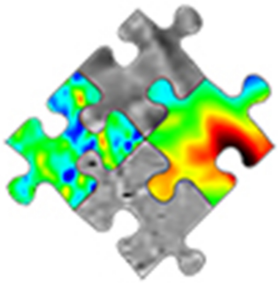


GeophPy Documentation

Release 0.20

Lionel Darras & Philippe Marty

September 10, 2015

1 Introduction

This project was initiated through cooperation between the french CNRS units UMR5133-Archeorient (FR) and UMR7619-Metis (FR)

2 Description

GeophPy is a python module which aims at building tools to display and process sub-surface geophysics data in the field of archaeology, geology, and other.

The main feature of this module is to build a "geophysic data set" composed by series of data values Z in the format (X,Y,Z) with (X,Y) being the point position of the geophysic Z value, to process or display maps of Z values.

This module has been developped to be used in a graphical user interface software, WuMapPy.

3 Features

- Building a data set from one or severals data files.

- Displaying geophysicals maps in 2D or 3D Dimensions.
- Processing data sets with geophysicals methods.
- Compatibility with Python 3.x

4 Installation

Download the zip file "GeophPy-vx.y" and unzip it.

You can install now GeophPy with this command:

```
$ python setup.py install
```

WuMapPy and GeophPy are using others python modules. If the installation of one of these modules failed on windows, you can install independently thes modules using this useful web site : http://www.lfd.uci.edu/~gohlke/pythonlibs/

5 Quick Start

```
>>> from geophpy.dataset import *
```

5.1 Opening files

You can open files indicating the column number (1...n) of the data set to be processed :

- ".dat" issued from Geometrics magnetometer G-858 (named 'ascii' format with ' ' as delimiter):

```
>>> dataset = DataSet.from_file(["test.dat"], format='ascii',
delimiter=' ')
```

- or XYZ files (files with column titles on the first line, and data values on the others, with X and Y in the first two columns, and Z1,...,Zn in the following columns, separated by a delimiter 't' ',' ';'...)

```
>>> dataset = DataSet.from_file(["test.xyz"], format='ascii',
delimiter='\t')
```

XYZ file example:

X	Y	Z
0	0	0.34
0	1	-0.21
0	2	2.45
...	...	...

You can easily obtain the list of the available file formats with the command :

```
>>> list = fileformat_getlist()
>>> print(list)
['ascii']
```

It is possible to build a data set from a concatenation of severals files of the same format :

- To open several selected files:

```
>>> dataset = DataSet.from_file(["abcde.dat","fghij.dat"],
              format='ascii', delimiter=' ', z_colnum = 5)
```

- To open all files ".dat" beginning by "file":

```
>>> dataset = DataSet.from_file(["file*.dat"], format='ascii',
              delimiter=' ', z_colnum = 5)
```

5.2 Checking files compatibility

Opening several files to build a data set needs to make sure that all files selected are in the same format.

It's possible to check it by reading the headers of each files:

```
>>> compatibility = True
>>> columns_nb = None
>>> for file in fileslist :
>>>    col_nb, rows = getlinesfrom_file(file)
>>>    if ((columns_nb != None) and (col_nb != columns_nb)) :
>>>        compatibility = False
>>>        break
>>>    else :
>>>        columns_nb = col_nb
```

5.3 Data set Description

A data set contains 3 objects:

- info.
- data.
- georef.

The "info" object contains informations about the data set:

- x_min = minimal x coordinate of the data set.
- x_max = maximal x coordinate of the data set.
- y_min = minimal y coordinate of the data set.
- y_max = maximal y coordinate of the data set.
- z_min = minimal z value of the data set.
- z_max = maximal z value of the data set.
- x_gridding_delta = delta between 2 x values in the interpolated image grid.
- y_gridding_delta = delta between 2 y values in the interpolated image grid.
- gridding_interpolation = interpolation name used for the building of the image grid.

The "data" object contains :

- fields = names of the fields (columns) : ['X', 'Y', 'Z']
- values = 2D array of raw values before interpolating (array with (x, y, z) values) : [[0, 0, 0.34], [0, 1, -0.21], [0, 2, 2.45], ...]
- z_image = 2D array of current gridded z data values : [[z(x0,y0), z(x1,y0), z(x2,y0), ...], [z(x0,y1), z(x1,y1), z(x2,y1), ...], ...]

(Note : the z_image structure is not built after openinf file, but by using gridding interpolation function)

The "georef" object contains :

- active = True if contains georeferencing informations, False by default.
- pt1 = Point number 1 Coordinates, in local and utm referencing (local_x, local_y, utm_easting, utm_northing, utm_zonenumber, utm_zoneletter).
- pt2 = Point number 2 Coordinates, in local and utm referencing (local_x, local_y, utm_easting, utm_northing, utm_zonenumber, utm_zoneletter).

5.4 Data set operating

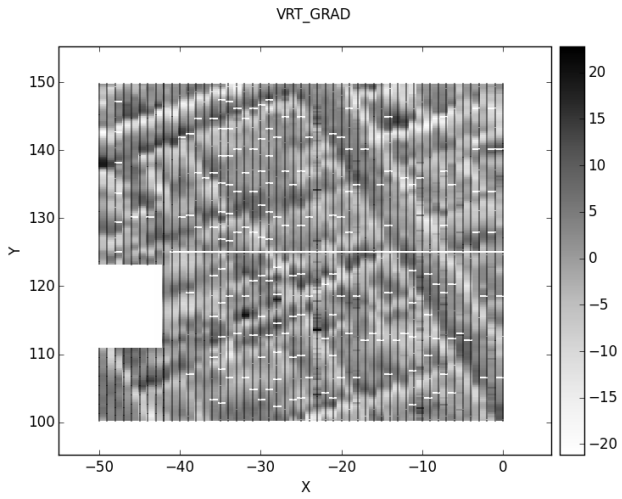
You can duplicate a data set to save, for example, a raw data set before processing it :

```
>>> rawdataset = dataset.copy()
```

After having opened a file, you can interpolate (or not) data, with severals gridding interpolation methods ('none', 'nearest', 'linear', 'cubic') to build z_image structure :

```
>>> dataset.interpolate(interpolation="none")
```

After doing this operation, it's easy to see on a same plot the map and the plots (on a grid or not if no gridding interpolation is selected):



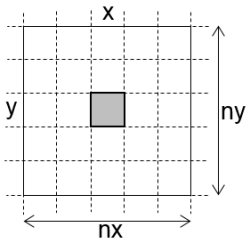
5.5 Data set processing

Peak filtering

Peak filtering allows to filtering values lower than 'setmin' and upper than 'setmax'. It can replace theses values by 'NaN' value (if setnan at True), or median value (if setmed at True (and setnan at False)).

Median filtering

Median filtering allows to filtering values lower or upper than a percentage of the meaning of the (nx*ny) grid of points around.



Festoon filtering

... To Be Developped ...

Regional trend filtering

... To Be Developped ...

Wallis filtering

... To Be Developped ...

Plough filtering

... To Be Developped ...

Constant destriping

... To Be Developped ...

Curve destriping

... To Be Developped ...

Logarithmic transformation

Introduction :

This processing is described here : “Enhancement of magnetic data by logarithmic transformation”, Bill Morris, Matt Pozza, Joe Boyce and Georges Leblanc - The Leading EDGE August 2001, Vol. 20,Num 8.

Input parameters :

- multfactor : it is a multiplying factor of the data (x5, x10, x20, x100).

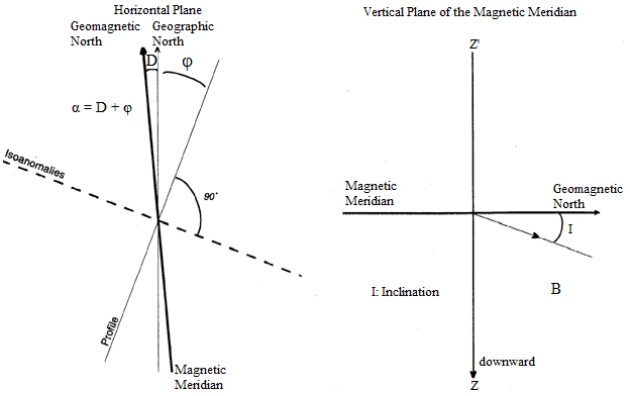
This factor depends on the precision of the data and is in the inverse order of this precision.

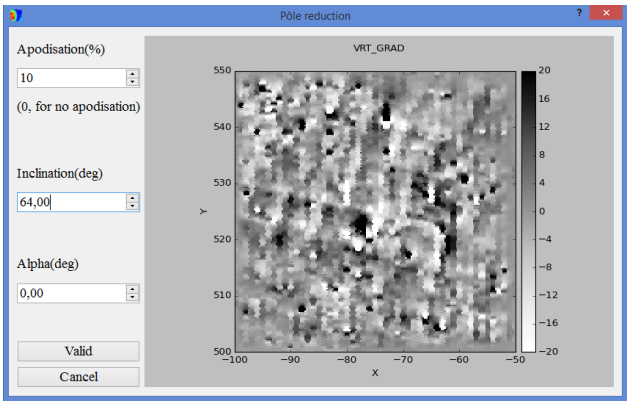
Pole Reduction

The reduction to the magnetic pole sets the data easy to read and be compared. This processing calculates the anomaly which be obtained for the full incination of the magnetic field. It uses a fast fourier series algorithm to work in the spectral range.

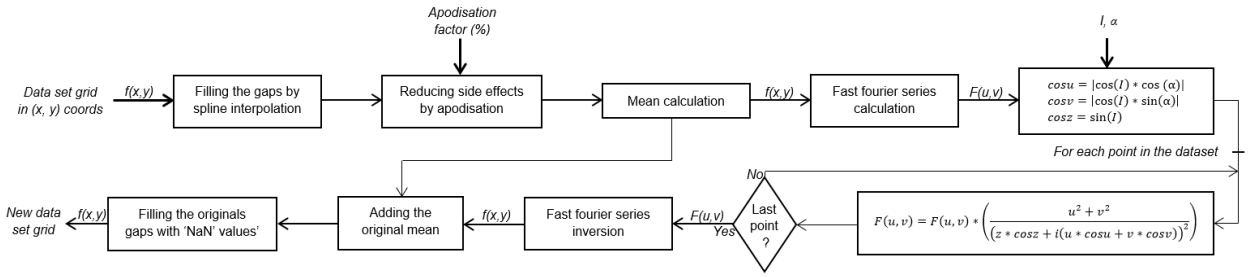
Input parameters :

- factor of apodisation, to reduce side effects.
- inclination angle, angle between the geomagnetic North and the magnetic field measured at the soil surface, in the vertical plane.
- alpha angle : it's the angle between geomagnetic North and profiles direction, in the horizontal plane.





Algorithm :



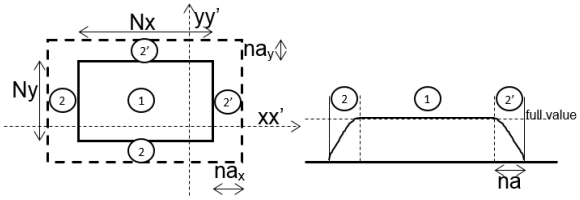
u is the spatial frequency corresponding at the x direction, and v is the spatial frequency corresponding at the y direction.

- Filling gaps :

To use the fast fourier algorithme, the data set don't have to contain gaps or 'NaN' values (Not A Number). If the data set grid is not over interpolated, the first step will be profile by profile to fill the gaps using a spline interpolation method.

Apodisation :

It is an operation to attenuate sides effects. The factor of apodisation (0, 5, 10, 15, 20 or 25%) precises the size to extend the data values zone. Values of this extension will be attenuated by a cosinus formula :



$na = N * \left(\frac{fap}{100}\right)$, with N = Nx or Ny, and fap the apodisation factor (in %)

$Zone\ 1 : value = full\ value$

$Zone\ 2 : value = full\ value * \cos\left(\frac{\pi}{2} * \frac{na - n}{na}\right), with\ na = nax\ or\ nay, and\ 0 \leq n \leq na$

$Zone\ 2' : value = full\ value * \cos\left(\frac{\pi}{2} * \frac{n}{na}\right), with\ na = nax\ or\ nay, and\ 0 \leq n \leq na$

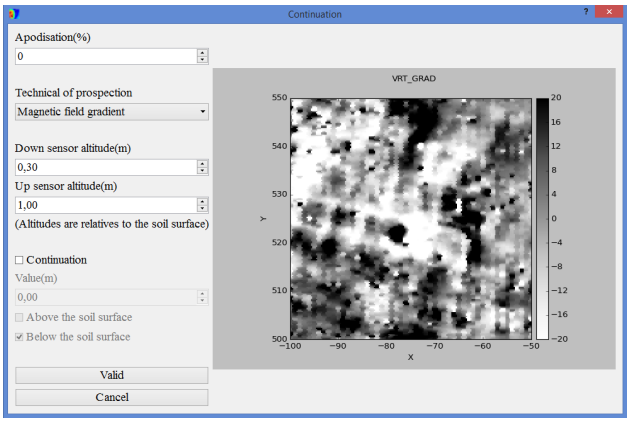
After processing, the size of the data values zone will become the same than before this apodisation.

Continuation

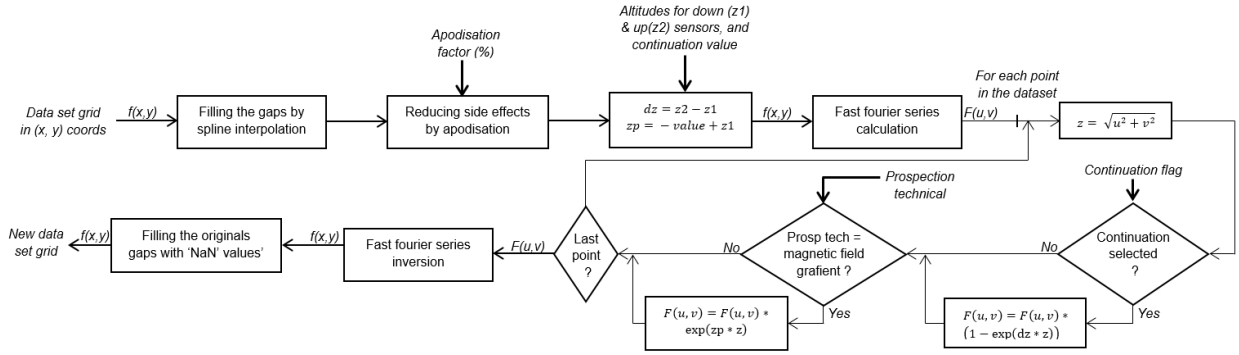
The downward continuation is a solution to reduce spread of anomalies and to correct coalescences calculating the anomaly if measures would be done at a lower level. The upward continuation allows to smooth data.

Input parameters :

- factor of apodisation, to reduce side effects.
- Technical of prospection ("Magnetic field", "Magnetic field gradient", Vertical component gradient").
- Sensors altitudes, relatives to th soil surface.
- Continuation value, above or below the soil surface.



Algorithm :



The method of filling gaps or apodisation to reduce side effects are the sames than used in the pole reduction function.

High level processing functions

The calling protocol of these functions are describeds in the end of this document, but about the detailed code of these processing functions, it's there :

geophpy.processing.general

DataSet Object general processing routines.

copyright Copyright 2014 Lionel Darras, Philippe Marty and contributors, see AUTHORS.

license GNU GPL v3.

`geophpy.processing.general.`**`peakfilt`**(*dataset*, *setmin=None*, *setmax=None*, *setmed=False*, *setnan=False*, *valfilt=False*)

cf. dataset.py

geophpy.processing.general.**medianfilt**(*dataset*, *nx=3*, *ny=3*, *percent=0*, *gap=0*, *valfilt=False*)

cf. dataset.py

geophpy.processing.general.**festoonfilt**(*dataset*, *method='Pearson'*, *shift=0*, *valfilt=False*)

cf. dataset.py

geophpy.processing.general.**regtrend**(*dataset*, *nx=3*, *ny=3*, *method='rel'*, *component='loc'*, *valfilt=False*)

cf. dataset.py

geophpy.processing.general.**ploughfilt**(*dataset*, *nx=3*, *ny=3*, *apod=0*, *angle=None*, *cutoff=None*, *valfilt=False*)

cf. dataset.py

geophpy.processing.general.**destripecon**(*dataset*, *Nprof=0*, *setmin=None*, *setmax=None*, *method='add'*, *valfilt=False*)

cf. dataset.py

geophpy.processing.magnetism

DataSet Object general magnetism processing routines.

copyright Copyright 2014 Lionel Darras, Philippe Marty and contributors, see AUTHORS.

license GNU GPL v3.

geophpy.processing.magnetism.**logtransform**(*dataset*, *multfactor=5*)

To transform the data in logarihtmics values

Parameters :

Dataset DataSet Object to do the treatment

Multfactor multiplying factor

geophpy.processing.magnetism.**polereduction**(*dataset*, *apod=0*, *inclineangle=65*, *alphaangle=0*)

To reduce at the magnetic pôle from a DataSet Object.

Parameters:

Dataset DataSet Object to do the treatment

Apod apodisation factor (%), to reduce bord effects

Inclinangle incline angle of the magnetic field

Alphaangle angle of magnetic field and y axe

geophpy.processing.magnetism.**continuation**(*dataset*, *prosptech*, *apod*, *downsensoraltitude*, *upsensoraltitude*, *continuationflag*, *continuationvalue*)

Continuation of the magnetic field

Parameters :

Dataset DataSet Object to do the treatment

Prosptech Prospection technical

Apod apodisation factor (%), to reduce bord effects

Downsensoraltitude Altitude of the down magnetic sensor

Upsensoraltitude Altitude of the up magnetic sensor

Continuationflag Continuation flag, False if not continuation

Continuationvalue Continuation altitude, greater than 0 if upper earth ground, lower than 0 else

geophpy.operation.general

DataSet Object general operations routines.

copyright Copyright 2014 Lionel Darras, Philippe Marty and contributors, see AUTHORS.

license GNU GPL v3.

geophpy.operation.general.**apodisation2d**(*val*, *apodisation_factor*)

2D apodisation, to reduce side effects

Parameters :

Val 2-Dimension array

Apodisation_factor apodisation factor in percent (0-25)

Returns :

- apodisation pixels number in x direction
- apodisation pixels number in y direction
- enlarged array after apodisation

5.6 Data set plotting

Plot type:

It is possible to plot the data set thanks to several plot types.

To see the plot types available , you can use :

```
>>> list = plottype_getlist()
>>> print(list)
['2D-SURFACE','2D-CONTOUR', '2D-CONTOURF']
```

Interpolation:

It is possible to plot the data set by choosing several interpolations for the surface display.

To get the plotting interpolations available , you must type:

```
>>> list = interpolation_getlist()
>>> print(list)
['nearest', 'bilinear', 'bicubic', 'spline16', 'sinc']
```

Color map:

To plot a data set, you have to choose a color map.

To see the color maps available, you type:

```
>>> cmaplist = colormap_getlist(creversed=False)
>>> print(cmaplist)
['Blues', 'BrBG', 'BuGn', 'BuPu', 'CMRmap', 'GnBu', 'Greens', 'Greys',
 'OrRd', 'Oranges', 'PRGn', 'PiYG', 'PuBu', 'PuOr', 'PuRd','Purples',
 'RdBu','RdGy','RdPu','RdYlBu', 'RdYlGn', 'Reds', 'Spectral', 'Wistia',
 'YlGn', 'YlGnBu', 'YlOrBr', 'YlOrRd', 'afmhot', 'autumn', 'binary',
 'bone', 'bwr', 'copper', 'gist_earth', 'gist_gray', 'gist_heat',
 'gist_yarg', 'gnuplot', 'gray', 'hot', 'hsv', 'jet', 'ocean', 'pink',
 'spectral', 'terrain']
```

Using the parameter creversed=True, you obtain the same number of color maps but with reversed colors, with a "_r" extension:

```
>>> for i in range (cmapnb):
>>>     colormap_plot(cmaplist[i-1], filename="CMAP_" +
        str(i) + ".PNG")
```

Examples :

or you can build figure and plot objects to display them in a new window:

```
>>> cm_fig = None
>>> first_time = True
>>> for cmapname in cmaplist:
>>>     cm_fig = colormap_plot(cmapname, fig=cm_fig)
>>>     if (first_time == True):
>>>         fig.show()
>>>         first_time = False
>>>     fig.draw()
```

Histogram:

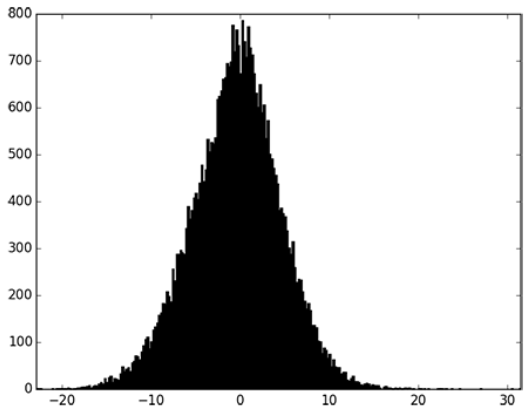
To adjust the limits of color map you must view the limits of the data set:

```
>>> zmin, zmax = dataset.histo_getlimits()
```

You can also plot the histogram curve :

```
>>> dataset.histo_plot("histo.PNG", zmin, zmax, dpi=100,
    transparent=True)
```

to obtain :



or you can build figure and plot objects to display them in a window:

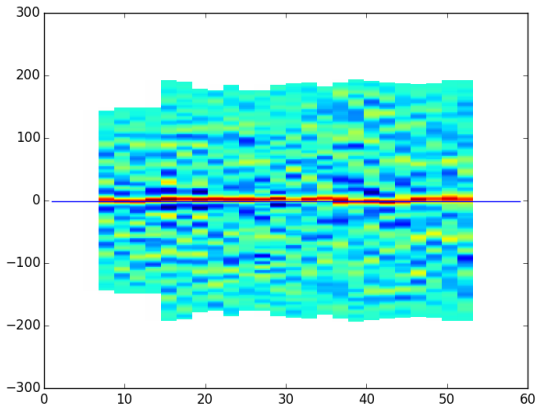
```
>>> h_fig = dataset.histo_plot()
```

Correlation map:

You can plot the correlation map of a dataset :

```
>>> dataset.correlation_plotmap("corrmap.PNG", dpi=100, transparent=True)
```

to obtain :



or you can build figure and plot objects to display them in a window:

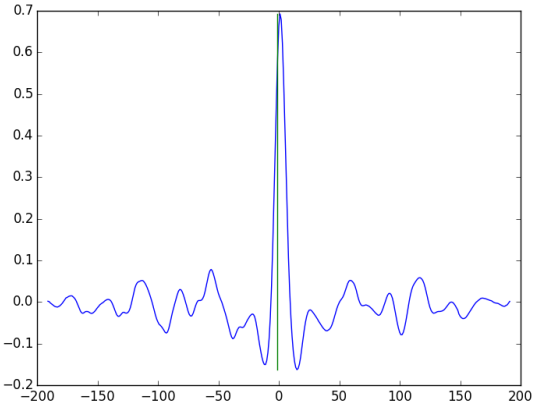
```
>>> h_fig = dataset.histo_plot()
```

Correlation sums:

You can plot the correlation sums of a dataset :

```
>>> dataset.correlation_plotsum("corrsum.PNG", dpi=100, transparent=True)
```

to obtain :



Map plotting:

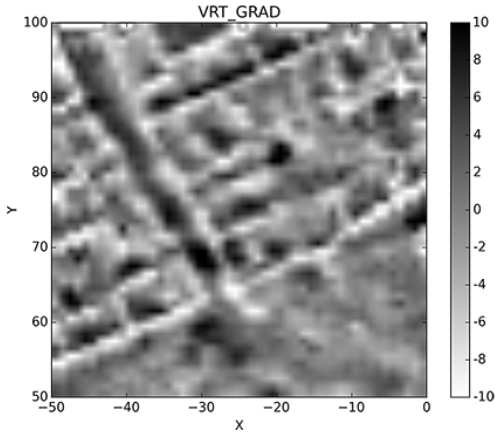
You can plot a data set using one of these plot types:

```
>>> dataset.plot('2D-SURFACE', 'gray_r', plot.PNG,
    interpolation='bilinear', transparent=True, dpi=400)
```

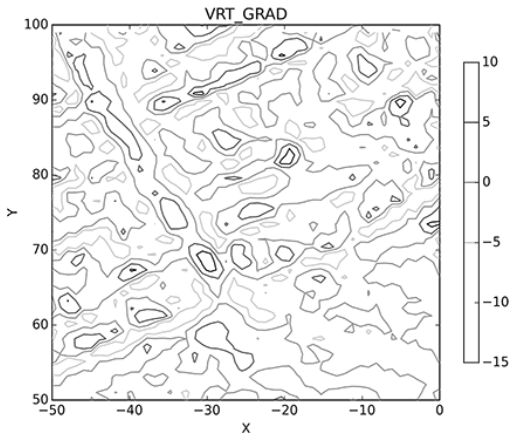
Examples :

Different plot types (‘2D-SURFACE’, ‘2D-CONTOUR’, ‘2D-CONTOURF’):

‘2D-SURFACE’ :

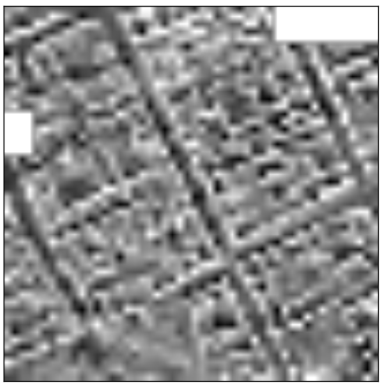


‘2D-CONTOUR’ :



Different interpolations for a “2D-SURFACE” plot type (‘bilinear’, ‘bicubic’):

With ‘bilinear’ interpolation:



With ‘bicubic’ interpolation:

It is possible not to display the color map and axis to import the picture in a SIG software.

5.7 Data Set Saving

You can save the data set in a file.

For the time being, it's only possible to save data in xyz files as it's described above:

```
>>> dataset.to_file(save.csv, format='xyz')
```

5.8 Geographic Positioning Set

It's possible to open one or several files with geographics coordinates of the geophysics prospection zone : These files can be ascii files with points positions, or shapefiles.

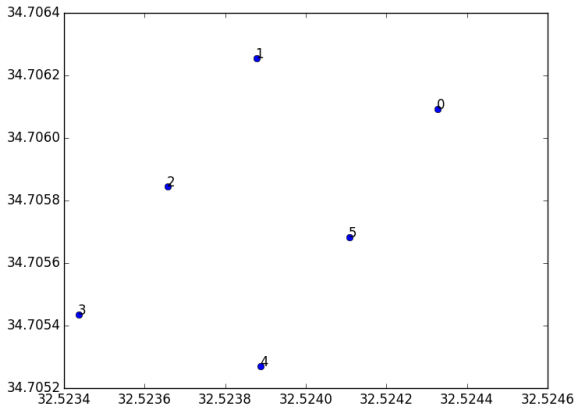
```
>>> from geoposset import *
>>> gpset = GeoPosSet.from_file(refsys='WGS84', type='shapefile',
    ["pt_topo"])
```

You can get so the numbers and list of points :

```
>>> list = gpset.points_getlist()
>>> print(list)
>>> [[0, 32.52, 34.70], [1, 32.52, 34.70]]  #with [num, x or lon, y or lat]
```

You can plot them :

```
>>> fig = gpset.plot()
>>> fig.show()
```



And converting a shapefile in a kml file :

```
>>> gpset.to_kml("shapefile.kml")
```

to view the points, lines, and surfaces described in the shapefile, on google earth :

It's possible to save theses points into an ascii file (.csv) composed by :

> Line 1 : "'WGS84"', or "'UTM', utm_zoneletter, utm_zonenumber"
>
> Others lines : point_number; longitude; latitude; X; Y

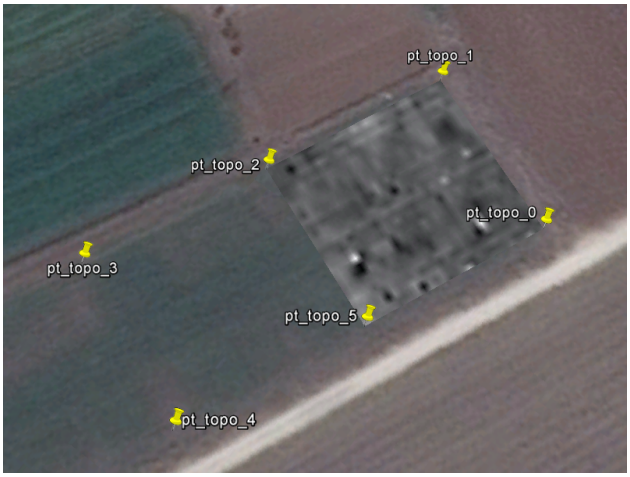
Example :

> WGS84 0;32.52432754649924;34.70609241062902;0.0;0.0
> 1;32.52387864354049;34.70625596577242;45.0;0.0 2;32.52365816268757;34.70584594601077;45.0;50.0
> 3;32.52343735426504;34.70543469612403;; # (X=None, Y=None) => point not referenced in local positioning

It is possible to georeference a data set with at less 4 points.

With the data set georeferenced, it is possible to export the data set in a kml file :

```
>>> dataset.to_kml('2D-SURFACE', 'gray_r', "prospection.kml",
    cmmin=-10, cmmax=10, dpi=600)
```



Exporting the data set as a raster in a SIG application (as ArcGis, QGis, Grass, ...) is possible with severals picture file format ('jpg', 'png', 'tiff') :

```
>>> dataset.to_raster('2D-SURFACE', 'gray_r', "prospection.png",
    cmmin=-10, cmmax=10, dpi=600)
```

A world file containing positioning informations of the raster is created ('jgw' for JPG, 'pgw' dor PNG, and 'tfw' for TIFF picture format) with :

Line 1: A: pixel size in the x-direction in map units/pixel

Line 2: D: rotation about y-axis

Line 3: B: rotation about x-axis

Line 4: E: pixel size in the y-direction in map units, almost always negative[3]

Line 5: C: x-coordinate of the center of the upper left pixel

Line 6: F: y-coordinate of the center of the upper left pixel

Example :

0.0062202177595

-0.0190627320737

0.0131914192417

0.00860610262817

660197.8178

3599813.97056

6 Using a Graphic User Interface

The name of the "Graphic User Interface" used has to be mentionned in the "matplotlibrc" file:

to use QT4 GUI :

backend : Qt4Agg

Note : to use QtAgg with PySide module, add :

backend.qt4 : PySide

or to use Tkinter GUI:

backend : TkAgg

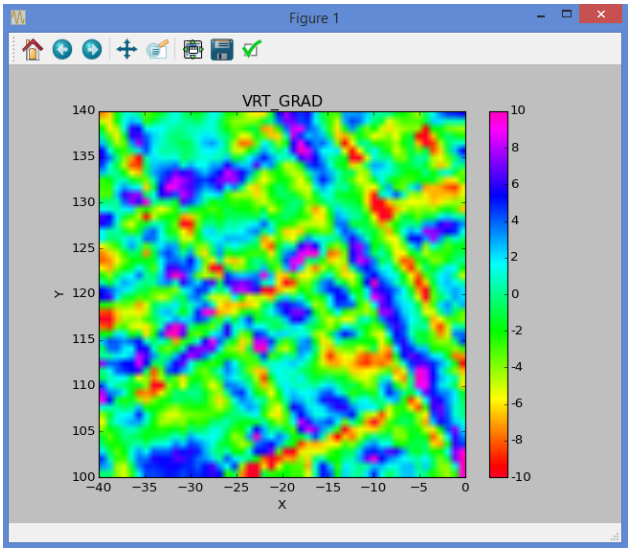
Note: *in Windows environment, this file is in the "C:\PythonXY\Lib\site-packages\matplotlib\mpl-data" directory.

*in Linux environment, this file is in the "/etc" directory.

With QT4Agg GUI, you can plot data in a windows :

```
>>> from geophpy.dataset import *
>>> success, dataset = DataSet.from_file("DE11.dat", delimiter=' ',
    z_colnum=5)
>>> if (success = True):
>>>     fig, cmap = dataset.plot('2D-SURFACE', 'gist_rainbow',
       dpi=600, axisdisplay=True, cmapdisplay=True, cmmin=-10, cmmax=10)
>>>     fig.show()
```

to obtain :



You can also plot data in a windows with several color maps :

```
>>> from geophpy.dataset import *
>>>                 # to get the list of color maps availables
>>> list = colormap_getlist()
>>> first = True                                # first plot
>>> fig = None                                  # no previous figure
>>> cmap = None                                 # no previous color map
>>> success, dataset = DataSet.from_file("DE11.dat", delimiter=' ',
    z_colnum=5)
>>> if (success == True):                       # if file opened
>>>                 # for each color map name in the list
>>>    for colormapname in list :
>>>        fig, cmap = dataset.plot('2D-SURFACE', 'gist_rainbow',
          dpi=600, axisdisplay=True, cmapdisplay=True, cmmin=-10,
          cmmax=10)
>>>        if (first == True):                   # if first plot
>>>           fig.show()                         # displays figure windows
>>>           first = False                      # one time only
>>>                 # updates the plot in the figure windows
>>>        p.draw()
>>>                 # removes it to display the next
>>>        cmap.remove()
>>>                 # waits 3 seconds before display the plot
>>>                 # with the next color map
>>>        time.sleep(3)
```

7 High level API

7.1 GeophPy.dataset

DataSet Object constructor and methods.

copyright Copyright 2014 Lionel Darras, Philippe Marty and contributors, see AUTHORS.

license GNU GPL v3.

geophpy.dataset.**getlinesfrom_file**(*filename*, *fileformat=None*, *delimiter='\t'*, *skipinitialspace=True*, *skiprowsnb=0*, *rowsnb=1*)

Reads lines in a file.

Parameters :

Fileformat file format

Filename file name with extension to read, "test.dat" for example.

Delimiter delimiter between fields, tabulation by default.

Skipinitialspace if True, considers severals delimiters as only one : " " as ' '.

Skiprowsnb number of rows to skip to get lines.

Rowsnb number of the rows to read, 1 by default.

Returns:

Colsnb number of columns in all rows, 0 if rows have different number of columns

Rows rows.

geophpy.dataset.**fileformat_getlist**()

Get list of format files availables

Returns: list of file formats availables, ['ascii', ...]

geophpy.dataset.**plottype_getlist**()

Get list of plot type availables

Returns : list of plot type availables, ['2D_SURFACE', '2D_CONTOUR', ...]

geophpy.dataset.**interpolation_getlist**()

Get list of interpolation methods availables

Returns : list of interpolation methods availables, ['bilinear', 'bicubic', ...]

geophpy.dataset.**colormap_getlist**()

Getting the colormap list.

Returns : list of colormap availables, ['gray', ...]

geophpy.dataset.**colormap_plot**(*cmname*, *creversed=None*, *fig=None*, *filename=None*, *dpi=None*, *transparent=False*)

Plots the colormap.

Parameters :

Cmname Name of the colormap, 'gray_r' for example.

Creversed True to add '_r' at the cmname to reverse the color map

Fig figure to plot, None by default to create a new figure.

Filename Name of the color map file to save, None if no file to save.

Dpi 'dot per inch' definition of the picture file if filename != None

Transparent True to manage the transparency.

Returns:

Fig Figure Object

geophpy.dataset.**pictureformat_getlist**()

Get list of pictures format availables

Returns: list of picture formats availables, ['jpg', 'png', ...]

geophpy.dataset.**rasterformat_getlist**()

Get list of raster format files availables

Returns : list of raster file formats availables, ['jpg', 'png', ...]

geophpy.dataset.**correlmap**(*dataset*, *method*)

To compute the correlation map from an image :

- each profile (from the input list) is incrementally shifted and correlation coefficient computed against neighbouring profiles for each shift value
- profiles are considered to be vertical, and the shift performed along the profile (hence vertically)
- the correlation map size is then "twice the image vertical size" (shift may vary from -ymax to +ymax) by "number of profiles" (correlation is computed for each column listed in input)

Parameters:

Dataset DataSet Object containing the Z-image to be correl-mapped

Cols column numbers (1D array) of the image profiles to be correlated

Method correlation method to use (from festooncorrelation_getlist())

Returns:

Cor1 correlation map (shape = (2*zimg.shape[0]-1 , cols.size))

Pva1 weight map (see correlation functions description)

`geophpy.dataset.`**`griddinginterpolation_getlist`**`()`
To get the list of available gridding interpolation methods.

`geophpy.dataset.`**`festooncorrelation_getlist`**`()`
To get the list of available festoon correlation methods.

class `geophpy.dataset.`**`DataSet`**
Creates a DataSet Object to process and display data.

info = Info() data = Data() georef = GeoRefSystem()

`continuation`(*prosptech*, *apod=0*, *downsensoraltitude=0.3*, *upsensoraltitude=1.0*, *continuationflag=False*, *continuationvalue=0.0*)
Continuation of the magnetic field

Parameters :

Prosptech Prospection technical

Apod apodisation factor, to limit side effects

Downsensoraltitude Altitude of the down magnetic sensor

Upsensoraltitude Altitude of the upper magnetic sensor

Continuationflag Continuation flag, False if not continuation

Continuationaltitude Continuation altitude, greater than 0 if upper earth ground, lower than 0 else

`copy`()
To duplicate a DataSet Object.

Parameters:

Dataset DataSet Object to duplicate

Returns:

Newdataset duplicated DataSet Object

`destripecon`(*Nprof=0*, *setmin=None*, *setmax=None*, *method='add'*, *valfilt=False*)
To destripe a DataSet Object by a constant (Zero Mean Traverse)

Parameters:

Dataset DataSet Object to be destriped

Nprof number of profiles over which to compute the mean of reference ; if set to 0 (default), compute the mean over the whole data

Setmin while computing the mean, do not take into account data values lower than setmin

Setmax while computing the mean, do not take into account data values greater than setmax

Method if set to "add" (default), destriping is done additively ; if set to "mult", it is done multiplicatively

Valfilt if set to True, then filters data.values instead of data.zimage

festoonfilt (*method='Pearson'*, *shift=0*, *valfilt=False*)

To filter festoon-like artefacts out of DataSet Object

Parameters:

Dataset DataSet Object to be filtered

Method correlation method to use (from festooncorrelation_getlist())

Shift systematic shift value (in pixels) to apply ; if shift=0 then the shift value will be determined for each profile by correlation with neighbours

Valfilt if set to True, then filters data.values instead of data.zimage

Returns:

Shift shift value, modified if shift=0 in input parameter

static from_file (*filenameslist*, *fileformat=None*, *delimiter='\t'*, *x_colnum=1*, *y_colnum=2*, *z_colnum=3*, *skipinitialspace=True*, *skip_rows=1*, *fields_row=0*)

To build a DataSet Object from a file.

Parameters:

Filenameslist list of files to open ['file1.xyz', 'file2.xyz' ...] or ['file*.xyz'] to open all files with filename beginning by "file" and ending by ".xyz"

Fileformat format of files to open (None by default implies automatic determination from filename extension)

Note: all files must have the same format

Delimiter pattern delimitting fields within one line (e.g. ' ', ',', ';' ...)

X_colnum column number of the X coordinate of the profile (1 by default)

Y_colnum column number of the Y coordinate inside the profile (2 by default)

Z_colnum column number of the measurement profile (3 by default)

Skipinitialspace if True, several contiguous delimiters are equivalent to one

Skip_rows number of rows to skip at the beginning of the file, i.e. to skip header rows (1 by default)

Fields_row row number where to read the field names ; if -1 then default field names will be “X”, “Y” and “Z”

Returns:

Success true if DataSet Object built, false if not

Dataset DataSet Object build from file(s) (empty if any error)

Example:

success, dataset = DataSet.from_file(“file.csv”)

histo_getlimits()

Get the limits values.

Returns : zmin, zmax

histo_plot(*fig=None*, *filename=None*, *zmin=None*, *zmax=None*, *dpi=None*, *transparent=False*)

Plot histogram.

Parameters :

Fig figure to plot, None by default to create a new figure.

Filename Name of the histogram file to save, None if no file to save.

Zmin Minimal Z value to represent.

Zmax Maximal Z value to represent.

Dpi ‘dot per inch’ definition of the picture file if filename != None

Transparent True to manage the transparency.

Returns :

Fig Figure Object

logtransform(*multfactor=0*)

To transform the data in logarihtmics values

Parameters:

Dataset DataSet Object to be treated

Multfactor multiply factor

medianfilt(*nx=3*, *ny=3*, *percent=0*, *gap=0*, *valfilt=False*)

To process a DataSet Object with a median filter

Parameters:

Dataset Dataset Object to process

Nx filter size in x coordinate

Ny filter size in y coordinate

Percent deviation (in percents) from the median value (for absolute field measurements)

Gap deviation (in raw units) from the median value (for relative anomaly measurements)

Valfilt if set to True, then filters data.values instead of data.zimage

peakfilt(*setmin=None*, *setmax=None*, *setmed=False*, *setnan=False*, *valfilt=False*)

To eliminate peaks from a DataSet Object.

Parameters:

Dataset DataSet Object to eliminate peaks from

Setmin minimal threshold value

Setmax maximal threshold value

Setmed if set to True, then values beyond threshold are replaced by a median instead of the threshold value ; the median is computed ...TBD...

Setnan if set to True, then values beyond thresholds are replaced by nan instead of the threshold value

Note: if both "setnan" and "setmed" are True at the same time, then "nan" prevails

Valfilt if set to True, then filters data.values instead of data.zimage

plot(*plottype*, *cmapname*, *creversed=False*, *fig=None*, *filename=None*, *cmmin=None*, *cmmax=None*, *interpolation='bilinear'*, *cmapdisplay=True*, *axisdisplay=True*, *pointsdisplay=False*, *dpi=None*, *transparent=False*, *logscale=False*, *rects=None*, *points=None*)

Plot in 2D or 3D dimensions the cartography representation.

Parameters :

Plottype plotting type, '2D-SURFACE', '2D-CONTOUR', ...

Cmapname name of the color map used, 'gray_r' for example.

Creversed True to add '_r' at the cmname to reverse the color map

Fig figure to plot, None by default to create a new figure.

Filename name of the picture file to save, None if no file to save.

Cmmin minimal value to display in the color map range.

Cmmax maximal value to display in the color map range.

Interpolation interpolation mode to display DataSet Image ('bilinear', 'bicubic', 'spline16', ...), 'bilinear' by default.

Cmapdisplay True to display color map bar, False to hide the color map bar.

Axisdisplay True to display axis and labels, False to hide axis and labels

Pointsdisplay True to display grid points, False to not display them.

Dpi 'dot per inch' definition of the picture file if filename != None

Transparent True to manage the transparency for the zones of lack of data

Logscale True to display with the log scale

Rects [[x0, y0, w0, h0], [x1, y1, w1, h1], ...], None if no selection rectangles to display

Returns : fig, plt, cmap. None, None, None if filename with a wrong picture format

Fig Figure Object

Cmap ColorMap Object

`ploughfilt` (*nx=3*, *ny=3*, *apod=0*, *angle=None*, *cutoff=None*, *valfilt=False*)

To apply anti-ploughing filter to a DataSet Object

Parameters:

Dataset DataSet Object to be filtered

Nx filter size in x coordinate

Ny filter size in y coordinate

Apod apodisation factor (percent)

Angle ...TBD... float/degrees

Cutoff ...TBD... float/frequency

Valfilt if set to True, then filters data.values instead of data.zimage

`polereduction` (*apod=0*, *inclineangle=65*, *alphaangle=0*)

To do a reduction at the pole of DataSet Object

Parameters:

Apod apodisation factor, to limit side effects

Inclineangle magnetic field incline

Alphaangle magnetic field alpha angle

regtrend (*nx=3*, *ny=3*, *method='rel'*, *component='loc'*, *valfilt=False*)
To filter a DataSet Object from its regional trend

Parameters:

Dataset DataSet Object to be filtered

Nx filter size in x coordinate

Ny filter size in y coordinate

Method set to "rel" to filter by relative value (resistivity) or to "abs" to filter by absolute value (magnetic field)

Component set to "loc" to keep the local variations or to "reg" to keep regional variations

Valfilt if set to True, then filters data.values instead of data.zimage

to_file (*filename*, *fileformat=None*, *delimiter='\t'*, *description=None*)
Save a DataSet Object to a file.

Parameters :

Filename name of file to save.

Fileformat format of output file ...

Delimiter delimiter between fields in a line of the output file, ' ', ',', ';', ...

Returns :

Success boolean

wallisfilt (*nx=3*, *ny=3*, *setmean=None*, *setstdev=None*, *setgain=None*, *limit=None*, *edgefactor=None*, *valfilt=False*)
To apply Wallis filter to a DataSet Object

Parameters:

Dataset DataSet Object to be filtered

Nx filter size in x coordinate

Ny filter size in y coordinate

Setmean ...TBD... float

Setstdev ...TBD... float

Setgain ...TBD... float

Limit ...TBD... float

Edgefactor ...TBD... float{0..1}

Valfilt if set to True, then filters data.values instead of data.zimage

8 Feedback & Contribute

Your feedback is more than welcome.

Write email to lionel.darras@mom.fr or philippe.marty@upmc.fr

9 Changelog

9.1 Version 0.20

Released on 2015-09-10

- Initial version.