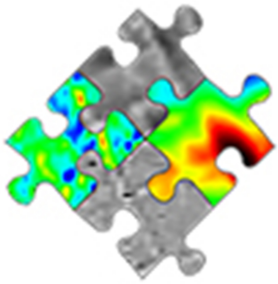


GeophPy Documentation

Release 0.3

Lionel Darras, Philippe Marty & Quentin Vitale

Dec 19, 2017

1 About GeophPy

1.1 Introduction

The GeophPy project was initiated in 2014 through cooperation between two units of the CNRS[1] (UMR5133-Archeorient and UMR7619-Metis). Since 2017, it is also developped by the LabCom Geo-Heritage (a cooperation between UMR5133-Archeorient and Eveha International).

1.2 Description

GeophPy is an open-source python module which aims at building tools to display and process sub-surface geophysical data in the field of archaeology, geology, and other sub-surface applications.

The main feature of this module is to build a geophysical *DataSet Object* composed by series of data in the format (X,Y,Z) with (X,Y) being the point position of the geophysical value Z, in order to process and/or display maps of Z values.

This module is designed to be used in a graphical user interface software, WuMapPy, but can also be used independently in a script or in command line.

1.3 Features

- Building a data set from one or severals data files.
- Displaying geophysical maps in 2D or 3D.
- Processing data sets with geophysicals methods and filters.

[1] French National Center for Scientific Research

- Compatibility with Python 3.x

1.4 Main authors

- **Lionel DARRAS**

  French National Center for Scientific Research, UMR 5133 Archeorient, Lyon, France

  lionel.darras@mom.fr

- **Philippe MARTY**

  French National Center for Scientific Research, UMR 7619 METIS, Paris, France

  philippe.marty@upmc.fr

- **Quentin VITALE**

  Eveha International, Lyon, France

  quentin.vitale@eveha.fr

1.5 Licence

2 Installation

A Python (3.x) installation is necessary to install this package.

2.1 Manual installation

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

Place yourself in the "GeophPy-vx.y" unzipped folder and install GeophPy with the following command:

```
>>>  python setup.py install
```

Note: Installation on Windows

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

2.2 Dependencies

GeophPy uses packages that should be automatically installed with GeophPy. If their installation failed they can be installed independently.

GeophPy dependencies:

- numpy-1.11.0+mkl

- scipy-0.17.1
- matplotlib-1.4.2
- netCDF4-1.2.4
- Pillow-3.2.0
- PySide-1.2.2
- pyshp-1.2.3
- simplekml
- utm

3 Command-line usage

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

3.1 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)
```

3.2 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
```

3.3 Data Set Object 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 automatically built after opening file but by using gridding interpolation function `dataset.interpolate()`. See *Data Set operation* for details.

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).

3.4 Data Set operation

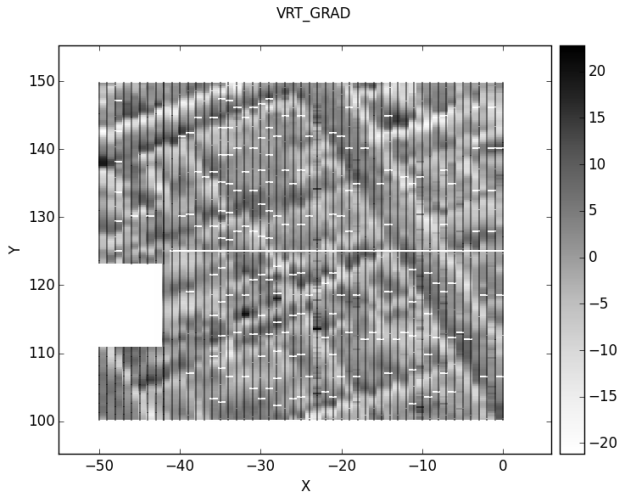
You can duplicate a data set before processing it to save the raw data:

```
>>> 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):



3.5 Data Set Processing

The dataset can be processed using the available processing techniques in a simple comand line of the form:

```
>>> dataset.ProcessingTechnique(option1='True', option2='10',...)
```

The available processing techniques are listed below.

Peak filtering

Peak filtering allows data thresholding for values outside of the [**setmin**, **setmax**] interval.

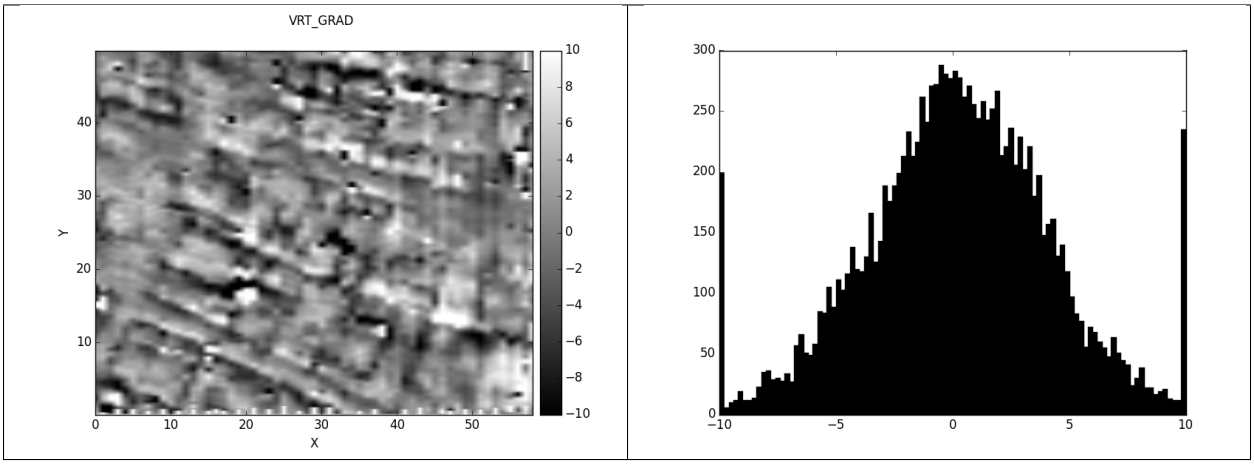
Out of range data can be replaced by:

- the interval bounds (**setmin** and **setmax**);
- NaN values (if **setnan** is True),
- the profiles' medians (if **setmed** is True).

Examples:

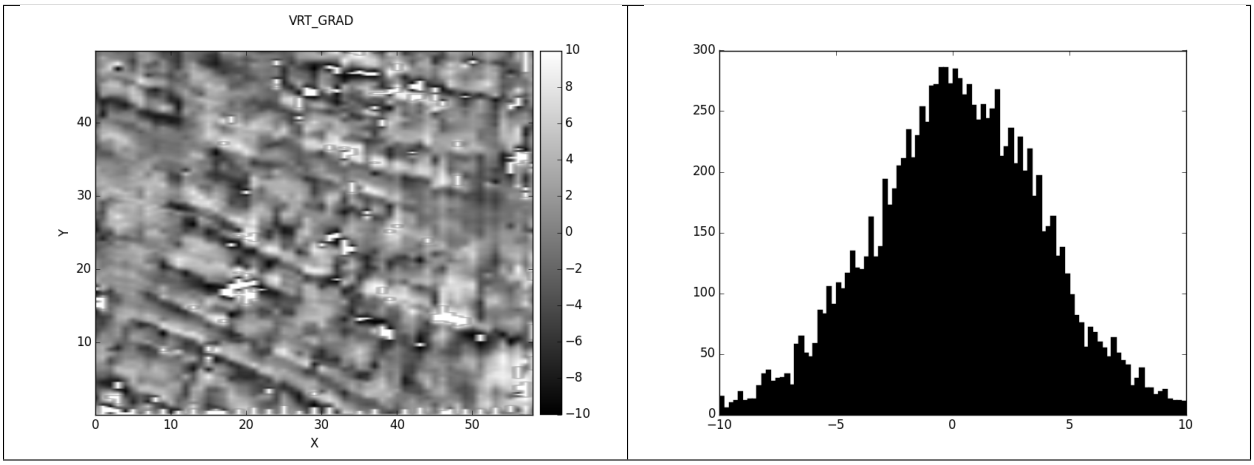
- replacing by lower and upper bounds:

```
>>> dataset.peakfilt(setmin=-10, setmax=10)
```



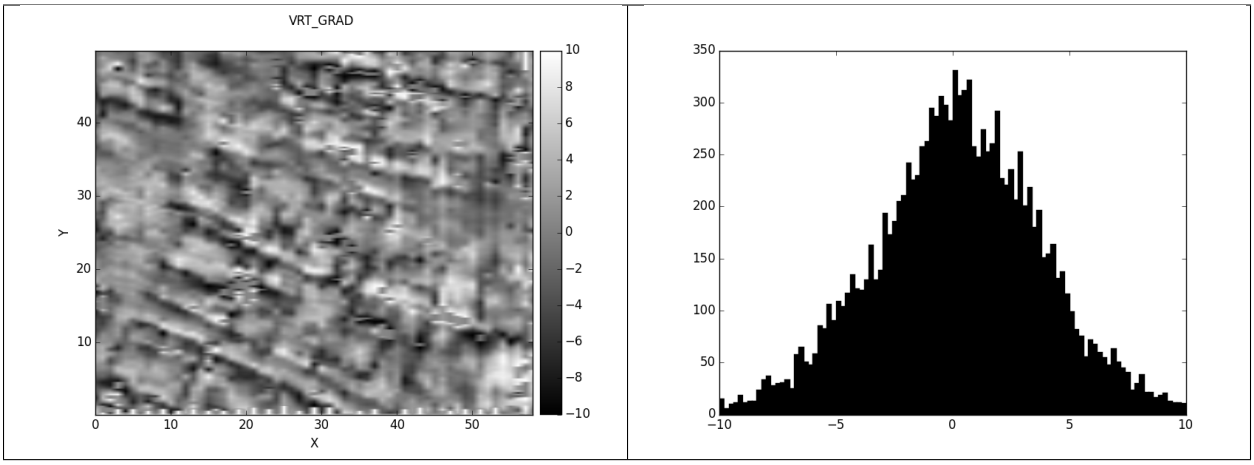
- replacing by NaNs:

```
>>> dataset.peakfilt(setmin=-10, setmax=10, setnan=True)
```



- replacing by profile's median:

```
>>> dataset.peakfilt(setmin=-10, setmax=10, setmed=True)
```



Note: Input parameters:

- **setmin**: minimal threshold value,
- **setmax**: maximal threshold value,
- **setnan**: (if True) out of range data are replaced by nan,
- **setmed**: (if True) out of range data are replaced by the profile's medians.

Median filtering

"Median filtering is a non linear process useful in reducing impulsive, or salt-and-pepper noise" *[LimJ90]*. It is capable of smoothing a few out of bounds pixels while preserving image's discontinuities without affecting the other pixels.

A window is slided along the image and the local median value is calculated. The window size (**nx**, **ny**) is a critical filter parameter and depens on every image. A trial and error approach is recommanded for choosing a suitable value.

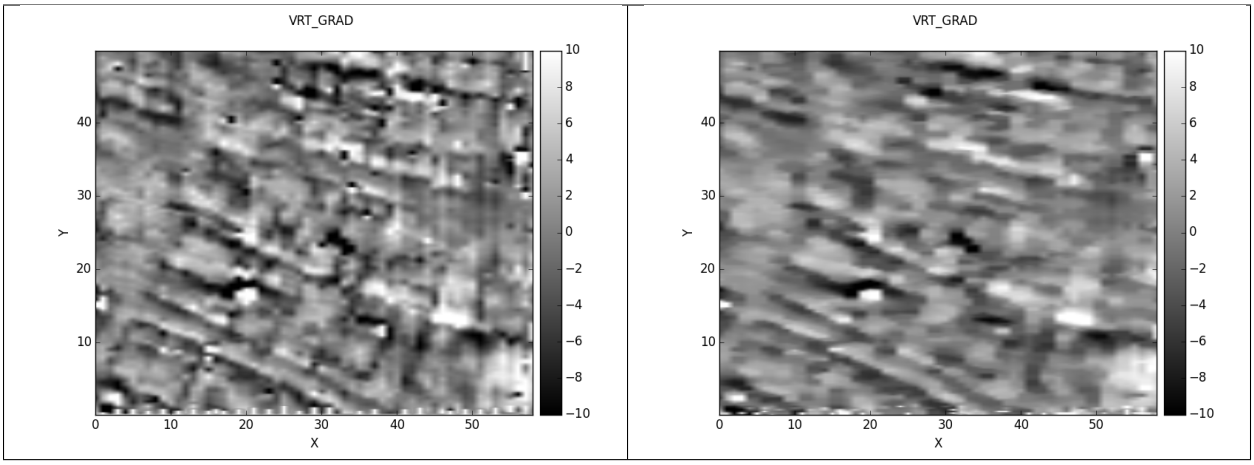
A threshold value can be set for the replacement by the local median value. The threshold deviation from the local median can be set:

- in percentage (**percent** =10) or raw units (**gap** =5),
- if no threshold is given, all pixels are replaced by their local medians.

Examples:

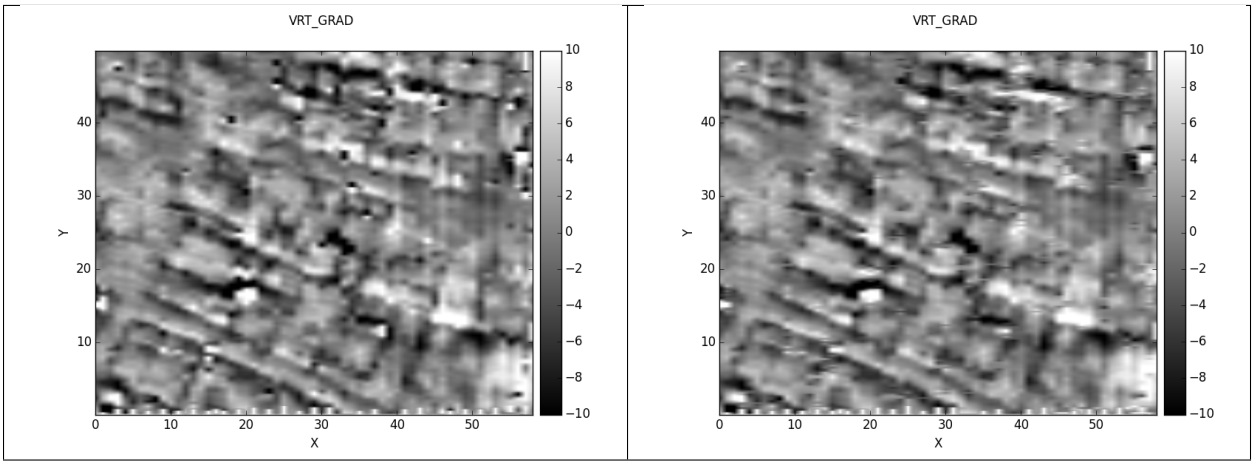
- No threshold:

```
>>> dataset.medianfilt(nx=3, ny=3)
```



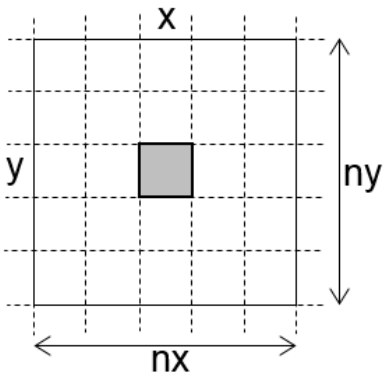
- Threshold (in raw value):

```
>>> dataset.medianfilt(nx=3, ny=3, gap=5)
```



Principle:

For each pixel in the dataset, the local median of the (**nx** * **ny**) neighboring points is calculated.



A threshold value is defined and if the deviation from the local median is higher than this threshold, then the center pixel value is replaced by the local median value. The threshold deviation from the local median can be defined as a percentage of the local median or directly in raw units.

Note: Input parameters:

- **nx**: filter size in x coordinate,
- **ny**: filter size in y coordinate,
- **percent**: deviation (in percents) from the median value,
- **gap**: deviation (in raw units) from the median value.

Festoon filtering

The festoon filter is a destagger filter that reduces the positionning error along the survey profiles that result in a festoon-like effect. An optimum shift is estimated based on the correlation of a particular profile and the mean of its surrounding profiles.

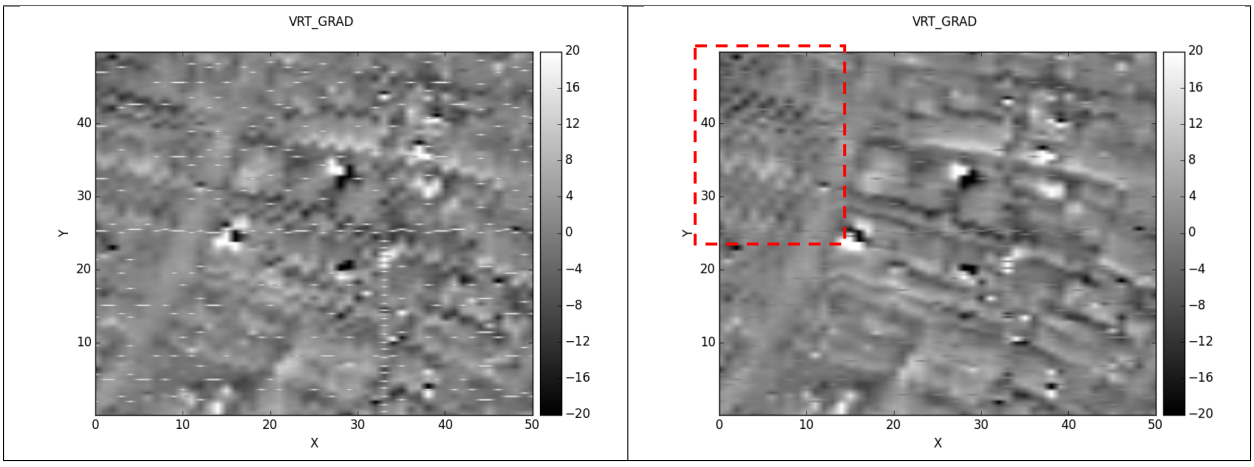
This optimum shift can be:

- uniform throughout the map (**uniformshift** =True),
- or different for each profile (**uniformshift** =False).

Examples:

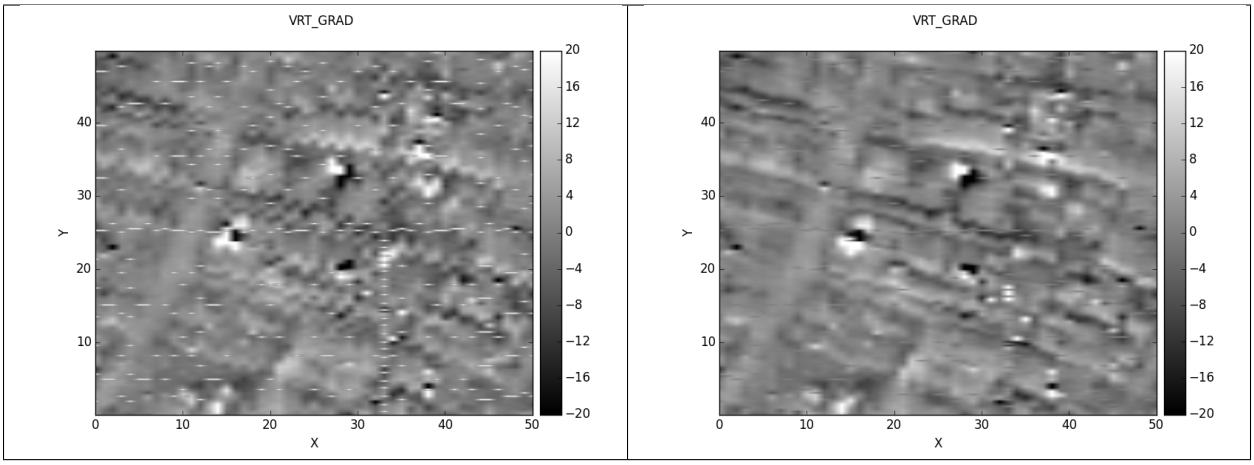
- Uniform shift

```
>>> dataset.festoonfilt(method='Crosscorr', uniformshift=True)
```



- Non uniform shift

```
>>> dataset.festoonfilt(method='Crosscorr', uniformshift=False)
```

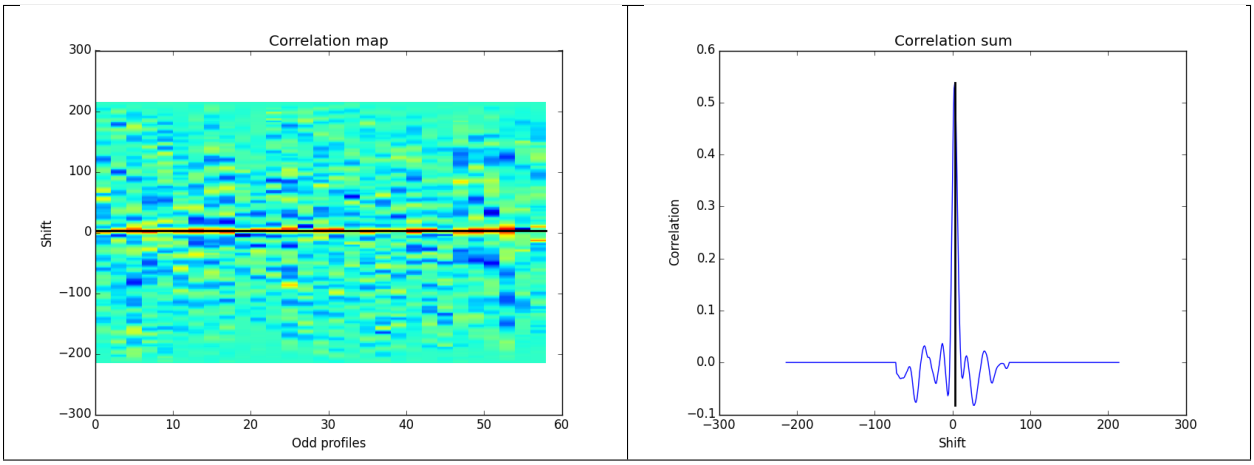


Principle:

For every odd profiles (columns) in the dataset, an optimum shift is estimated based on the correlation of the profile and the mean of its surrounding profiles. A correlation map is hence produced. This optimum shift can be uniform throughout the map (**uniformshift** =True) or different for each profile (**uniformshift** =False). If the shift is set uniform, the mean correlation profile is used as correlation map.

At the top and bottom edges of the correlation map (high shift values), high correlation values can arrise from low sample correlation calculation. To prevent those high correlation values to drag the best shift estimation, a limitation is set to only consider correlation with at least 50% overlap between profiles. Similarly, a minimum correlation value (**corrmin**) can be defined to prevent profile's shift if the correlation is too low.

```
>>> fig = dataset.correlation_plotmap(method='Crosscorr')
>>> fig.show()
>>> fig = dataset.correlation_plotsum(method='Crosscorr')
>>> fig.show()
```



Note: Input parameters:

- **method**: correlation method to use ('Crosscorr', 'Pearson', 'Spearman', 'Kendall'),
- **shift**: values (in pixels) to apply if known,
- **corrmin**: minimum correlation coefficient value for profile shifting [0-1],

- **uniformshift**: flag for shift estimation.
- **uniformshift**: (if True) the shift is set uniform on the map; (if False) the shift is profile dependent

---

Regional trend filtering

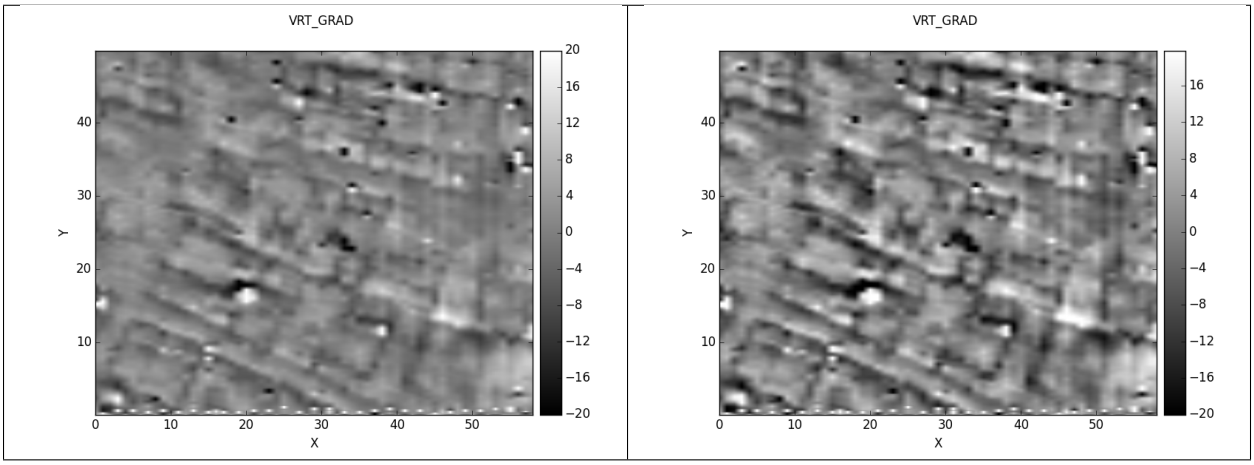
Remove the background (or regional response) from a dataset to enhance the features of interest.

Wallis filtering

The Wallis filter is a locally adaptative contrast enhancement filter *[STHH90]*. It is based on the local statistical properties of sub-window in the image. It adjusts brightness values (grayscale image) in the local window so that the local mean and standard deviation match target values.

Examples:

```
>>> dataset.wallisfilt()
```



Principle:

A window of size (**nx**, **ny**) is slided along the image and at each pixel the Wallis operator is calculated. The Wallis operator is defined as:

$$\frac{A\sigma_d}{A\sigma_{(x,y)} + \sigma_d}[f_{(x,y)} - m_{(x,y)}] + \alpha m_d + (1 - \alpha)m_{(x,y)}$$

where

- A is the amplification factor for contrast,
- σ_d is the target standard deviation,
- $\sigma_{(x,y)}$ is the standard deviation in the current window,
- $f_{(x,y)}$ is the center pixel of the current window,
- $m_{(x,y)}$ is the mean of the current window,
- α is the edge factor (controlling portion of the observed mean, and brightness locally to reduce or increase the total range),

- m_d is the target mean.

As the Wallis filter is design for grayscale image, the data are internally converted to brightness level before applying the filter. The conversion is based on the minimum and maximum value in the dataset and uses 256 levels (from 0 to 255). The filtered brightness level are converted back to data afterwards.

A quite large window is recomanded to ensure algorithm stability.

Note: Input parameters:

- **nx**: filter window size in x coordinate,
- **ny**: filter window size in y coordinate,
- **targmean**: target mean level (m_d),
- **targstdev**: target standard deviation level (σ_d),
- **setgain**: amplification factor for contrast (A),
- **limitstdev**: limitation on the window standard deviation in the window to prevent too high gain value when data are dispersed,
- **edgefactor**: brightness forcing factor that controls ratio of edge to background intensities (α).

Plough filtering

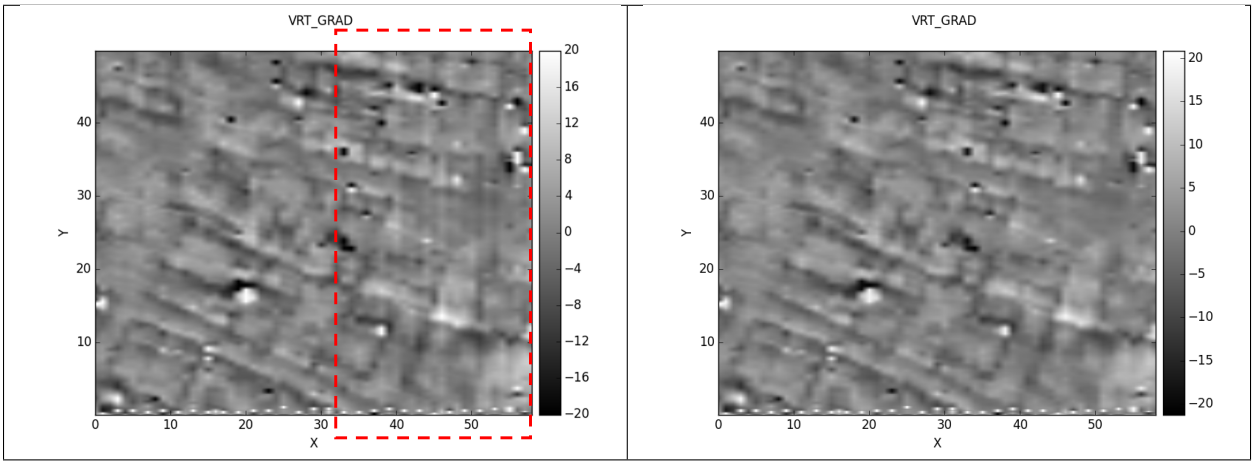
... To Be Developped ...

Constant destriping

Remove from the dataset the strip noise effect arising from profile-to-profile differences in sensor height, orientation, drift or sensitivity (multi-sensors array). Constant destriping is done using Moment Matching method *[GaCs00]*.

Examples:

```
>>> dataset.destripecon(Nprof=4, method='add')
```



Principle:

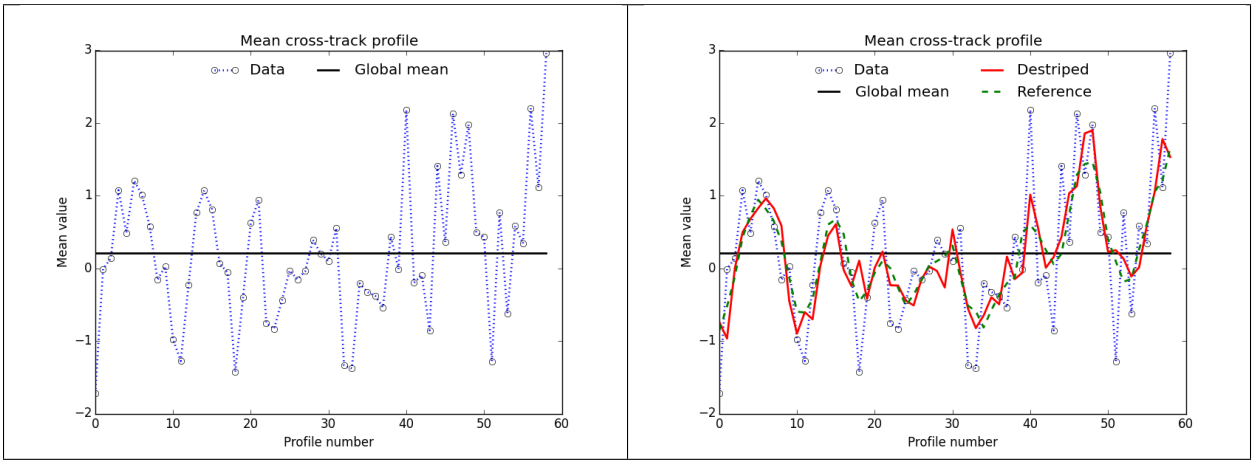
In constant destripping, a linear relationship is assumed between profile-to-profile offset (means) and gain (standard deviation). The statistical moments (mean m_i and standard deviation σ_i) of each profile in the dataset are computed and matched to reference values.

The reference values typically are:

- the mean (m_d) and standard deviation (σ_d) of the **Nprof** neighbouring profiles,
- the mean and standard deviation of the global dataset (**Nprof** =0),
- alternatively, one can use the median and interquartile range instead of mean and standard deviation.

The data mean cross-track profile before and after destriping can be displayed as follow:

```
>>> fig = dataset.destrip_plotmean(Nprof=4, method='add', Ndeg=None, plotflag='raw')
>>> fig.show()
>>> fig = dataset.destrip_plotmean(Nprof=4, method='add', Ndeg=None, plotflag='both')
>>> fig.show()
```



Considering and additive relation (**method** ='add'), the destripped value can be expressed as ([*RiJi06*], [*Scho07*]):

$$f_{corr} = \frac{\sigma_d}{\sigma_i}(f - m_i) + m_d$$

where

- f_{corr} is destripped value,
- σ_d is the reference standard deviation,
- σ_i is the current profile standard deviation,
- f is the current value,
- m_i is the current profile,
- m_d is the reference mean.

Unlike remote sensing, ground surveys often demonstate profile-to-profile offsets but rarely gain changes so that only matching profiles mean (or median) is usually appropriate (**configuration** ='mono'):

$$f_{corr} = f - m_i + m_d$$

Finally, a multiplicative relation can be considered (**method** ='mult'), and the destripped value are hence expressed as:

$$f_{corr} = f\frac{\sigma_d}{\sigma_i}\frac{m_d}{m_i}$$

Note: Input parameters:

- **Nprof**: number of profiles for the reference moments computation (0= computes over the whole dataset),
- **setmin**: do not take into account data values lower than setmin,
- **setmax**: do not take into account data values greater than setmax,
- **method**: 'additive' method for the destriping; 'multiplicative' method for the destriping,
- **reference**: 'mean': references are mean and standard deviation; 'median': references are median and interquartile range,
- **configuration**: 'mono' sensors: destriping with only offset matching; 'multi' sensors: destriping with both offset and gain.

Curve destriping

Remove from the dataset the strip noise effect by fitting and subtracting a polynomial curve to each profile on the dataset.

Logarithmic transformation

The logarithmic transformation is contrast enhancement filter originally used for geological magnetic data. It enhances information present in magnetic data at low-amplitude values while preserving the relative amplitude information via logarithmic transformation procedure *[MPBL01]*.

Principle:

Originally used for geological magnetic data, the logarithmic transformation principle can be summarized as follow:

- Contrast in magnetic susceptibility and magnetic remanence are the physical rock properties that controlls magnetic anomalies (the other controlling factors being the geometry and the position of the source body).
- Magnetic mineral content variation and borehole magnetic susceptibility are know to be best represented by a log-normal distribution.
- If a similar distribution is assumed to represent to represent lithologies on magnetic anomaly maps, then log transformation of the magnetic data should serve to normalize the distribution and highlight features having common amplitude.

The log-normal transformation of magnetic data f is implemented like so:

$$f_{log} = \begin{cases} -\log_{10}(-f) & for \quad f < -1 \\ \log_{10}(f) & for \quad f > 1 \\ 0 & for \quad -1 < f > 1 \end{cases}$$

Note: Input parameters:

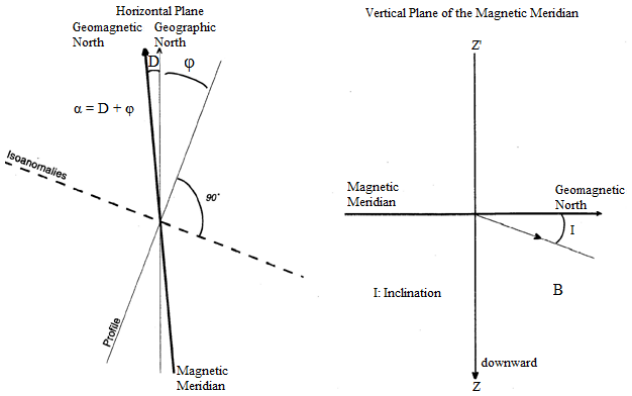
- **multfactor**: multiplying factor of the data (x5, x10, x20, x100).

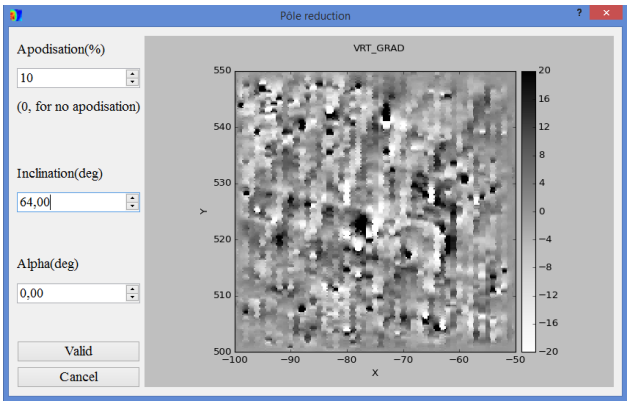
Pole reduction

The reduction to the magnetic pole is a way to facilitate magnetic data interpretation and comparizon. The reduction to the pole calculates the anomaly that would have been obtained if the survey had been done at the pole where the inclination of the magnetic field is maximum (vertical). *[BLAK96]*

Principle:

Due to the dipolar nature of the geomagnetic field, magnetic anomalies (if not located at the magnetic poles) are asymmetric with a geometry that depends on the magnetic inclination (I). Assuming that the remanent magnetism of the source is small compared to the induced magnetism, the filter symmetrize the anomalies and palce them directly above the source. It uses a fast Fourier series algorithm to work in the spectral (frequency) domain. A similar processing (reduction to the equator) is used when data are recored at low magnetic inclinations.

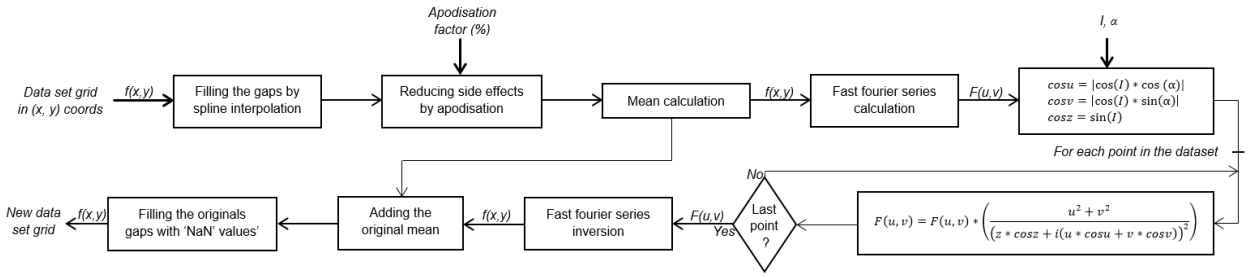




Note: Input parameters:

- **apod**: apodisation factor.
- **inclineangle**: inclination angle (angle between the magnetic North and the magnetic field measured at the soil surface in the vertical plane).
- **alphaangle**: angle between magnetic North and profiles direction in the horizontal plane.

Algorithm:



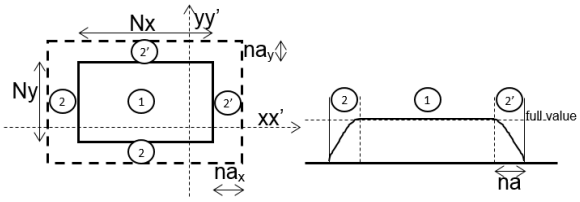
u is the spatial frequency corresponding to the x-direction and v is the spatial frequency corresponding to the y-direction.

- Filling gaps:

To use the fast Fourier algorithm, the data set must not contains gaps or 'NaN' values (Not A Number). If the data set grid is not interpolated and contains NaNs, the blancs will be automatically filled using profile by profile spline interpolation.

- 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 %)

$Zone\ 1 : value = full\ value$

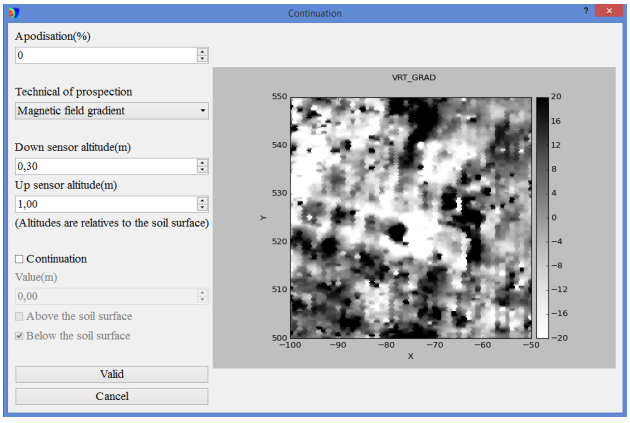
$Zone\ 2 : value = full\ value * \cos\left(\frac{\pi}{2} * \frac{na - n}{na}\right), with\ na = nax\ or\ nay, and\ 0 \leq n \leq na$

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

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. *[BLAK96]*

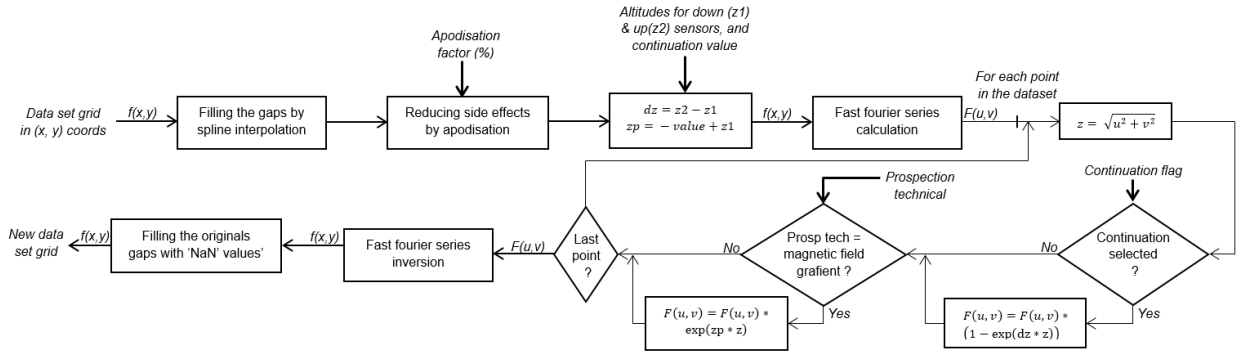


Note: Input parameters:

- **apod**: factor of apodisation(%).
- **prosptech**: Survey configuration ("Magnetic field", "Magnetic field gradient", "Vertical component gradient").

- **downsensoraltitude, upsensoraltitude**: Sensors altitudes relatives to th soil surface.
- **continuationflag**: Continuation flag, False if not continuation
- **continuationvalue**: Continuation altitude (greater than 0 if upper earth ground, lower than 0 otherwise).

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 is described in the end of this document (see *High level API*) but about the detailed source code of is available in this section.

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='Crosscorr'*, *shift=0*, *corrmin=0.4*, *uniformshift=False*, *valfilt=False*)

cf. dataset.py

`geophpy.processing.general.`**`regtrend`**(*dataset*, *nx=3*, *ny=3*, *method='relative'*, *component='local'*, *valfilt=False*)

cf. dataset.py

`geophpy.processing.general.`**`wallisfilt`**(*dataset*, *nx=11*, *ny=11*, *targmean=125*, *targstdev=50*, *setgain=8*, *limitstdev=25*, *edgefactor=0.1*, *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='additive'*, *reference='mean'*, *config='mono'*, *valfilt=False*)

cf. dataset.py

geophpy.processing.general.**destripecub**(*dataset*, *Nprof=0*, *setmin=None*, *setmax=None*, *Ndeg=3*, *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*)

cf. dataset.py

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

cf. dataset.py

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

cf. dataset.py

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

3.6 DataSet Plotting

General Plot Information

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.

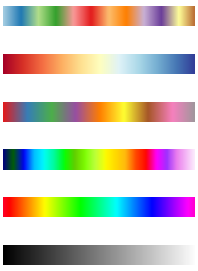
To see the color maps available, you type:

```
>>> 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

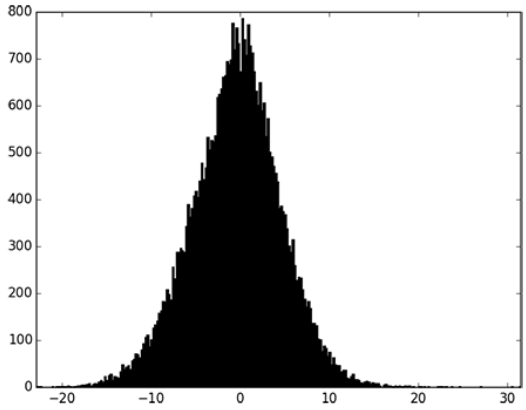
To adjust the limits of color map you must view the limits of the data set:

```
>>> 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:

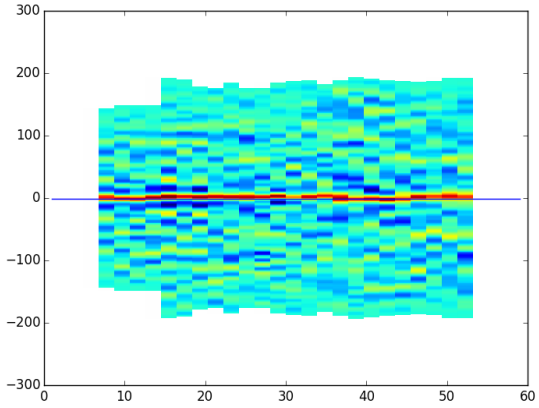
```
>>> h_fig = dataset.histo_plot()
```

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:

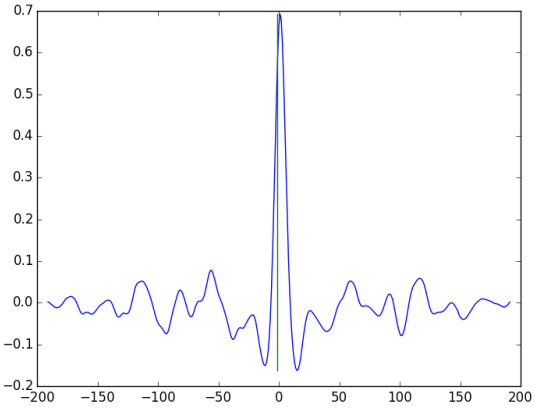
```
>>> h_fig = dataset.histo_plot()
```

Correlation sums

You can plot the correlation sums of a dataset:

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

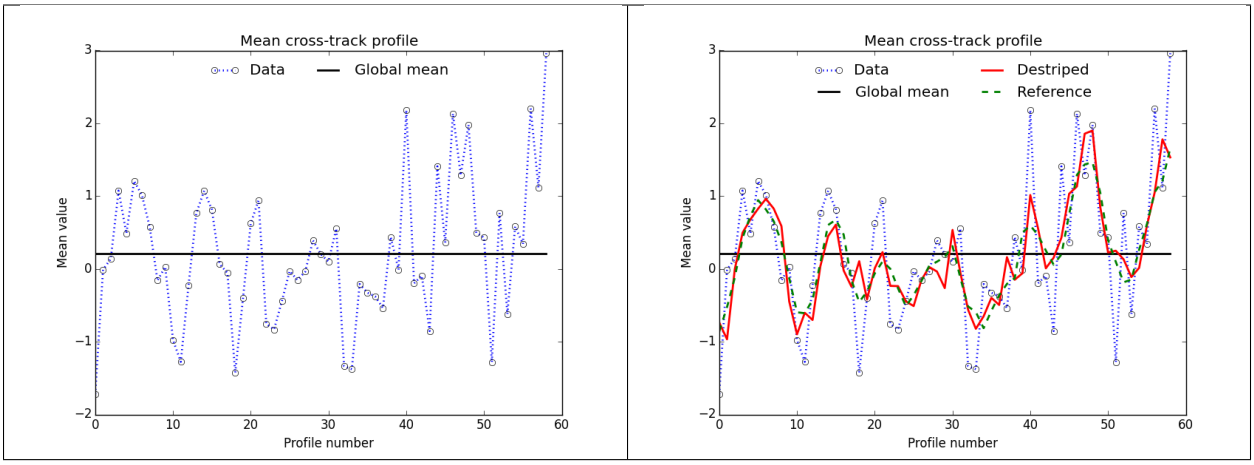
to obtain:



Mean cross-track profile

Before and after destriping mean cross-track profiles can be displayed with the following commands:

```
>>> fig = dataset.destrip_plotmean(Nprof=4, method='add', Ndeg=None, plotflag='raw')
>>> fig.show()
>>> fig = dataset.destrip_plotmean(Nprof=4, method='add', Ndeg=None, plotflag='both')
>>> fig.show()
```



Map plotting

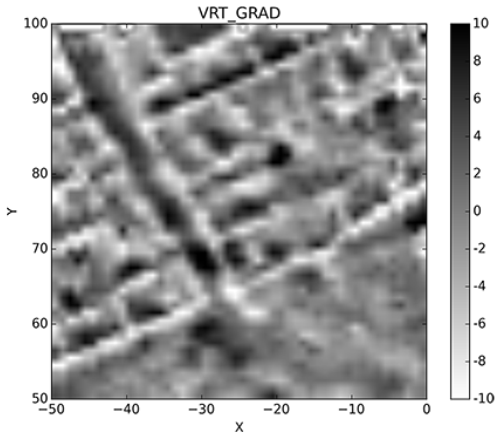
You can plot a data set using one of these plot types:

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

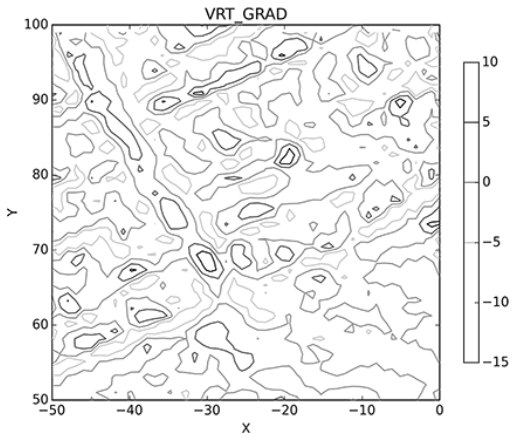
Examples:

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

‘2D-SURFACE’:

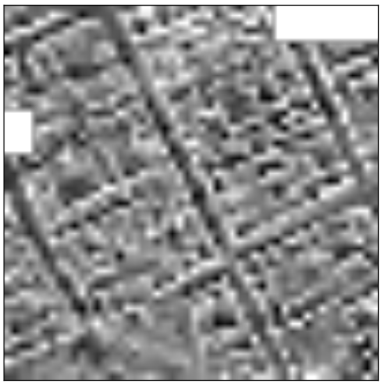


‘2D-CONTOUR’:



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.

High level plotting functions

The calling protocol of these functions is described in the end of this document (see *High level API*) but about the detailed source code of is available in this section.

geophpy.plotting.histo

Map Plotting Histofeam Managing.

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

license GNU GPL v3.

geophpy.plotting.histo.**getlimits**(*self*)
getting limits values of histogram.

geophpy.plotting.histo.**plot**(*dataset*, *fig=None*, *filename=None*, *zmin=None*, *zmax=None*, *cmapname=None*, *dpi=None*, *transparent=False*)
plotting the histogram curve.

geophpy.plotting.correlation

Map Plotting Correlation Managing.

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

license GNU GPL v3.

geophpy.plotting.correlation.**plotmap**(*dataset*, *fig=None*, *filename=None*, *method='Crosscorr'*, *dpi=None*, *transparent=False*)
plotting the correlation map.

geophpy.plotting.correlation.**plotsum**(*dataset*, *fig=None*, *filename=None*, *method='Crosscorr'*, *dpi=None*, *transparent=False*)
plotting the correlation sum.

geophpy.plotting.destrip

Map Plotting Destriping Managing.

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

license GNU GPL v3.

geophpy.plotting.destrip.**plotmean**(*dataset*, *fig=None*, *filename=None*, *Nprof=0*, *method='add'*, *reference='mean'*, *config='mono'*, *Ndeg=None*, *plotflag='raw'*, *dpi=None*, *transparent=False*)
Plotting the mean cross-track (mean of each profile).

geophpy.plotting.plot

Map Plotting.

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

license GNU GPL v3.

geophpy.plotting.plot.**plot**(*dataset*, *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*)

plotting with a representation type selected

3.7 DataSet 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')
```

3.8 Geographic Positioning Set

You can open a file containing the geographic coordinates of the geophysical survey corresponding you *DataSet* object. you can use

- an ascii file (*.csv* for instance):

  The first line must contain the geographic reference system and the other lines contain the point number followed by its GPS coordinates (longitude and lattitude or UTM Easting and Northing) and eventually its local coordinates:

  ```
  >>> WGS84
  >>> 1        66.84617533     37.74956917
  >>> 2        66.84649517     37.7489535
  >>> 3        66.8472475      37.74972867
  >>> 4        66.84689417     37.7491385
  >>> 5        66.84691867     37.7491025
  >>> ...      ...             ...
  ```

  Or

  ```
  >>> UTM
  >>> 1        745038.191      4656005.727     150     0
  >>> 2        745068.172      4656045.663     150     50
  >>> 3        745028.43       4656076.057     100     50
  >>> 4        744988.466      4656105.978     50      50
  >>> 5        744998.428      4656036.093     100     0
  >>> ...      ...             ...             ...     ...
  ```

- or a shapefile (*.shp*).

To open your Geographic Positioning Set file simply use:

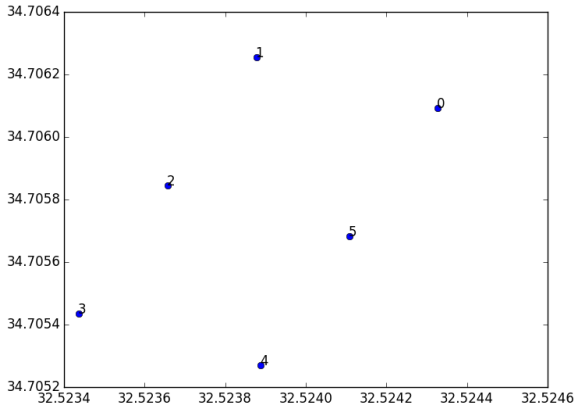
```
>>> from geoposset import *
>>> gpset = GeoPosSet.from_file(refsys='WGS84', type='shapefile',
    ["pt_topo"])
```

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

4 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

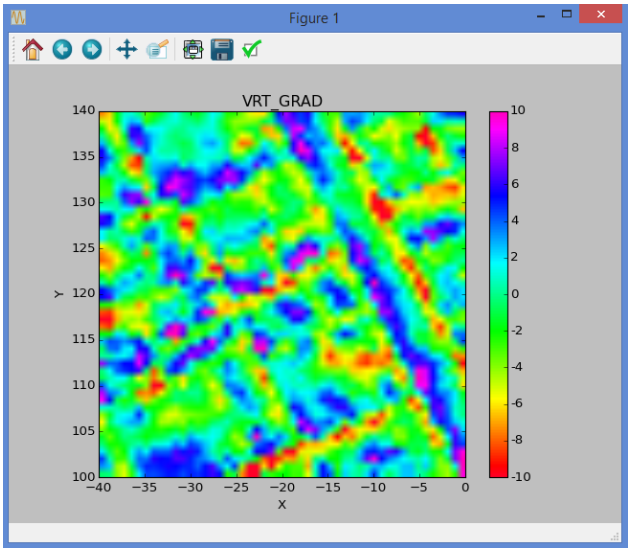
Note:

- 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)
```

5 High level API

5.1 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*)

Computes the correlation map from DataSet Object:

- each odd profile in the dataset is incrementally shifted and the 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

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='additive'*, *reference='mean'*, *config='mono'*, *valfilt=False*)

To destripe a DataSet Object by a constant (Moment Matching method).

The statistical moments (mean and standard deviation) of each profile in the dataset are computed and matched to reference values.

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 'additive' (default), destriping is done additively ; if set to 'multiplicative', it is done multiplicatively

Reference if set to 'mean' (default), destriping is done using mean an standard deviation ; if set to 'median', it is done using median and interquartile range

Config if set to 'mono' (default), destriping is done using only offset matching (mean / median) ; if set to 'multi', it is done using both offset and gain (standard deviation/interquartile range)

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

destripecub(*Nprof=0*, *setmin=None*, *setmax=None*, *Ndeg=3*, *valfilt=False*)

To destripe a DataSet Object by a cubic curvilinear regression (chi squared)

Parameters:

Dataset DataSet Object to be destriped

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

Setmin while fitting the polynomial curve, do not take into account data values lower than setmin

Setmax while fitting the polynomail curve, do not take into account data values greater than setmax

Ndeg polynomial degree of the curve to fit

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

festoonfilt(*method='Crosscorr'*, *shift=0*, *corrmin=0.4*, *uniformshift=True*, *valfilt=False*)

Filters festoon-like artefacts out of DataSet Object using correlation between profiles or provided shift.

If no shift (0) is provided, the correlation map for each odd profile and the mean of its neighbours is computed and used to estimate the best shift.

Parameters:

Dataset DataSet Object to be filtered

Method correlation method to use (from festooncorrelation_getlist(): 'Crosscorr', 'Pearson', 'Spearman' or 'Kendall')

Shift shift value (in pixels) to apply ; if shift=0 the shift value will be determined for each profile by correlation with neighbours ; if shift is a vector each value in shift will be applied to its corresponding odd profile. In that case shift must have the same size as the number of odd profiles.

Corrmin minimum correlation coefficient value for profile shifting [0-1].

Uniformshift if set to True, the shift is uniform on the map ; if set to False the shift is profile dependent.

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*, *cmapname=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.

Cmapname name of the color map used, 'gray_r' for example; if None black histogram is used.

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.

...To Be Developed...

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='relative'*, *component='local'*, *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 "relative" to filter by relative value (resistivity) or to "absolute" to filter by absolute value (magnetic field)

Component set to "local" to keep the local variations or to "regional" 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=11*, *ny=11*, *targmean=125*, *targstdev=50*, *setgain=8*, *limitstdev=25*, *edgefactor=0.1*, *valfilt=False*)
Applies a Wallis (contrast enhancement) filter to a DataSet Object.

The Wallis filter is a locally adaptative contrast enhancement filter based on the local statistical properties of sub- window in the image.

Parameters:

Dataset DataSet Object to be filtered

Nx filter window size in x coordinate

Ny filter window size in y coordinate

Targmean float, the target mean level (m_d)

Targstdev float, the target standard deviation (sigma_d)

Setgain float, amplification factor for contrast (A)

Limitstdev float, limitation on the window standard deviation to prevent too high gain value if data are dispersed

Edgefactor float {0..1} , brightness forcing factor (alpha), controls ratio of edge to background intensities.

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

6 Feedback & Contribute

Your feedback is more than welcome.

Write email to lionel.darras@mom.fr, philippe.marty@upmc.fr or quentin.vitale@eveha.fr

To cite this software : "Marty, P., Darras, L. (2015). GeophPy. Tools for geophysical survey data processing (version x.y) [software]. Available at https://pypi.python.org/pypi/GeophPy."

7 Changelog

7.1 Version 0.30

Released on 2017-12-01.

- Updated GeophPy documentation.
- Added options for constant destriping filter.
- Added Mean cross-track profile plot for destripping filters.
- Implemented Wallis filter.
- Implemented replace by profile's median in peak filtering.
- Added automatic delimiter search in delimited files.
- Fixed reading delimited file issues.
- Added non uniform shift for Festoon.
- Fixed correlation and cross-correlation map calculation.
- Added color map for histogram plot.
- Fixed bug in histo.plot.

7.2 Version 0.21

Released on 2016-05-01

- Initial version.

8 References

References

[BLAK96] Blakely R. J. 1996. Potential Theory in Gravity and Magnetic Applications. Cambridge University Press.

[GaCs00] Gadallah F. L., Csillag F. 2000. Destriping multisensor imagery with moment matching. Int. J. Remote Sensing, vol. 21, no. 12, p2505-2511.

[LimJ90] Lim, J. S. 1990. Two-Dimensional Signal and Image Processing. p469-476. Prentice-Hall.

[MPBL01] Morris B., Pozza M., Boyce J. and Leblanc G. 2001. Enhancement of magnetic data by logarithmic transformation. The Leading Edge, vol. 20, no. 8, p882-885.

[RiJi06] Richards, J. A. and X. Jia 2006. Remote Sensing Digital Image Analysis - An Introduction, 4rth edition. Chapter 2.2.3 p37. Springer.

[Scho07] Schowengerdt R. A. .2007. Remote Sensing: Models and Methods for Image Processing, 3rd edition. Chapter 7.4 p325. Elsevier.

[STHH90] Scollar I., Tabbagh A., Hesse A. and Herzog I. 1990. Archaeological Prospecting and Remote Sensing (Topics in Remote Sensing 2). 647p. Cambridge University Press.