wellpathpy Documentation

Release 0.2.0

Robert Leckenby, Jørgen Kvalsvik, Brendon Hall

Aug 17, 2023

Contents:

CHAPTER 1

Introduction

wellpathpy is a LGPL-3.0 licensed library to import well deviations in md, inc, azi format, calculate their tvd values using a choice of methods and return them as positional logs in tvd, northing, easting format.

CHAPTER 2

Wellpathpy tutorial

This document aims to provide a sample workthrough using wellpathpy showing:

2.1 Abbreviations

m	metres
ft	feet
md	measured depth
inc	inclination
azi	azimuth
tvd	true vertical depth
east_offset	horizontal distance away from wellhead towards the east
north_offset	horizontal distance away from wellhead towards the north
tvdss	true vertical depth subsea
mE	horizontal distance in meters away from surface location towards the east
mN	horizontal distance in meters away from surface location towards the north

2.2 Imports

In this tutorial, we will alias wellpathpy as wp:

```
import wellpathpy as wp
```

2.3 Loading a deviation

Wellpathpy provides a fairly simple loading function for reading a deviation survey from CSV.

2.3.1 Loading a deviation from CSV

A valid input file must be a CSV file containing the columns: md, inc, azi in that order, as shown in this example:

```
md,inc,azi
0,0,244
10,11,220
50,43,254
150,78.5,254
252.5,90,359.9
```

- column headers are generally expected but will be skipped when the file is read
- if no headers are provided, the `skiprows` argument can be set to `0`
- md must increase monotonically
- as inc and azi cannot be distinguished numerically it is the user's responsibility to ensure the data are passed in this order
- inc must be in range 0 <= inc < 180 (to allow for horizontal wells to climb)
- azi must be in range 0 <= azi < 360

You can then load them into wellpathpy using:

```
md, inc, azi = wp.read_csv(fname)
```

`wp.read_csv` simply calls `np.loadtxt` with `delimiter=','` and `skiprows=1`. These can be changed if required; for example the `delimiter` and `skiprows` can be changed with:

```
md, inc, azi = wp.read_csv(fname, delimiter='\t', skiprows=0)
```

Additional `kwargs` accepted by `np.loadtxt` can also be passed in, for example:

```
md, inc, azi = wp.read_csv(fname, comments='$')
```

Notes:

Some simple sanity checks are performed to reject bad CSVs. `wp.read_csv` supports all options `np.loadtxt` supports. Only those columns named md, inc, azi will be read.

If the deviation survey is not in CSV, is generated in a different place in your program, or is from some other source, wellpathpy is still useful. If you provide three `np.ndarray` md, inc, and azi, the rest of wellpathpy works fine.

Observe that the same basic requirements still apply:

- md, inc and azi have the same shape
- md increases monotonically
- inc is in range 0-180
- azi is in range 0-360

Once `md`, `inc` and `azi` have been returned from `wp.read_csv()`, an instance of the `wp.deviation()` class is created with:

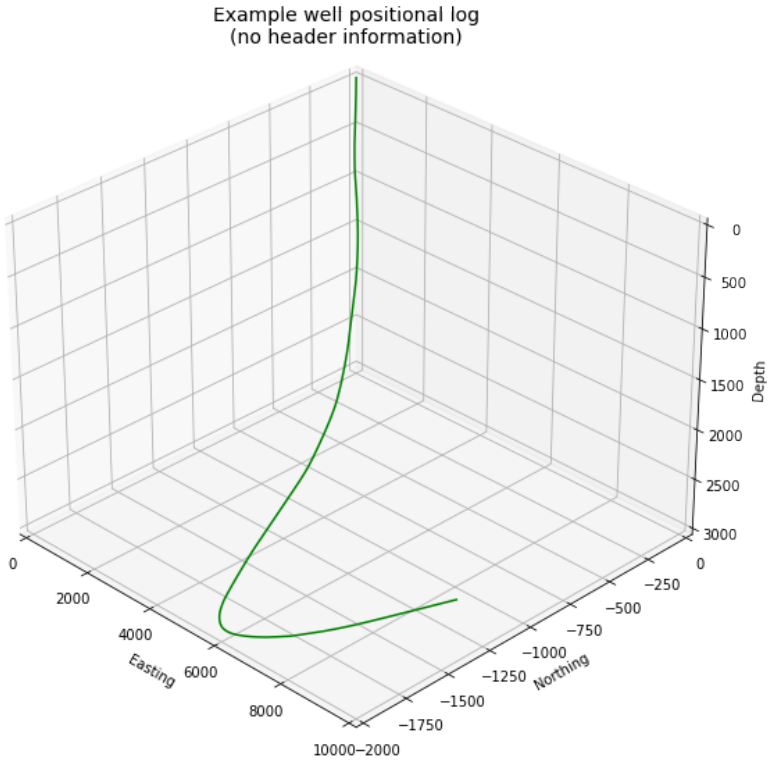
```
dev = wp.deviation(
    md = md,
    inc = inc,
    azi = azi
)
```

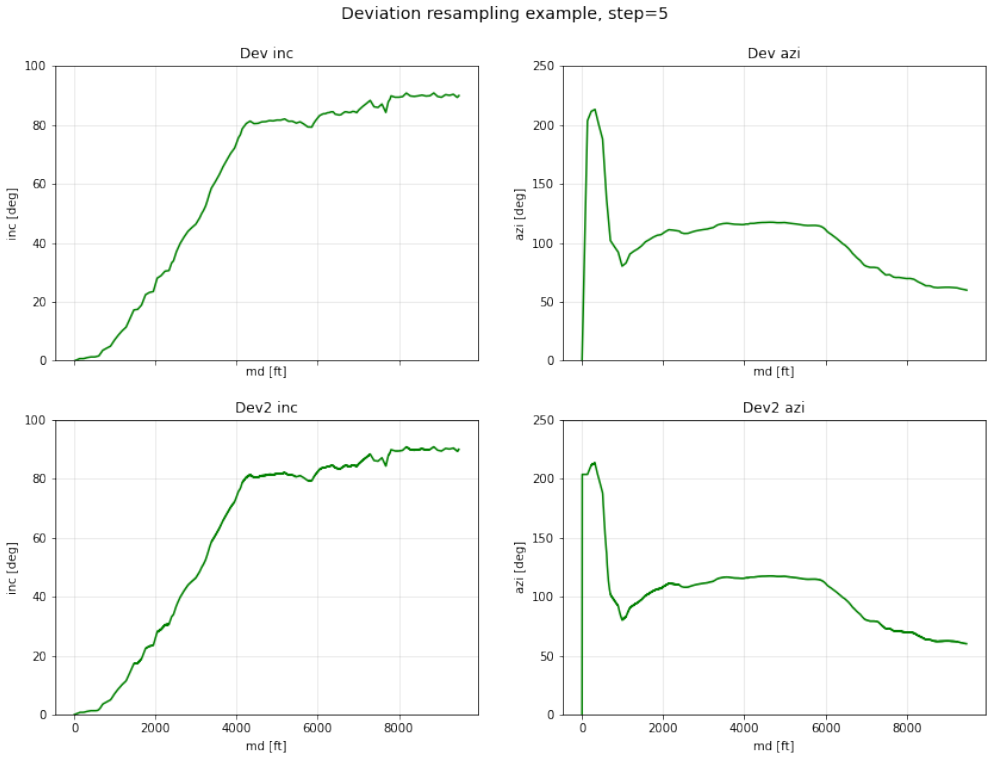
With this, it is then possible to resample the depths using the `minimum_curvature()` method and go back to a deviation survey in `md`, `inc` and `azi`:

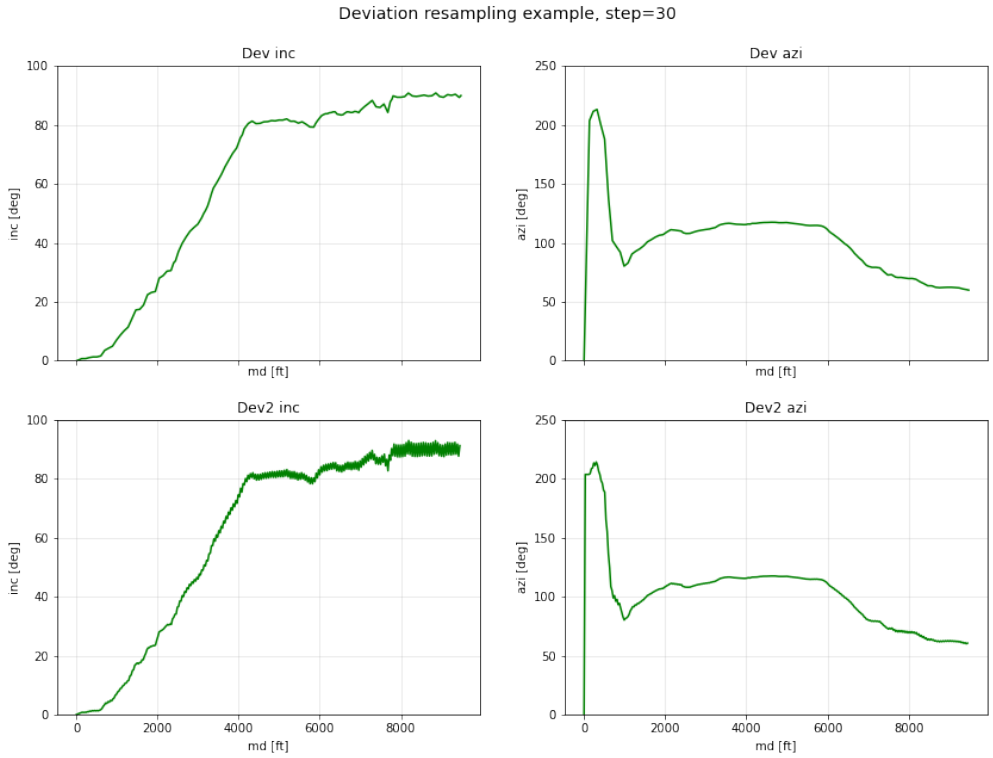
```
step = 30
depths = list(range(0, int(dev.md[-1]) + 1, step))
pos = dev.minimum_curvature().resample(depths = depths)
dev2 = pos.deviation()
```

Notes:

With increasing step size, float uncertainty can introduce some noise as shown in the figures below. First we see an overview of the well in 3D, followed by plots of inclination and azimuth versus depth.


Example well positional log
(no header information)
Depth
0
500
1000
1500
2000
2500
3000
Easting
0
2000
4000
6000
8000
10000
Northing
0
−250
−500
−750
−1000
−1250
−1500
−1750
−2000


Deviation resampling example, step=5
Dev inc
Dev azi
Dev2 inc
Dev2 azi
inc [deg]
azi [deg]
md [ft]
0
20
40
60
80
100
50
150
200
250
2000
4000
6000
8000



2.4 Loading the well header

To make sense of the deviation position, wellpathpy supports reading a survey header from json file. The header requires the following keys:

```
{
"datum": "kb",
"elevation_units": "m",
"elevation": 100.0,
"surface_coordinates_units": "m",
"surface_easting": 1000.0,
"surface_northing": 2000.0
}
```

```
header = wp.read_header_json(fname)
```

Notes:

This function is provided for convenience - wellpathpy does not care about the source of this data. It will simply use `json.load()` to read the JSON file and save it as a python `dict`.

2.5 Converting deviation surveys to positional logs

All these methods can be accessed from the *deviation* object created with:

```
dev = wp.deviation(
    md = md,
    inc = inc,
    azi = azi,
    )
```

2.5.1 Standard method

The standard method for converting a **deviation surveys** [md, inc, azi] into a **positional logs** [tvd, northing, easting] is the *minimum curvature* method. This method is provided by wellpathpy and is recommended for most use cases.

- **minimum curvature method** [`dev.minimum_curvature()`] This method uses angles from upper and lower end of survey interval to calculate a curve that passes through both survey points. This curve is smoothed by use of the ratio factor defined by the tortuosity or dogleg of the wellpath. This method returns a dogleg severity calculated for a given course_length.

2.5.2 Comparison methods

Other methods are provided should the need arise to compare *mininum curvature* to older surveys that may have been calculated with one of these methods. In general these other methods are **not recommended**.

- **radius of curvature method** [`dev.radius_curvature()`] Calculate TVD using radius or curvature method. **Caution**: this will yield unreliable results when data are closely spaced or when the borehole is straight but deviated. This method uses angles from upper and lower end of survey interval to calculate a curve that passes through both survey points.
- **average tan method** [`dev.tan_method()`] Calculate TVD using average tangential method. This method averages the inclination and azimuth at the top and bottom of the survey interval before taking their sine and cosine, this average angle is used to estimate tvd.
- **balanced tan method** [`dev.tan_method(choice='bal')`] Calculate TVD using balanced tangential method. This method takes the sines and cosines of the inclination and azimuth at the top and bottom of the survey interval before averaging them, this average angle is used to estimate tvd. This will provide a smoother curve than the average tan method but requires closely spaced survey stations to avoid errors.
- **high tan method** [`dev.tan_method(choice='high')`] Calculate TVD using high tangential method. This method takes the sines and cosines of the inclination and azimuth at the bottom of the survey interval to estimate tvd. This method is **not recommended** as it can make gross tvd and offset errors in typical deviated wells.
- **low tan method** [`dev.tan_method(choice='low')`] Calculate TVD using low tangential method. This method takes the sines and cosines of the inclination and azimuth at the top of the survey interval to estimate tvd. This method is **not recommended** as it can make gross tvd and offset errors in typical deviated wells.

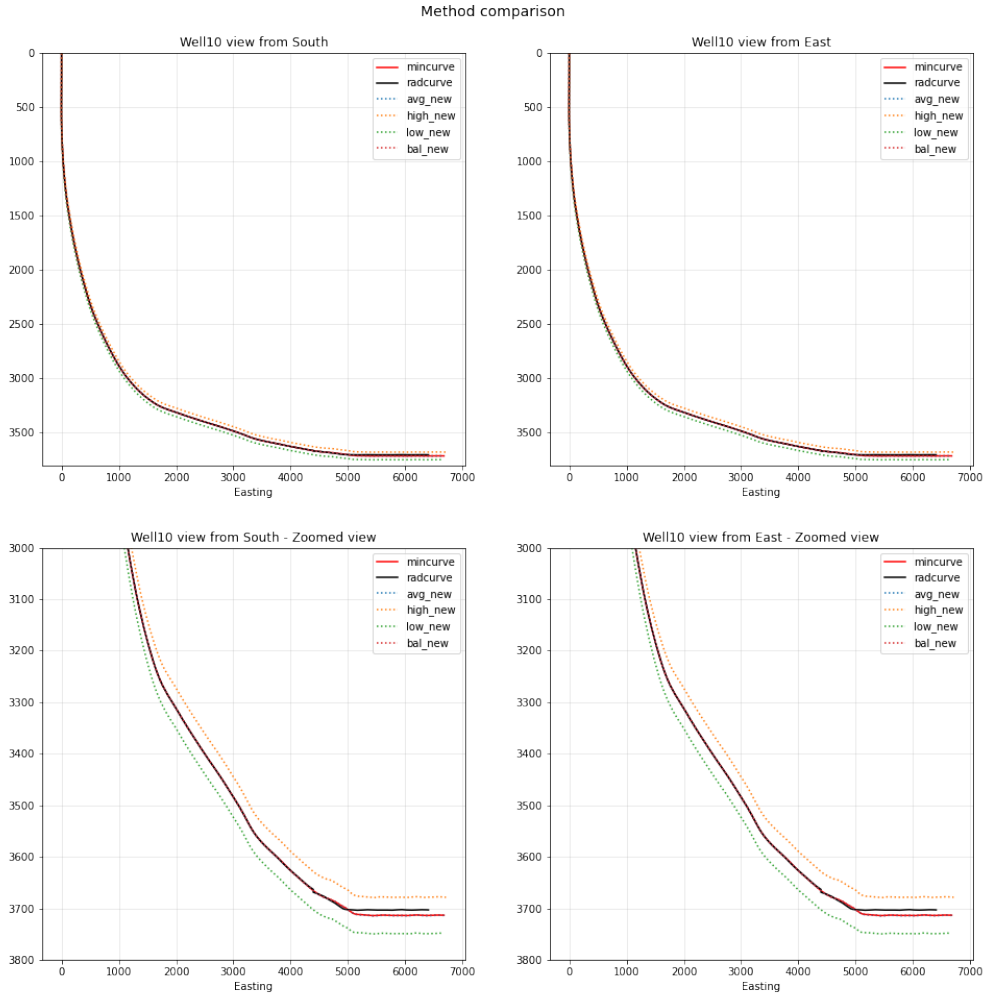
2.5.3 Usage

In order to use these functions, you first need a deviation object as described in *Loading a deviation*. You can then run the following methods once you've imported your *deviation* and *header* and done any unit conversion required as described above.

```
# The recommended method for most use-cases
tvd, northing, easting, dls = dev.mininum_curvature(course_length=30)

# Comparison methods to contrast with older deviation surveys
tvd, northing, easting      = dev.radius_curvature()
tvd, northing, easting      = dev.tan_method() # for the default 'avg' method
tvd, northing, easting      = dev.tan_method(choice='bal')
tvd, northing, easting      = dev.tan_method(choice='high')
tvd, northing, easting      = dev.tan_method(choice='low')
```

We can compare the outputs of all these methods in the figure below:



2.6 Well location and tvdss

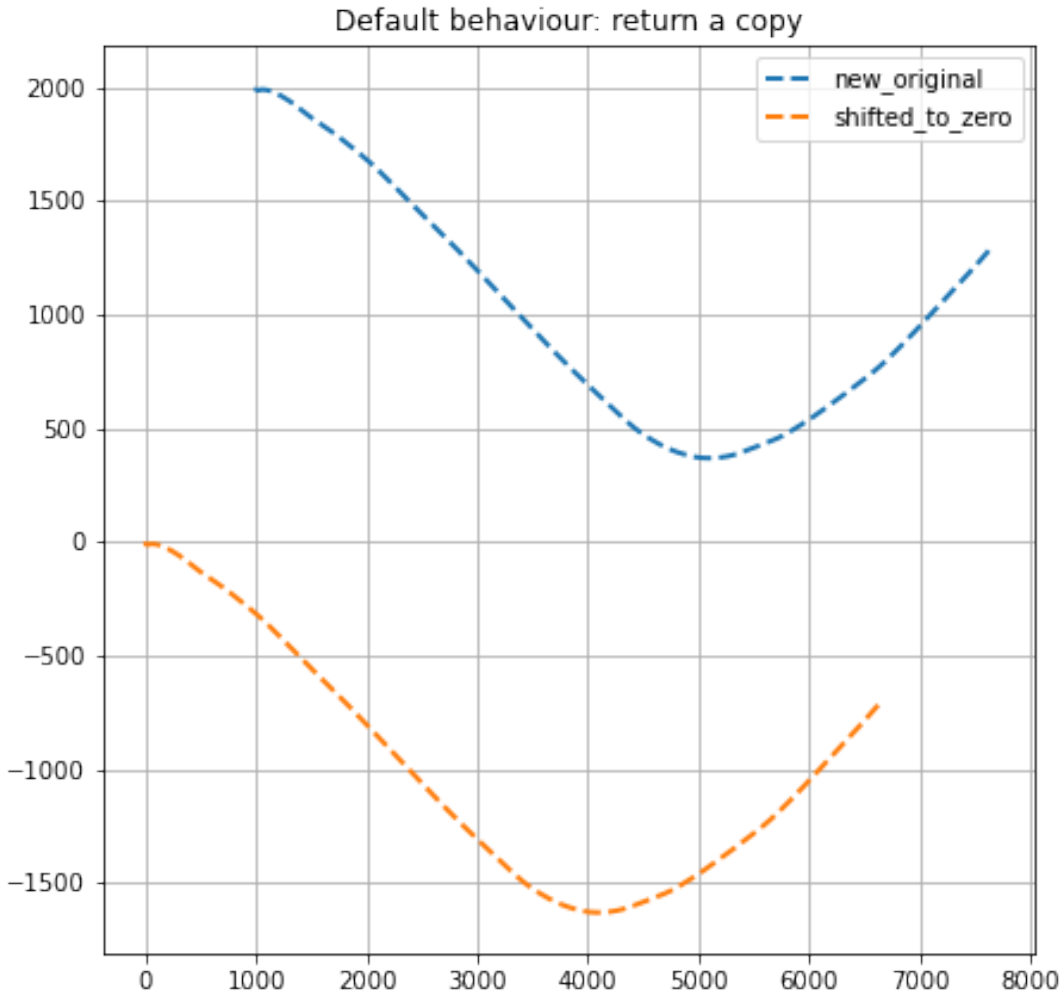
The methods above are not aware of surface location or datum elevation. If you want to move the positional log to a given surface location, to 0,0 coordinates, or shift the tvd to tvdss, you can use the following functions which return a copy of the positional log by default (`inplace=False`).

- to shift a positional log to a wellhead location

```
pos_wellhead = pos.to_wellhead(surface_northing=surface_northing,
                               surface_easting=surface_easting)
```

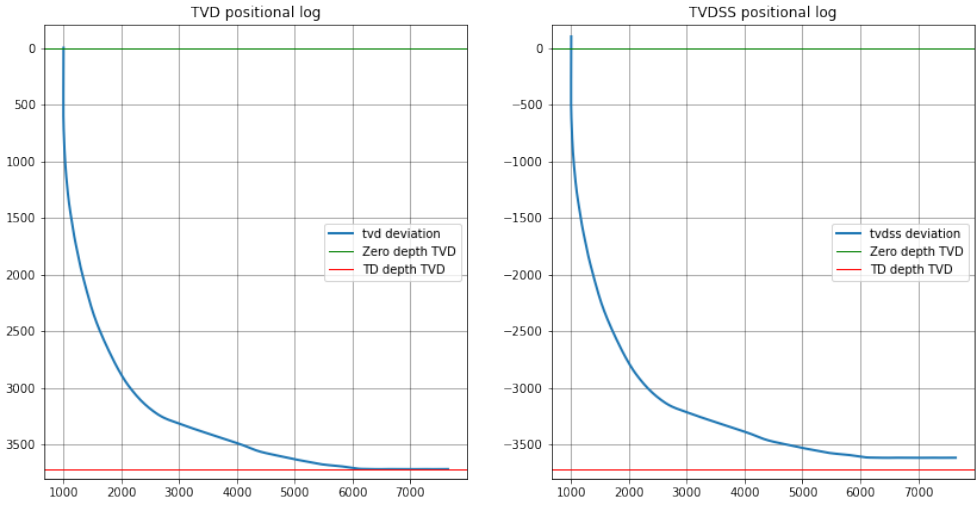
- to shift a positional log to a 0,0 coordinate location

```
pos_zero = pos_wellhead.to_zero(surface_northing=surface_northing,
                                surface_easting=surface_easting)
```



- to shift a positional log to tvdss

```
pos_tvdss = pos.to_tvdss(datum_elevation=header['elevation'])
```



If you have a header loaded as shown in the *Loading the well header* section, you can use that object to access the required properties with:

```
surface_northing = header['surface_northing']
surface_easting  = header['surface_easting']
datum_elevation  = header['datum_elevation']
```

Notes:

Wellpathpy is oblivious to units, and assumes the input units are consistent.

2.7 Exporting results

The two main `wellpathpy` objects; `deviation` and `position` logs can be written to CSV via object methods as shown below. These both call `np.savetxt` with `fmt='%.3f'` and `delimiter=','`. The `deviation` also has `header='md,inc,azi'` and the `postition` has `header='easting,northing,depth'`. Other `kwargs` accepted by `np.savetxt` are also accepted.

- for a deviation survey:

```
dev.to_csv('./deviation.csv')
```

- for a positional log:

```
pos.to_csv('./position.csv')
```

Additional `kwargs` can be passed like:

```
dev.to_csv('./deviation.csv', fmt='%.2e')
pos.to_csv('./position.csv', header='X,Y,Z', comments='$')
```

This is a pretty straight-forward function convenient CSV writing. If you need more control, or more sophisticated output, you must implement your own writer.

CHAPTER 3

API reference

wellpathpy.**read_header_json**(*fname*)

Read deviation header

The deviation header information is needed for surface location and true vertical depth subsea calculations. This function loads data from a JSON file into a dict.

Parameters **fname** (*str*) – JSON object or path to a JSON file

Notes

required keys: elevation_units, elevation, surface_coordinates_units, surface_easting, surface_northing

optional key: datum

datum [str] kb, dfe or rt. datum is not used in calculation

kb (kelly bushing), dfe (drill floor elevation), rt (rotary table)

elevation_units [str] datum elevation units

elevation [float] datum elevation in elevation_units above mean sea level

surface_coordinates_units [str] surface coordinate units of wellhead

surface_easting [float] wellhead surface location in surface_coordinates_units east of reference

surface_northing [float] wellhead surface location in surface_coordinates_units north of reference

Returns **header**

Return type dict

wellpathpy.**read_csv**(*fname*, *delimiter=','*, *skiprows=1*, ***kwargs*)

Read a deviation file in CSV format

A header row containing the column names *md*, *inc*, *azi* in that order is generally expected to be included as the first row in the file. By default, this header is skipped with the *skiprows* argument set to *1* but this can

be changed to *0* if no header is inclduded. The data must be ordered as *md*, *inc*, *azi* as the data cannot be distinguished numerically.

Parameters

- **fname** (*str*) – path to a CSV file with this format: `md,inc,azi 0,0,244 10,11, 220 50,43,254 150,78.5,254 252.5,90,359.9`
- **delimiter** (*str*) – the character used as a delimiter in the CSV
- **skiprows** (*int*) – number of rows to skip, normally the header row

Other Parameters ****kwargs** (All other keyword arguments are passed to *np.loadtxt*)

Returns **md, inc, azi** – md, inc and azi are of type np.ndarray

Return type tuple

Notes

md [float] measured depth (units not defined)

inc [float] well inclination in degrees from vertical

azi [float] well azimuth in degrees from Grid North

wellpathpy.**deviation_to_csv**(*fname*, *md*, *inc*, *azi*, *fmt='%.3f'*, *delimiter=','*, *header='md, inc, azi'*, ***kwargs*)

Write a log to a comma-separated values (csv) file.

Parameters

- **fname** (*str or file handle*) – file path or object the CSV will be written to.
- **md** (*array-like,*) – measured depth
- **inc** (*array-like,*) – inclination from vertical
- **azi** (*array-like,*) – azimuth from north
- **fmt** (*str*) – this is the fmt argument to numpy.savetxt, see: https://numpy.org/doc/stable/reference/generated/numpy.savetxt.html
- **delimiter** (*str*) – String or character separating columns.
- **header** (*str*) – String that will be written at the beginning of the file. Beware if changing the header that it does not change the order in which the data are written, which remains: *md*,'inc','azi'.

Other Parameters ****kwargs** (All other keyword arguments are passed to *np.savetxt*)

Notes

This function is totally unit unaware, the user is responsible to handle units.

Caution: deviation_to_csv uses Python write mode set to the default: 'w' therefore existing files will be overwritten.

wellpathpy.**position_to_csv**(*fname*, *depth*, *northing*, *easting*, *fmt='%.3f'*, *delimiter=','*, *header='easting, northing, depth'*, ***kwargs*)

Write a log to a comma-separated values (csv) file.

Parameters

- **fname** (*str or file handle*) – file path or object the CSV will be written to.
- **depth** (*array-like,*) – true vertical depth (tvd) or true vertical depth subsea (tvdss)
- **northing** (*array-like,*) – distance north of reference point
- **easting** (*array-like,*) – distance east of reference point,
- **fmt** (*str*) – this is the fmt argument to numpy.savetxt, see: https://numpy.org/doc/stable/reference/generated/numpy.savetxt.html
- **delimiter** (*str*) – String or character separating columns.
- **header** (*str*) – String that will be written at the beginning of the file. Beware if changing the header that it does not change the order in which the data are written, which remains: *easting*,'northing','depth'.

Other Parameters ****kwargs** (All other keyword arguments are passed to *np.savetxt*)

Notes

This function is totally unit unaware, the user is responsible to handle units.

Caution: position_to_csv uses Python write mode set to the default: 'w' therefore existing files will be overwritten.

class wellpathpy.**deviation**(*md*, *inc*, *azi*)

Deviation

The deviation is a glorified triple (md, inc, azi), with some interesting operations.

Notes

Glossary: md : measured depth inc : inclination (in degrees) azi : azimuth (in degrees)

minimum_curvature(*course_length=30*)

This function calls mincurve.minimum_curvature with self

Notes

You can access help with *wp.mincurve.minimum_curvature?* in *ipython*

radius_curvature()

This function calls rad_curv.radius_curvature with self

Notes

You can access help with *wp.rad_curv.radius_curvature?* in *ipython*

tan_method(*choice='avg'*)

This function calls tan.tan_method with self

Notes

You can access help with *wp.tan.tan_method?* in *ipython*

to_csv(*fname*, ***kwargs*)

This function calls write.deviation_to_csv with self

Notes

You can access help with *wp.write.deviation_to_csv?* in *ipython*

class wellpathpy.**position_log**(*src*, *depth*, *northing*, *easting*)
Position log

The position log is the computed positions of the well path. It has no memory of the method that created it, but it knows what deviation it came from. In its essence, it's a glorified triplet (tvd, northing, easting) with some interesting operations.

Notes

Glossary: tvd : true vertical depth

to_csv(*fname*, ***kwargs*)
This function calls write.position_to_csv with self

Notes

You can access help with *wp.write.position_to_csv?* in *ipython*

to_tvdss(*datum_elevation*, *inplace=False*)
This function calls location.location_to_tvdss with self

Notes

You can access help with *wp.location.to_tvdss?* in *ipython*

to_wellhead(*surface_northing*, *surface_easting*, *inplace=False*)
Create a new position log instance moved to the wellhead location

Parameters

- **surface_northing** (*array_like*) –
- **surface_easting** (*array_like*) –
- **inplace** (*bool*) –

to_zero(*surface_northing*, *surface_easting*, *inplace=False*)
Create a new position log instance moved to 0m North and 0m East

Parameters

- **surface_northing** (*array_like*) –
- **surface_easting** (*array_like*) –
- **inplace** (*bool*) –

class wellpathpy.**minimum_curvature**(*src*, *depth*, *n*, *e*, *dls*)

deviation()
Deviation survey

Compute an approximate deviation survey from the position log, i.e. the measured that would be convertable to this well path. It is assumed that inclination, azimuth, and measured-depth starts at 0.

Returns dev

Return type *deviation*

resample(*depths*)

Resample the position log onto a new measured-depth.

Parameters **depths** (*array_like*) – The measured depths to resample onto

Returns **resampled** – Resampled position log

Return type *minimum_curvature*

Examples

Resample onto a regular, 1m measured depth interval:

```
>>> depths = list(range(int(dev.md[-1]) + 1))
>>> resampled = pos.resample(depths = depths)
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index