array=rho,
...                                                 step_
↪descent=1000)
>>>  print(pseudo_depth)
```

csamtpy.utils.plot_utils.**average_rho_with_locals_minmax**(*array*)

How to compute mean value between local maxima and local minima and keep locals minima and maxima value

on the final data

Parameters **array** (*array_like*) – data to compute the local minima and local maxima

Returns array mean with data local value averaged

Return type array_like

Example

```
>>> from csamtpy.utils import plot_utils as punc
>>> mean1= punc.average_rho_with_locals_minmax(tth[0])
>>> mean2= punc.average_rho_with_locals_minmax(tth[1])
>>> print(mean1)
>>> print(mean2)
```

csamtpy.utils.plot_utils.**build_new_station_id**(*station_id*, *new_station_name*)

Fonction to build new station id including station name provided , if the length provided doesnt match the length of station id

Parameters

- **id** (*station*) – new sites names
- **mess** (*str*) – message for debugging, Default is None

Example

```
>>> from csamtpy.utils import plot_utils as punc
>>> ts =[   28.,  200.,  400.,  600.,  800., 1000., 1200., 1400.,␣
↪1600., 1800., 1807.]
>>> sto = ['S{0}'.format(i) for i in range(7)]
>>> print(sto)
>>> stn, fu =punc.build_new_station_id(station_id = sto, new_
↪station_name =ts)
>>> print(stn, fu)
```

csamtpy.utils.plot_utils.**build_resistivity_barplot**(*depth_values*, *res_values*)

Allow to build bar plot resistivity function of investigation depth .

Parameters

*** depth_values** [aray_like]

model investigation depth

- **res_values** [array_like] model_resistivities at each depth values

Returns

d [array_like]

resistivity barplot depth .

r [array_like] specific structure resistivities

sumd [float] checker number that cover in fact the total depth. this numbe must absolutely match the total depth .

csamtpy.utils.plot_utils.**controle_delineate_curve**(*res_deline=None*, *phase_deline=None*)

fonction to controle delineate value given and return value ceilling .

Parameters

- **res_deline** (*float | int | list*) – resistivity value todelineate. unit of Res in *ohm.m*
- **phase_deline** (*float | int | list*) – phase value to delineate , unit of phase in degree

Returns delineate resistivity or phase values

Return type array_like

csamtpy.utils.plot_utils.**delineate_curve**(*dict_loc*, *value*, *atol=0.2*, *replace_value=nan*)
function to delineate value of rho and phase .

Parameters

- **dict_loc** (*dict*) – dictionnary composed of keys = stations id and values
- **value** (float |list) – value to delineate curve . for single value.
- **atol** (*float*) – tolerance parameter <=1 . Most the param is closest to 0 , most the selected data become severe.default is 0.2
- **replace_value** (*float or else*) – could be None or np.nan , Default is np.nan

Returns dict of delineate data

Return type dict

Example

```
>>> from csamtpy.utils import plot_utils as punc
>>> ts = np.array([0.1, 0.7, 2, 3, 8, 1000, 58,55, 85, 18])
>>> to =np.array([51, 78, 0.25, 188, 256, 7])
>>> tt ={'S00': ts, 'S01':to, 'S02': np.array([0, 5, 125, 789])}
>>> print(punc.delineate_curve(dict_loc = tt, value=[50,70], atol =␣
↪0.1))
```

csamtpy.utils.plot_utils.**delineate_sparseMatrix**(*dict_loc*, *delineate_dict*, *replace_value=nan*)
Build from delineate dict a matrix according to frequency length . value doesnt exist in the delineate dict will be repalce by replacevalue . Default is *nan*.

Parameters

- **delineate_dict** (*dict*) – delineation value
- **dict_loc** (*dict*) – dictionnary composed of keys = stations id and values
- **replace_value** (*float*) – value to replece other value like build a sparse matrix

Returns dit of sparse matrix

Return type dict

csamtpy.utils.plot_utils.**depth_of_investigation**(*doi*)
Depth of investigation converter

Parameters **doi** (*str | float*) – depth of investigation if value float is provided , it will considered as default units in meter

:returns doi:value in meter :rtype: float

csamtpy.utils.plot_utils.**find_closest_station**(*offset_indice*, *model_offsets*, *site_offsets*)
Get the indice of the closest offset

Parameters

*** offset_indice: int**

if the indix of the offset at the selected resistivity point.

- **model_offset: array_like** is a large band of x_nodes resistivities gnerated by mesh files
- **sites_offsets: array_like** the data set offset from Occam Data file

Returns

indexoff [int]

index of data offset

get_offs [float] value of the offset at that index

csamtpy.utils.plot_utils.**find_local_maxima_minima**(*array*)

function to find minimum local and maximum local on array

Parameters **data** (*array_like*) – value of array to find minima , maxima

Returns tuple of index of minima and maxima local index and array of minima maxima value

Return type array_like

Example

```
>>> from csamtpy.utils import func_utils as func
>>> from csamtpy.utils import plot_utils as punc
>>> ts =np.array([2, 3, 4, 7, 9, 0.25, 18, 28, 86, 10, 5])
>>> te=np.array([0.2, .3, 18, 1.6, 0.2, 0.6, 0.7, 0.8, 0.9, 1., 23.
↪])
>>> th =func.concat_array_from_list([ts, te], concat_axis=1)
.. tth =th.T
... print(tth)
... print(punc.find_local_maxima_minima(tth[0]))
```

csamtpy.utils.plot_utils.**find_path**(*path=None*, *ptol=0.7*)

check path and return filepath , edipath or jpath .

Parameters

- **path** (*str*) – full path to *edi*, *avag* or *j* file or directory
- **ptol** (*float*) – tolerance that given by the program to judge if the number of typical file [EDIlJ] to declare as path found is either "edipath" or "jpath" if none ,return None . less or equal to 1.

Returns specific path

Return type str

csamtpy.utils.plot_utils.**fmt_text**(*data_text*, *fmt='~'*, *leftspace=3*, *return_to_line=77*)

Allow to format report with data text , fm and leftspace

Parameters

- **data_text** (*str*) – a long text
- **fmt** (*str*) – type of underline text
- **leftspae** (*int*) – How many space do you want before starting wrinting report .
- **return_to_line** (*int*) – number of character to return to line

csamtpy.utils.plot_utils.**get_conductive_and_resistive_zone**(*data*, *site_names*, *purpose='groundwater'*, ***kws*)

function to get the probability of conductive and resistive zone . It is not absolutely True but give an overview of decison . It is not sufficient to declare that the zone is favorable for any drill , but just work with probability

Parameters

- **data** (*ndarray*) – resistivity datata of survey area
- **site_name** (*list*) – list of sites names
- **purpose** (*str*) – type of exploration , default is *groundwater*

Returns report of exploration area

Return type str

csamtpy.utils.plot_utils.**get_frequency_id**(*freq_array*, *frequency_id*)

function to get id of frequency . Frequency to plot

Parameters

- **freq_array** (*nd.array,1*) – array of frequency
- **frequency_id** (*list or float*) – frequency to plot .

Returns new close frequency id

Return type floatllist

csamtpy.utils.plot_utils.**get_station_id_input_resistivities**(*station_rho_value*, *number_of_layer=None*)

Get a special station input resistivities is much benefit and more close to reality of plot . Indeeed , it take only the maximum value of resistivities below the site and the minimum , then cut out 7 Considering all the model data and choose the max resistivities and the minim resistivities to build automatic resistivities whom COULD match the deth is less sure . Geeting a input resitivities to aplom the site , give a merly and better interpretation.

Parameters

- **station_rho_value** (*array_like*) – value of resistivities under the site thin the maximum depth
- **number_of_layer** (*int*) – number of layer to top to bottom.

csamtpy.utils.plot_utils.**get_stationid**(*stations*, *station_id*)

Tip to get station id from user by input either integer of station name .

Parameters

- **stations** (*list*) – list of stations known
- **station_id** (*list, str, or int*) – staion expect to plot.

Returns constructed list for plotting

Return type array_like

Example

```
>>> from csamtpy.utils import plot_utils as punc
>>> teslist = ['S{0:02}'.format(ii) for ii in range(23)]
>>> ss = punc.get_stationid (stations=teslist ,  station_id=('S04',
↪13))
>>> print(ss)
```

csamtpy.utils.plot_utils.**resetting_colorbar_bound**(*cbmax*, *cbmin*, *number_of_ticks=5*, *logscale=False*)

Function to reset colorbar ticks more easy to read

Parameters

- **cbmax** (*float*) – value maximum of colorbar
- **cbmin** (*float minimum data value*) – minimum data value
- **number_of_ticks** (*int*) – number of ticks should be located on the color bar . Default is 5.
- **logscale** (*bool*) – set to True if your data are lograith data .

Returns array of color bar ticks value.

Return type array_like

csamtpy.utils.plot_utils.**resetting_ticks**(*get_xyticks*, *number_of_ticks=None*)

resetting xyticks modulo , 100

Parameters

- **get_xyticks** (*list*) – xyticks list , use to ax.get_xlyticks()
- **number_of_ticks** (*int*) – maybe the number of ticks on x or y axis

Returns a new_list or ndarray

Return type list or array_like

csamtpy.utils.plot_utils.**share_props_for_each_plot**(*number_of_plot=3*, ***kwargs*)

Function to set properties for each plot. Easy to customize line and markers. Function can add other properties which are not in kwargs.keys(). It will set according the number of subplotsplots we assume subplot are define on one columns

Parameters **number_of_plot** (*int*) – number of subsplot you want show.

Returns dictionarry of labels and properties.

Return type dict

csamtpy.utils.plot_utils.**slice_csamt_matrix**(*block_matrix*, *station_offsets*, *depth_offsets*, *offset_MinMax=(0, 1000)*, *doi='2000m'*)

Using Wannamaker FE elements mesh to define rho matrix blocks , need after inversion to slice the model resistivity according the offset and depth we need . This function is easy tool to slice matrix and to keep the part we need . station offset , depth and model resistivity

Parameters

* **block_matrix** [ndarray(station_offsets.shape[0],]

matrix of station depth Resistivity model depth_offsets.shape[0])

- **depth_offset** [array_like] depth of investigation after generating by mesh file :>z_nodes .
- **station_offsets** [array_like] station _offsets : offset generate by mesh_file :>x_nodes .
- **offset_MinMax** [tuple]

  the interval of data to keep . eg if station location start by 0 : off[0] = min and off[-1]=max (min, max):–> index 0 : minimum value of station location –>index 1 : maximum value of station location *default* is (0,1000)

- **doi** [str , float] investigation depth , migth be [m|km]. If value is provided is float number , it might take value as a default unit 'meter'. i.e : 1000="1000m"

Returns

tuple new sliced station offset , new sliced depth offset , new_matrix block ,

csamtpy.utils.plot_utils.**slice_matrix**(*base_matrix*, *freq_array*, *doi=2000*)

Function get a matrix and give new matrice slice from limit value

Parameters

- **base_matrix** (*ndarray*) – arrays (yaxis lenghth, station_length)
- **freq_array** (*ndarray,1*) – frequency array range
- **doi** – expect to be the depth of investigation from which data muts be selected in *m* or *km*

:type doi:float

Returns matrix sliced according to doi

Return type ndarray

CHAPTER

SEVEN

PACKAGE VISUALIZATION

7.1 Module Plot1D2D

This file is part of pyCSAMT.

pyCSAMT is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pyCSAMT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with pyCSAMT. If not, see <https://www.gnu.org/licenses/>.

Created on Mon Dec 28 14:28:06 2020

@author: KLaurent alias @Daniel03

class viewer.plot.**Plot1d**(***kwargs*)

plot 1d class Deal with all 1D plots.

Key Words	Description
fig_dpi	dots-per-inch resolution of the figure *default* is 300
fig_num	number of the figure instance *default* is 'Mesh'
fig_size	size of figure in inches (width, height) *default* is [5, 5]
fs	size of font of axis tick labels, axis labels are fs+2. *default* is 6
ls	['-' | '.' | ':'] line style of mesh lines *default* is '-'
marker	marker of stations *default* is r"$lacktriangledown$"
ms	size of marker in points. *default* is 5

Methods	Description
plot_topo_sep_azim	plot_topography , station separation and azimuth profile can plot individually or grouped by.
penetrated1D	skindepth plot. penetration depth at different frequencies
plot_static_correction	plot rho and rho corrected by different filter defalut filter is TMA.
plot_freqVSRhoPhase	Resistivity and phase plot.
plot_curves	plot data curves : specific for Zonge Engineering AVG file.
plot_RhoPhase errors	plot errors bar of resistivities in ohm.m and phase in degree.

Methods

penetrated1D([fn, profile_fn, ...])	Pentration1D depth : Show skin depth at selected frequencies .
plotRMS([fn, target, savefig])	Plot RMS .
plot_curves([fn, savefig, selected_stations])	Plot Zonge Engineering AVG file with different components E and H at differents frequencies.
plot_freqVSRhoPhase([fn, profile_fn, ...])	Method to plot apparent resistivity |phase vs frequency .
plot_multiStations([X, Y, path, profile_lines])	Plot multisations of site sof survey area
plot_static_correction(data_fn[, ...])	plot coorected apparent resistivities at different stations by reducing the problem of static shift .
plot_station_profile([fn, straighten_type, ...])	Method to plot original station profile and coordinate reajustment profiles.
plot_topo_sep_azim([fn, profile_fn, savefig])	Method to plot topographic , stations separation and azimuth profiles .

penetrated1D (*fn=None*, *profile_fn=None*, *selected_frequency=None*, ***kwargs*)

Pentration1D depth : Show skin depth at selected frequencies . for multiples frequencies , put argument *selected_frequency* on list. If frequency provided is not on the frequency range , it will be interpolated.

Parameters

- **fn** (*str*) – full path to [AVG|EDI|J] file
- **profile_fn** (*str*) – full path to *stn* station file . If user used EDI or J files , Dont need to add profile_file
- **selected_frequency** (*list*) – list , list of freauency want to see the penetration depth. must be on a list . i.e [8, 511,1024]

Note: browse to see others plot config.

Params	Default	Description
rename_station	list	Bring the station name . Be sure the length of station name you provided match the size of the data station name .
rotate_station	int	rotation station name . *Default* is 90 degree.
fs	float	can change the size of marker. *Default* is .7 : eg ms =9*fs
lw	float	change the linewdth
plot_grid	bool	add grid on your plot . Default is False

Example

```
>>> from viewer.plot import Plot1d
>>> path =  os.path.join(os.environ["pyCSAMT"],
...              'csamtpy','data', file_1)
>>> plot_1d_obj =Plot1d()
... plot1d_depth = plot_1d_obj.penetrated1D(fn =path ,
...                    profile_fn= os.path.join(
```

(continues on next page)

(continued from previous page)

```
...                          os.path.dirname(path), 'K1.stn'),
...                      selected_frequency =511)
```

plotRMS (*fn=None*, *target=1.0*, *savefig=None*, ***kwargs*)

Plot RMS . If occamlogfile is not available , set rms value , iteration value at each rms and|or roughness.

Parameters

*** fn** [str]

full path to occam2D logfile

- **savefig** [str] full directory to save fig
- **target** [float] target supposed RMS to reach . Default is 1.

.. note:: If occam2d logfile is availbale , dont need other parameters , except the path "fn" and as possible the "target".

========== ========== ==

Params Type Description

========== ========== ==

rms array_like RootMeanSquare array .

iteration int number of interation reached . iteration starts from "0". so we will add the number provided plus 1.

roughness array_like deGootHeldlin roughness parameters. number of params =Num(RMS)-1 . so we excluded the starting R.M.S

target float RMS target weexpected to reach . Default is 1.0

========== ========== ==

plot_curves (*fn=None*, *savefig=None*, *selected_stations=1*, ***kws*)

Plot Zonge Engineering AVG file with different components E and H at differents frequencies.

Parameters

- **fn** (*str*) – full path to Zonge Engineering file
- **profile_fn** (*str*) – full path to profile file .
- **savefig** (*str*) – path to figure plot

Params	Default	Description
fs	float	can change the size of marker. *Default is .7 : eg ms =9*fs
lw	float	change the linewdth
error_bar	bool	set to false to let invisible. Default is True

:Example :

```
>>> from viewer.plot import Plot1d
>>>path =  os.path.join(os.environ["pyCSAMT"],
...           'csamtpy','data', file_1)
>>> plot_1d_obj =Plot1d()
>>> plotcurves = plot_1d_obj .plot_curves(fn = path,
                                          selected_stations=[1,10, 20],
                                            error_bar=True)
```

plot_freqVSRhoPhase(*fn=None*, *profile_fn=None*, *station_id=1*, *rename_stations=None*, ***kwargs*)

Method to plot apparent resistivity |phase vs frequency .

Parameters

- **fn** (*str*) – full path to [AVG|EDI|J] file
- **profile_fn** (*str*) – full path to station profile .

Note: if user use drectly *AVG data must provide station profile '*.stn'

Others params	Default	Description
station_id	str or int	plot the name of station if string is povided make be sure that the station name is on the station list eg : station_id = 1 means plot S00 station =[1,13] means plot >S00,S12station [S05, 7, 8] – [S05, S06, S07]
rename_stations	list	bring the station name . Be sure the length of station name you provided match the data station name
show_errorbar	bool	if True , see errobar plot. *Default* is False.

plot_multiStations(*X=None*, *Y=None*, *path=None*, *profile_lines=None*, ***kwargs*)

Plot multisations of site sof survey area

Parameters

*** path** [str]

full path to station profile path . In the case where Zonge avg file is provided , use *stn* profile files. Group all *stn* file on a folder will call automatically

- **profile_lines** [list] name of profile lines . if profile lines is NOne will tale all *stn* profiles in the path directory
- **X** [list]

  list of arrays array of X coordinates values for each survey line. Can be easting or Northing

- **Y: list**

  list of arrays of Y coordinates values [can be easting] or northing

.. note:: `X` and `Y` MUST be the same length

plot_static_correction(*data_fn*, *profile_fn=None*, *frequency_id=1*, *ADD_FILTER='tma'*, ***kwargs*)

plot coorected apparent resistivities at different stations by reducing the problem of static shift .

Parameters

- **data_fn** (*str*) – full path to file , can be [AVG|EDI|J] files
- **profile_fn** (*str*) – pathLike full path to Zonge Engeneering *.station file .

Note: If user provide raw Zonge AVG file , may also add profile file (*.stn)

params	Default	Description
frequency_id	str or int	plot the filtered frequency, eg frequency_id = 1023 means plot uncorrected rho and static rho at that frequency . set on list to plot multiple frequency [8,1101].
ADD_FILTER	str	name of filter to apply . TMA Trimming moving average AMA Adaptative moving average FLMA Fixed Length moving average

Example

```
>>> from from viewer.plot import Plot1d
>>> path =  os.path.join(os.environ["pyCSAMT"],
...              'csamtpy','data', file_1)
>>> plot_1d_obj =Plot1d()
... plot_1d_obj.plot_static_correction(data_fn =path ,
                                       profile_fn= os.path.join(
                                        os.path.dirname(path), 'K1.
↪stn'),
                                       frequency_id =1023)
```

plot_station_profile(*fn=None*, *straighten_type='classic'*, *reajust_coordinates=(0, 0)*, *savefig=None*, ***kwargs*)

Method to plot original station profile and coordinate reajustment profiles. Deal with Zonge AVG file .

Parameters

*** fn** [str]

full path to profile station file of Zonge Engineering station profile file . format egal to *.stn

- **straighten_type** [str] type of straingther profile it may be *classic*, *equisistant* or *distord Default* is 'classic'
- **reajust_coordinates** [list] list of float x, y values

:Example:

```
>>> path =  os.path.join(os.environ["pyCSAMT"],
...           'csamtpy','data', 'avg', 'K1.stn')
>>> plot_1d_obj= Plot1d()
>>> plot_1d_obj.plot_station_profile(fn = path)
```

plot_topo_sep_azim(*fn=None*, *profile_fn=None*, *savefig=None*, ***kwargs*)

Method to plot topographic , stations separation and azimuth profiles . User can add station_names and and set _it to let the program to plot on the corresponding figure. He can alse force the program to plot its dipole length otherwise the program will compute it automatically. If "set_station_name" is False , No Name of station will be visible. User has the possibility to plot one by one figure or all by using a

"*" symbol or 123. To polt one figure , it may use keyword argument "plot" following the king ['topo', azimuth', 'sep'] or integer 1|2|3.Method is flexible .User can customize the plot , marker and line as he wants by putting on list the matplotlib labels properties . The program uses the label properties on order to set configuration lines and other properties . Topography plot correspond to index 0 , stations-separation to index 1 and azimuth to index 2.To plot individually , User doesnt need to put properties on list. Programm will recognize and set the poperties provided according the figure He wants.

Parameters

*** fn** [str]

full path to [EDI|J|AVG] file.

- **profile_fn** [str] path to file may Zonge Engineering *.stn file
- **plot** [str] type of plot , default is '*' mean of three profile.
- **Station_Names: list** list of station names , User could provide. Default is None compute automatically
- **set_station_names** [bool] display the station name on figure axis . Default is False.
- **elevation** [(ndarray,1)] Array_like of elevation
- **station_pk** [array_like,] array_like station dipole center value.
- **savefig** [str] path to save figure.

========== ==========

Key Words Description

========== ==========

lw line width . *default* is 1.5

ls ['-' | '.' | ':'] line style of lines *default* is "['-', ':', '-.']" for 3 profiles.

marker marker of stations *default* is 'o'

ms size of marker in points. *default* is 6

color color of line .*Default* is 'k'

alpha Marker transparence .*Defaut* is .2

markerfacecolor facecolor or markers .*Default* is 'k'

markeredgecolor eadgecolor of markers . default is "['w','r"gray']"

xtick_label_rotation xtick rotation angle . default* is 45.

ytick_label_rotation ytick rotatoion angle .default * is 45

xtick_labelsize xtick label size .defalut* is 12.

ytick_labelsize ytick label size .defalut* is 12.

========== ==========

:Example:

```
>>> import os
>>> file_stn='K6.stn'
>>> path =  os.path.join(os.environ["pyCSAMT"],
...                              'csamtpy','data', file_stn)
...  plot_1d_obj= Plot1d()
.... plot_1d_obj.plot_topo_sep_azim(profile_fn= path , plot=
↪'*',
                                     set_station_names=True,
                                    dipole_length_
↪curve=False)
```

class viewer.plot.**Plot2d**(***kws*)

class to plot 2D map Deal with all 2D plots

keywords	Description
cb_pad	padding between axes edge and color bar
cb_shrink	percentage to shrink the color bar
climits	limits of the color scale for resistivity in log scale (min, max)
cmap	name of color map for resistivity values
fig_aspect	aspect ratio between width and height of resistivity image. 1 for equal axes
fig_dpi	resolution of figure in dots-per-inch
fig_num	number of figure instance
fig_size	size of figure in inches (width, height)
font_size	size of axes tick labels, axes labels is +2
grid	['both' | 'major' | 'minor' | None] string to tell the program to make a grid on the specified axes.
ms	size of station marker
plot_yn	['y' | 'n'] 'y' –> to plot on instantiation 'n' –> to not plot on instantiation
station_color	color of station marker
station_font_color	color station label
station_font_pad	padding between station label and marker
station_font_rotation	angle of station label in degrees 0 is horizontal
station_font_size	font size of station label
station_font_weight	font weight of station label
station_id	index to take station label from station name
station_marker	station marker. if inputing a LaTex marker be sure to input as r"LaTexMarker" otherwise might not plot properly
title	title of plot. If None then the name of the iteration file and containing folder will be the title with RMS and Roughness.
xlimits	limits of plot in x-direction in (km)
xminorticks	increment of minor ticks in x direction
xpad	padding in x-direction in km
ylimits	depth limits of plot positive down (km)
yminorticks	increment of minor ticks in y-direction
ypad	padding in negative y-direction (km)
yscale	['km' | 'm'] scale of plot, if 'm' everything will be scaled accordingly.

Methods

penetration2D([fn, profile_fn, savefig, doi])	Plot penetration 2D.
plot_Pseudolog([station_id, iter_fn, ...])	Build pseudodrill from the model resistivity .
plot_Response([data_fn, response_fn, mode])	Function to plot forward value , and residual value from Occam 2D
plot_occam2dModel([model_fn, iter_fn, ...])	Plotoccam Model form Occam Model class
pseudocrossResPhase(fn[, profile_fn, ...])	Plot Pseudocrossection of resistivity and phase.

penetration2D (*fn=None*, *profile_fn=None*, *savefig=None*, *doi='2km'*, ***kwargs*)

Plot penetration 2D.

Parameters

*** fn** [str]

full path to [EDI|AVG|J] files.

- **doi** [float] depth assumed to be imaged , default is 2000m For CSAMT , 2km is enought to have more info about near surface.
  - Default* unit is "m".
- **profile_fn** [str] full path to profile *stn file
- **savefig** [str] outdir

========== ========== ==========

Params Type Description

========== ========== ==========

plot_style str pcolormesh or imshow Default is **pcolormesh**

ms int markersize: *Default is .7 ,ie.9*fs

cm str mpl.colormap .Default is "Purples".

rename_station list can set new_stationname.

========== ========== ==========

:Example:

```
>>> path =  os.path.join(os.environ["pyCSAMT"],
...      'csamtpy','data', K1.AVG)
>>> plot2d_obj = plot2d()
>>> plot2d_obj.penetration2D(fn = path,
...                          profile_fn=os.path.join(
...                              os.path.dirname(path),'K1.stn
↪'),
...                           plot_style='imshow',  doi=
↪'10000m')
```

plot_Pseudolog (*station_id='S00'*, *iter_fn=None*, *mesh_fn=None*, *data_fn=None*, *iter2dat_fn=None*, *bln_fn=None*, *model_fn=None*, ***kwargs*)

Build pseudodrill from the model resistivity .

Deal with true value of ressitivity obtained during survey .In fact , How to input these values into our model to produce an accuracy underground map is the chalenge.Building pseudolog allow to know how layers are disposal in underground so to emphasize the large conductive zone in the case of groundwater exploration. It is combinaison with geophysic data especially inversion data with geological data. Actually the program deal with Occam 2D inverison file or Bo Yang (x,y,z) file. We will extend this program later with other external softares files extension. If user have a golder software installed on its computer , can use the files generated by the software and to produce 2D map so to compare both . Model map and detail-sequences map to see the difference Details sequences map is most closest to the reality . When step descent parameter is small ,the detail sequences trend to model map . So More geological values are, more the accuracy of detail sequences logs becomes. Geological data allow to harmonize the value of resistivity produced by our model so to force the pogramm to make a correlation between data from true layers and the model values.

Parameters **station_id** (*str, int*) – Number or the site id of the survey area number starts from 1 to the end .

Note: User caneither use Occam 2D inversions files to plot or BoYang (x, y, file)+ station location file (*bln) to plot if the two types of files are provided , program with give priority to Occam 2D inversion files.

Params	Type	Description
model_fn	str	full path to Occam model file .
iter_fn	str	full path to occam iteration file
data_fn	str	full path to occam_data file
doi	str	depth of investigation might be float or str like “1km” =1000
depth_scale	str	scale of imaging depth can be “km” or “m”. Default is”m”
step_descent	float	step to enforce the model resistivities to keep truth layers values as reference data . if step descentis egal to doi max, data looks like model at 99.99%.Step decent is function of depth and rho.
lc_AD_curves	tuple	customize line color of average curve and details sequneces logs eg : ((0.5, 0.8, 0.),’blue’)
default_unknow_lcolor	str	In the case the name of layer is notin our data base , customize the layer color . default is “(1.0, 1.0, 1.0)”.
default_unknow_lpatter	str	In the case the name of layer is not in our data base , customize the layer pattern default is “+.+.+.”

Note: constrained_electrical _properties_of_rocks param keeps the Truth layers resistivities as reference resistivities. If value is false will check in our data base to find the resistivities that match better the given resistivities of

the layers. *Default* is True.

Customize your plot using matplotlib properties.

Example

```
>>> from viewer.plot import Plot2d
>>> path =os.path.join(os.environ ['pyCSAMT'],
...                    'csamtpy', 'data', 'occam2D')
>>> plot2d_obj = Plot2d(station_label_rotation=None,
...                     show_grid=True,
...                     font_size =8,
```

(continues on next page)

(continued from previous page)

```
...                     lc='r',
...                     fig_size=[5,8],
...                     markerfacecolor='k',
...                     markeredgecolor='k')
>>> plot2d_obj.plot_Pseudolog( station_id=[43],
...                            input_resistivities=[300, 500,
...                                                 1000, 2000,
...                                                 4000, 6000],
...                            input_layers =['alluvium',
...                                           'amphibolite',
...                                           'altered rock',
...                                            'augen gneiss',
...                                            'granite'],
...                              mesh_fn=os.path.join(path,
...                                                   'Occam2DMesh
↪')
...                              iter_fn = os.path.join(path,
...                                                     'ITER17.
↪iter'),
...                              model_fn =os.path.join(path,
...
↪'Occam2DModel') ,
...                              data_fn =os.path.join(path,
...
↪'OccamDataFile.dat'),
...                              doi='1km',
...                              step_descent=200.,
...                              plot_style= 'pcolormesh')
```

plot_Response(*data_fn=None*, *response_fn=None*, *mode=None*, ***kws*)

Function to plot forward value , and residual value from Occam 2D

list of params are below :

Params	Type	Description
response_fn	str	full path to occam iteration file
data_fn	str	full path to occam_data file
doi	str	depth of investigation might be float or str like “1km” =1000
show_station_id	str	show station names

Example

```
>>> from viewer.plot import Plot2d
>>> pathresp =os.path.join(os.environ ['pyCSAMT'],
...                        'csamtpy', 'data', 'occam2D',
↪'RESP17.resp')
>>> path_data =os.path.join(os.environ ['pyCSAMT'],
...                         'csamtpy', 'data', 'occam2D',
↪'OccamDataFile.dat' )
>>> plot2d_obj = plot2d()
... plot2d_obj.plot_Response(data_fn =path_data , response_fn= ␣
↪pathresp )
```

plot_occam2dModel(*model_fn=None*, *iter_fn=None*, *mesh_fn=None*, *data_fn=None*, *doi=1000*, ***kwargs*)

Plotoccam Model form Occam Model class

Parameters **model_fn** (*str*) – full path to Occam 2Dmodel file

Params	Type	Description
iter_fn	str	full path to occam iteration file
mesh_fn	str	full path to mesh_fn file
data_fn	str	full path to occam_data file
doi	str	depth of investigation might be float or str like "1km" =1000
depth_scale	str	scale of imaging depth can be "km" or "m". *Default* is"m"

Example

```
>>> data='OccamDataFile.dat'
>>> mesh = 'Occam2DMesh'
>>> model = 'Occam2DModel'
>>> iter_='ITER17.iter'
>>> path =os.path.join(os.environ ['pyCSAMT'],
...                         'data', 'occam2D', mesh)
>>> plot2d_obj = plot2d()
>>> plot2d_obj.plot_occam2dModel(mesh_fn=path,
...                              iter_fn = os.path.join(
...                                  os.path.dirname(path), iter_
↪),
...                              model_fn =os.path.join(
...                                  os.path.dirname(path),␣
↪model) ,
...                              data_fn =os.path.join(
...                                  os.path.dirname(path), data␣
↪), doi='1km')
```

pseudocrossResPhase (*fn*, *profile_fn=None*, *savefig=None*, *plot_style=None*, ***kws*)

Plot Pseudocrossection of resistivity and phase.

Parameters

- **fn** (*str*) – full path to ['AVG', 'EDI', 'J'] file .
- **profile_fn** (*str*) – full path to profile station file in the case fn is *AVG.
- **savefig** (*str*) – path to save figure

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

c

g

v

z

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z