well_profile

Release 0.3.10

Pro Well Plan AS

May 17, 2022

CONTENTS:

This is the official documentation of pwploads. Here you will find all the relevant information about any function included in this package.

CHAPTER

ONE

INSTALLATION

well_profile is written to be compatible with Python 3+. The best way to install is using pip.

```
$ pip install well_profile
```

This will make sure that all the dependencies are installed. This requirements are listed below.

1.1 Requirements

- numpy
- pandas
- openpyxl
- plotly

CHAPTER

TWO

CREATING A WELLBORE TRAJECTORY

well_profile.**get**(*mdt*, *profile='V'*, *build_angle=1*, *kop=0*, *eob=0*, *sod=0*, *eod=0*, *kop2=0*, *eob2=0*, ***kwargs*)

Generate a wellpath.

Parameters

- **mdt** (*num*) – target depth, m or ft
- **profile** (*str*) – 'V' for vertical, 'J' for J-type, 'S' for S-type, 'H1' for Horizontal single curve and 'H2' for Horizontal double curve
- **build_angle** (*num*) – building angle, °
- **kop** (*num*) – kick-off point, m or ft
- **eob** (*num*) – end of build, m or ft
- **sod** (*num*) – start of drop, m or ft
- **eod** (*num*) – end of drop, m or ft
- **kop2** (*num*) – kick-off point 2, m or ft
- **eob2** (*num*) – end of build 2, m or ft

Keyword Arguments

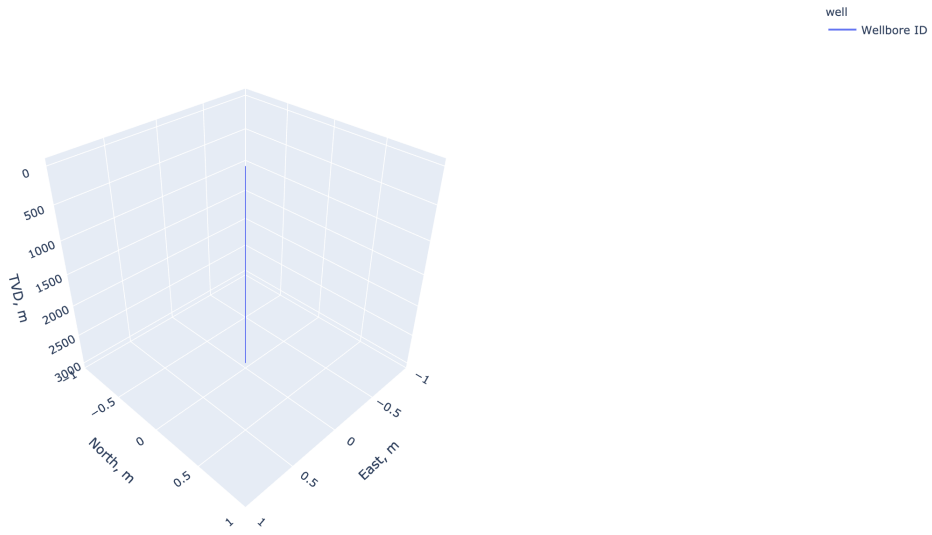
- **points** (*int*) – number of points
- **set_start** (*dict, None*) – set initial point in m {'north': 0, 'east': 0}.
- **change_azimuth** (*float, int, None*) – add specific degrees to azimuth values along the entire well.
- **set_info** (*dict, None*) – dict, {'dlsResolution', 'wellType': 'onshore'|'offshore', 'units': 'metric'|'english'}.

Returns **well** – A wellpath object with 3D position

Return type well object

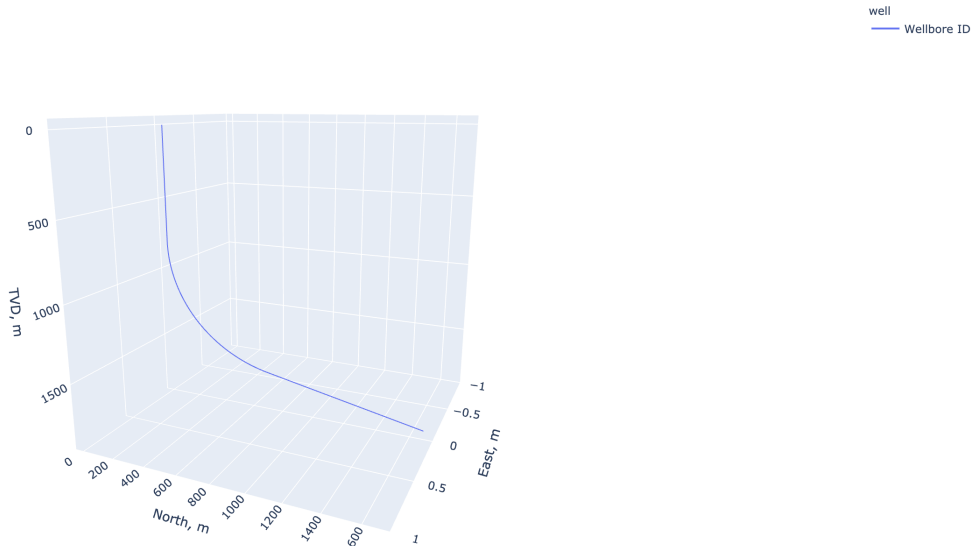
2.1 Vertical well

```
>>> import well_profile as wp
>>> well = wp.get(3000,   # define target depth (md) in m or ft
>>>               profile='V',    # set Vertical well profile
>>>               set_info={'dlsResolution': 30, 'wellType': 'offshore', 'units': 'metric
↪'},
>>>               # (optional) define the resolution for dls calculation, well type and␣
↪system of units 'metric'
>>>               # for meters or 'english' for feet
>>>               set_start={'north': 0, 'east': 0, 'depth': 0})    # (optional) set the␣
↪location of initial point
>>>               points=100,   # (optional) define number of points
>>> well.plot(names=['Wellbore ID']).show()
```



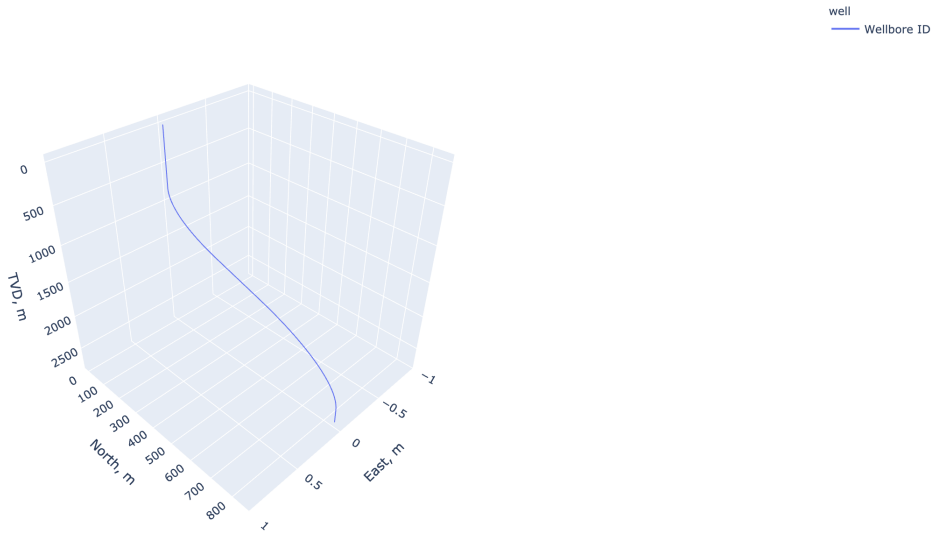
2.2 J-type well

```
>>> import well_profile as wp
>>> well = wp.get(3000,   # define target depth (md) in m or ft
>>>               profile='J',    # set J-type well profile
>>>               kop=800,    # set kick off point in m or ft
>>>               eob=2000,   # set end of build in m or ft
>>>               build_angle=78,   # set build angle in °
>>>               set_info={'dlsResolution': 30, 'wellType': 'offshore', 'units': 'metric
↪'},
>>>               # (optional) define the resolution for dls calculation, well type and␣
↪system of units 'metric'
>>>               # for meters or 'english' for feet
>>>               set_start={'north': 0, 'east': 0, 'depth': 0})    # (optional) set the␣
↪location of initial point
>>>               points=100,   # (optional) define number of points
>>> well.plot(names=['Wellbore ID']).show()
```



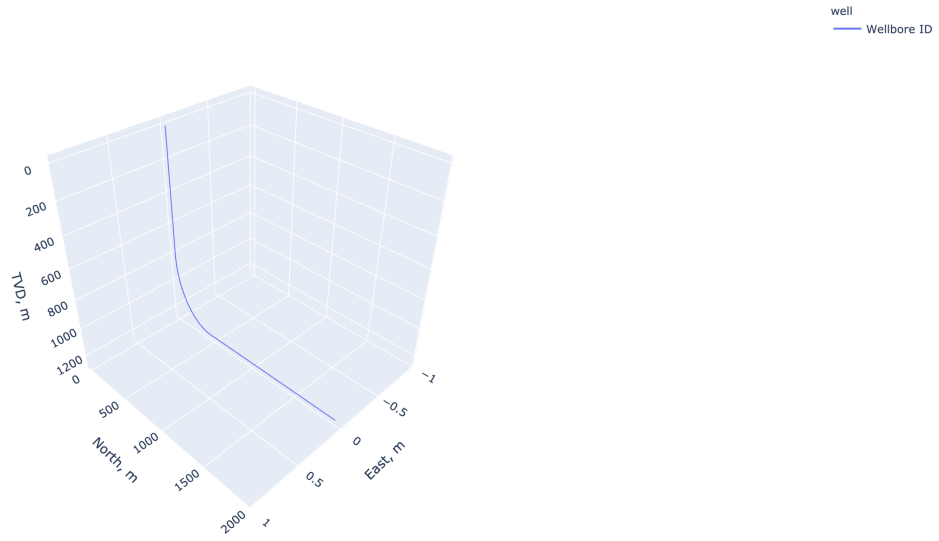
2.3 S-type well

```
>>> import well_profile as wp
>>> well = wp.get(3000,   # define target depth (md) in m or ft
>>>               profile='S',    # set S-type well profile
>>>               kop=800,    # set kick off point in m or ft
>>>               eob=1500,   # set end of build in m or ft
>>>               build_angle=45,   # set build angle in °
>>>               sod=1800,   # set start of drop in m or ft
>>>               eod=2800,   # set end of drop in m or ft
>>>               set_info={'dlsResolution': 30, 'wellType': 'offshore', 'units': 'metric
↪'},
>>>               # (optional) define the resolution for dls calculation, well type and␣
↪system of units 'metric'
>>>               # for meters or 'english' for feet
>>>               set_start={'north': 0, 'east': 0, 'depth': 0})    # (optional) set the␣
↪location of initial point
>>>               points=100,   # (optional) define number of points
>>> well.plot(names=['Wellbore ID']).show()
```



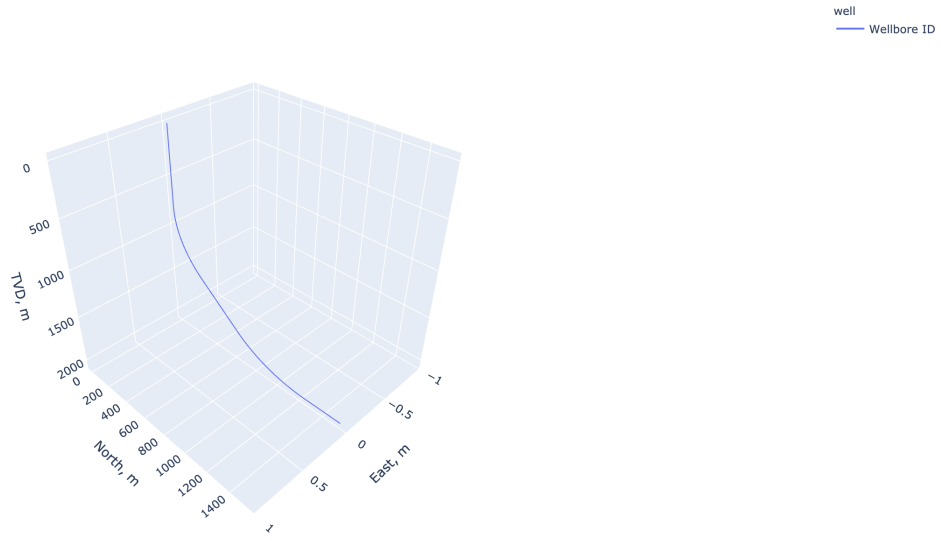
2.4 Horizontal single curve well

```
>>> import well_profile as wp
>>> well = wp.get(3000,   # define target depth (md) in m or ft
>>>               profile='H1',    # set horizontal single curve well profile
>>>               kop=800,    # set kick off point in m or ft
>>>               eob=1500,   # set end of build in m or ft
>>>               build_angle=45,   # set build angle in °
>>>               set_info={'dlsResolution': 30, 'wellType': 'offshore', 'units': 'metric
↪'},
>>>               # (optional) define the resolution for dls calculation, well type and␣
↪system of units 'metric'
>>>               # for meters or 'english' for feet
>>>               set_start={'north': 0, 'east': 0, 'depth': 0})    # (optional) set the␣
↪location of initial point
>>>               points=100,   # (optional) define number of points
>>> well.plot(names=['Wellbore ID']).show()
```



2.5 Horizontal double curve well

```
>>> import well_profile as wp
>>> well = wp.get(3000,   # define target depth (md) in m or ft
>>>               profile='H2',    # set horizontal double curve well profile
>>>               kop=800,    # set kick off point in m or ft
>>>               eob=1500,   # set end of build in m or ft
>>>               build_angle=45,   # set build angle in °
>>>               set_info={'dlsResolution': 30, 'wellType': 'offshore', 'units': 'metric
↪'},
>>>               # (optional) define the resolution for dls calculation, well type and␣
↪system of units 'metric'
>>>               # for meters or 'english' for feet
>>>               set_start={'north': 0, 'east': 0, 'depth': 0})    # (optional) set the␣
↪location of initial point
>>>               points=100,   # (optional) define number of points
>>> well.plot(names=['Wellbore ID']).show()
```



2.6 Using two points

This function allows to generate a wellbore trajectory by seeting kick-off point (KOP) and target.

`well_profile.`**`two_points`**(*points*, *inner_points=20*)

Parameters

- **points** – {'kickoff':{'north': num, 'east': num, 'tvd': num}, 'target': {'north': num, 'east': num, 'tvd': num}}
- **inner_points** – number of points between curved zone

Returns a wellpath object with 3D position

```
>>> import well_profile as wp
>>> well = wp.two_points({'kickoff': {'north': 0, 'east': 0, 'tvd': 100},
>>>                       'target': {'north': 500, 'east': 800, 'tvd': 800}})
```

CHAPTER

THREE

LOADING A WELLBORE TRAJECTORY

well_profile.**load**(*data*, ***kwargs*)

Load an existing wellpath.

Parameters **data** – Excel file, dataframe or list of dictionaries. Must contain at least md, inclination and azimuth. Can also contain tvd, northing and easting.

Keyword Arguments

- **set_start** (*dict, None*) – set initial point in m {'north': 0, 'east': 0}.
- **change_azimuth** (*float, int, None*) – add specific degrees to azimuth values along the entire well.
- **set_info** (*dict, None*) – dict, {'dlsResolution', 'wellType': 'onshore'|'offshore', 'units': 'metric'|'english'}.
- **inner_pts** (*num*) – include certain amount of inner points between survey stations.

Returns **well** – A wellpath object with 3D position

Return type well object

3.1 Load excel file

Example of a file is shown below. In case TVD is not included, the package calculates the values using the minimum curvature method. North and East coordinates also can be included.

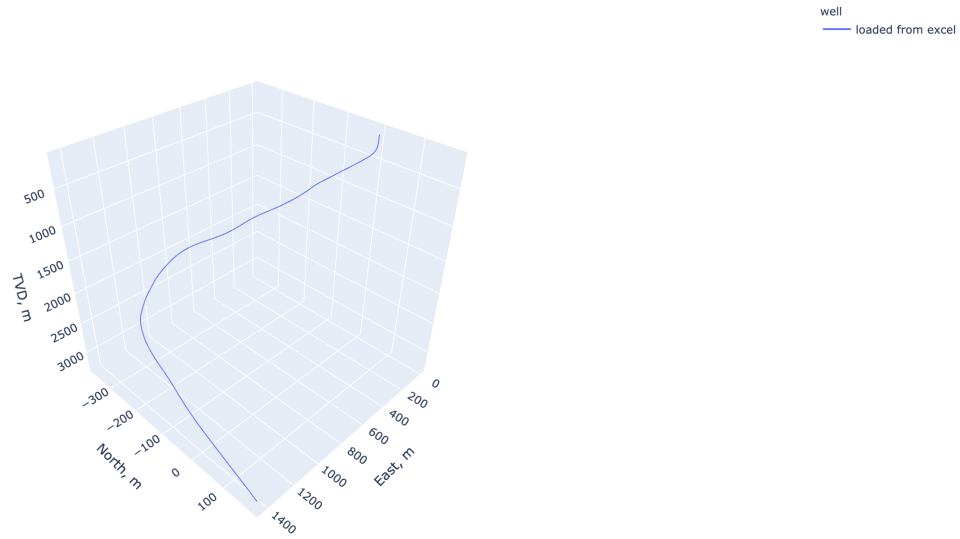
	A	B	C	D	E	F
1	md	inclination	azimuth	tvd		
2	145,9	0	0	145,9		
3	146	0	0	146		
4	154,9	0,16	221,03	154,9		
5	195,3	0,28	98,15	195,3		
6	221	0,51	63,39	221		
7	235,6	0,64	43,64	235,6		
8	276	1,51	188,03	276		
9	316,4	2,21	178,31	316,4		
10	356,7	4,11	173,29	356,6		
11	397,1	7,66	171,91	396,8		
12	437,5	9,13	178,88	436,8		
13	477,8	10,58	180,1	476,5		
14	518,2	11,35	177,18	516,1		
15	558,5	11,51	168,58	555,6		
16	598,9	12,38	160,99	595,2		

```
>>> import well_profile as wp
>>> well = wp.load('trajectory1.xlsx',   # loading excel file
>>>                set_info={'dlsResolution': 30, 'wellType': 'offshore', 'units': 'metric
↪'},
>>>                # (optional) define the resolution for dls calculation, well type and␣
↪system of units 'metric'
>>>                # for meters or 'english' for feet
```

(continues on next page)

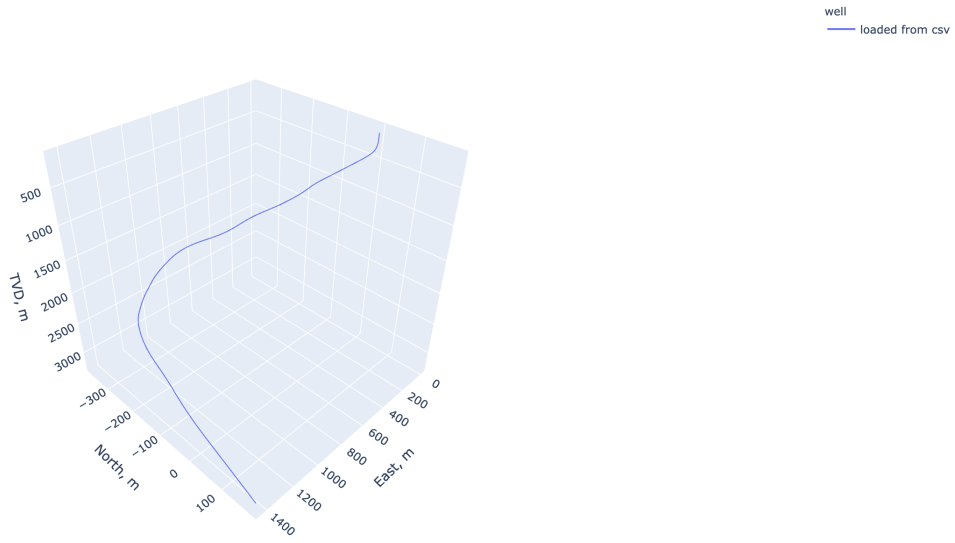
(continued from previous page)

```
>>>                 set_start={'north': 0, 'east': 0, 'depth': 0})  # (optional) set the␣
↪location of initial point
>>> well.plot(names=['loaded from excel']).show()
```



3.2 Load csv file

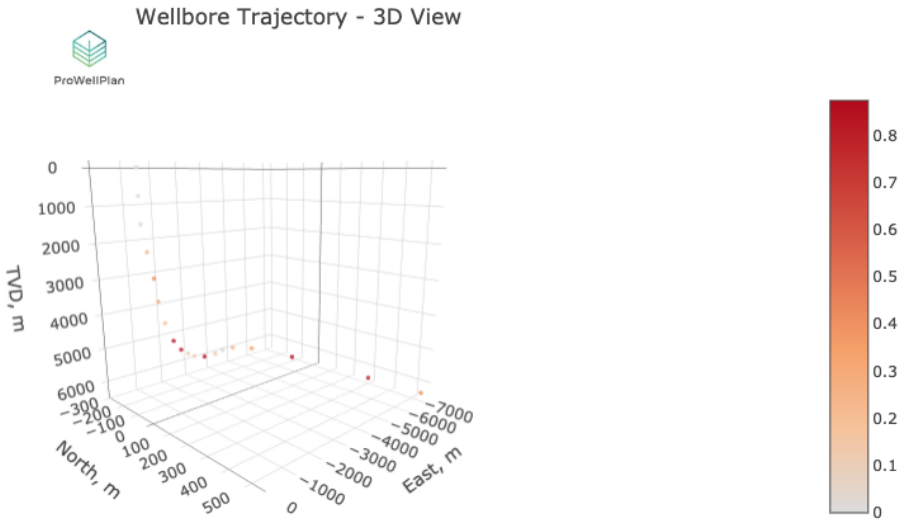
```
>>> import well_profile as wp           # loading csv file
>>> well = wp.load('trajectory1.csv',   # define target depth (md) in m or ft
>>>                 set_info={'dlsResolution': 30, 'wellType': 'offshore', 'units': 'metric
↪'},
>>>                 # (optional) define the resolution for dls calculation, well type and␣
↪system of units 'metric'
>>>                 # for meters or 'english' for feet
>>>                 set_start={'north': 0, 'east': 0, 'depth': 0})  # (optional) set the␣
↪location of initial point
>>> well.plot(names=['loaded from csv']).show()
```



3.3 Generate more data points from survey

With `well_profile` you can generate multiple data points between the survey stations by using the argument `inner_points`.

```
>>> import well_profile as wp
>>> well = wp.load('trajectory1.xlsx')   # loading file with only original survey points
>>> well.plot(style={'color': 'dls'}).show()
```

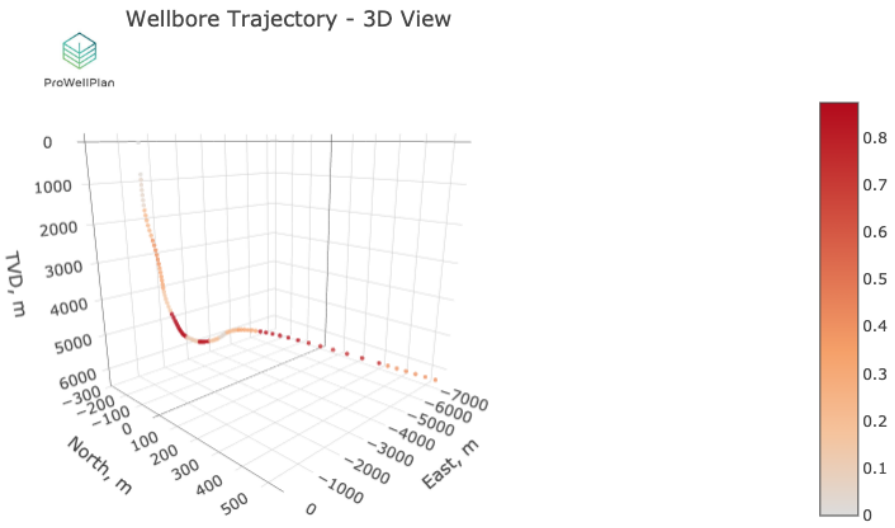



```
>>> import well_profile as wp
>>> well = wp.load('trajectory1.xlsx', inner_points=5)   # loading file with only␣
↪original survey points
```

(continues on next page)

(continued from previous page)

```
>>> well.plot(style={'color': 'dls'}).show()
```



CHAPTER

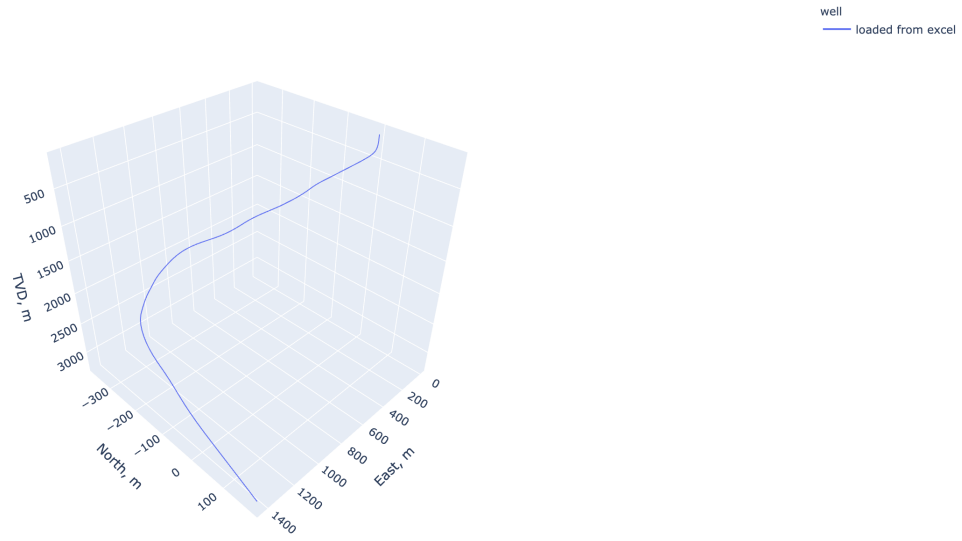
FOUR

INTERPOLATING POINTS ALONG THE TRAJECTORY

You can get all the relevant data for any specifc MD value along the wellbore trajectory.

```
>>> import well_profile as wp
>>> well = wp.load('trajectory1.xlsx')   # loading excel file
>>> wb1.get_point(3750)     # get data at 3750m MD
```

```
{'md': 3750,
 'dl': 0.5573264781490191,
 'inc': 52.9126735218509,
 'azi': 22.04,
 'north': 113.70285532280701,
 'east': 1340.3801459482888,
 'tvd': 3220.9879889843924,
 'pointType': 'interpolated',
 'sectionType': 'drop-off'}
```

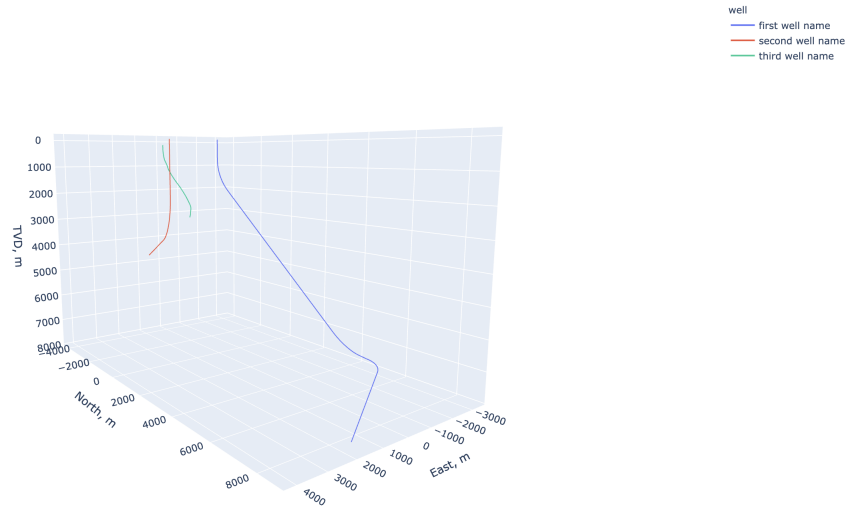


CHAPTER

FIVE

CHANGING THE AZIMUTH

```
>>> import well_profile as wp
>>> well_1 = wp.load('trajectory1.xlsx', change_azimuth=180)      # LOAD WELL 1 -->␣
↪Azimuth: + 180°
>>> well_2 = wp.get(6000, profile='J', kop=2000, eob=4000, build_angle=85,
>>>                 set_start={'east':2000}, change_azimuth=42)       # GET WELL 2 -->␣
↪Azimuth: + 42°
>>> well_3 = wp.load('trajectory2.xlsx', set_start={'north':-4000})        # LOAD WELL 3
>>> well_1.plot(add_well=[well_2, well_3],
>>>             names=['first well name',
>>>                    'second well name',
>>>                    'third well name']).show()        # Generate 3D plot for well 1␣
↪including wells 2 and 3
```



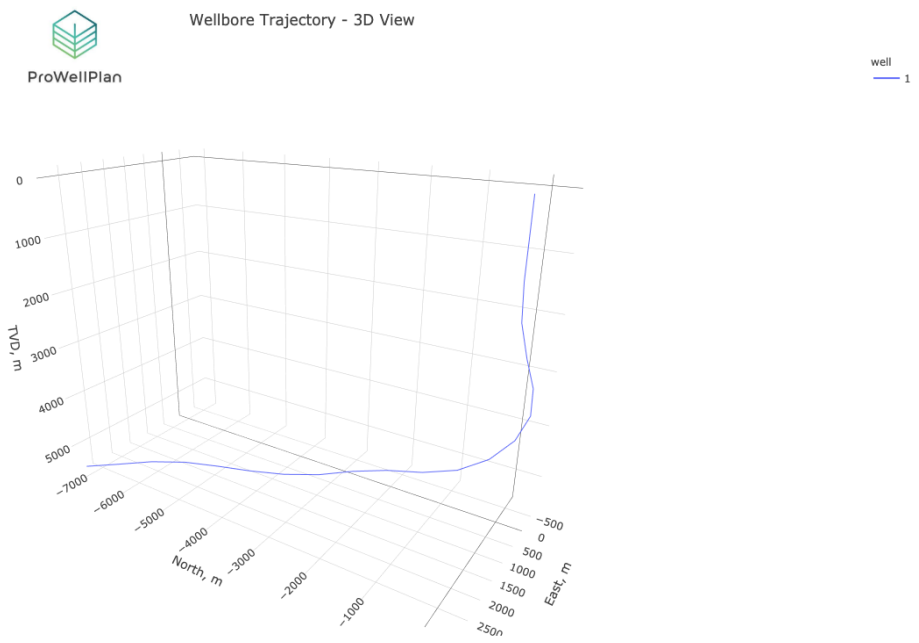
CHAPTER

SIX

PLOTTING TRAJECTORIES

6.1 3D view - single trajectory

```
>>> import well_profile as wp
>>> w1 = wp.load('wellbore1.xlsx')
>>> w1.plot().show()
```



6.2 3D view - plot multiple trajectories

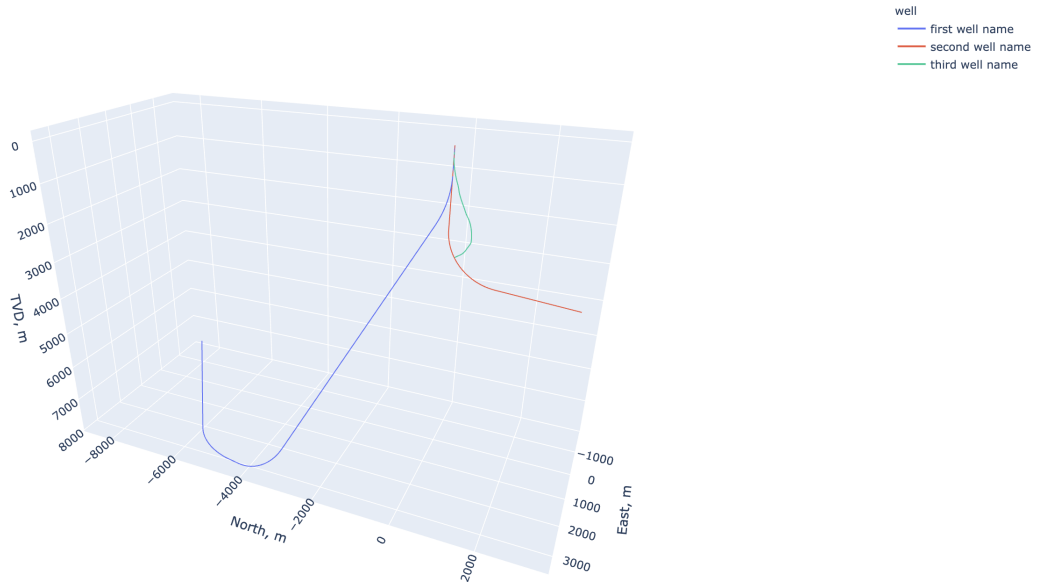
6.2.1 Same location

```
>>> import well_profile as wp
>>> well_1 = wp.load('trajectory1.xlsx')       # LOAD WELL 1
>>> well_2 = wp.get(5000, profile='J', kop=2000, eob=3000, build_angle=85)        # GET␣
↪WELL 2
```

(continues on next page)

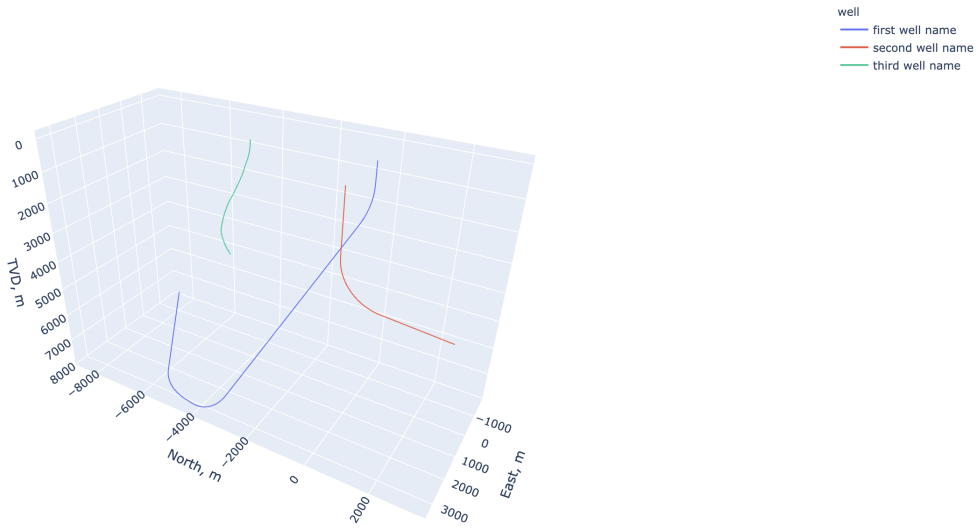
(continued from previous page)

```
>>> well_3 = wp.load('trajectory2.xlsx')         # LOAD WELL 3
>>> well_1.plot(add_well=[well_2, well_3],
>>>              names=['first well name',
>>>                     'second well name',
>>>                     'third well name']).show()        # Generate 3D plot for well 1␣
↪including wells 2 and 3
```



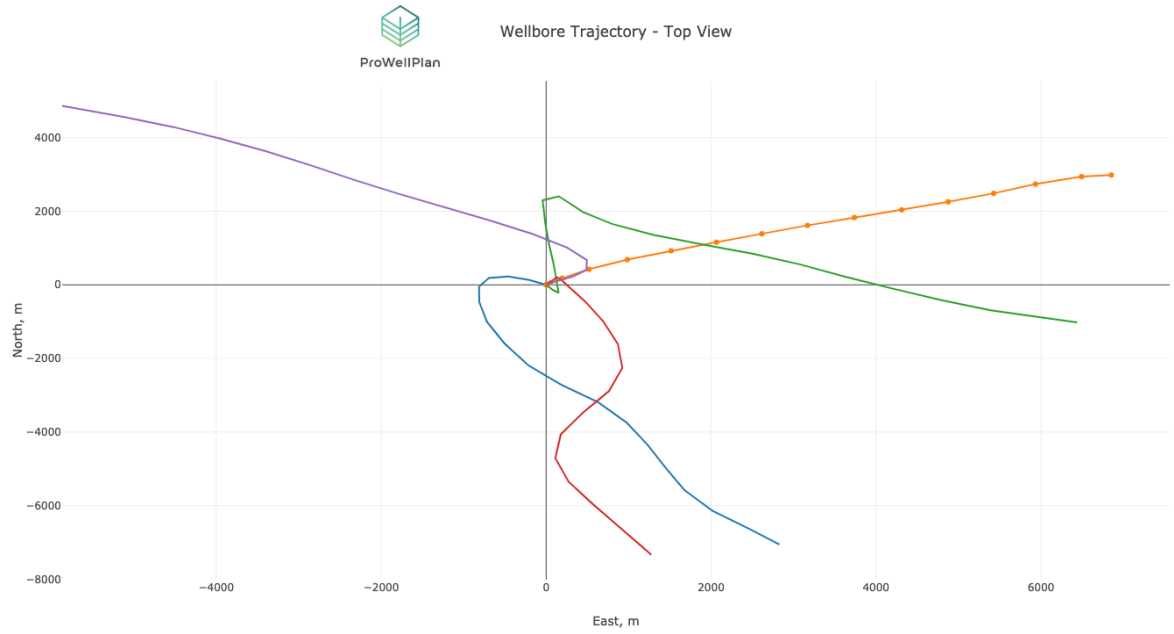
6.2.2 Different location

```
>>> import well_profile as wp
>>> well_1 = wp.load('trajectory1.xlsx')       # LOAD WELL 1
>>> well_2 = wp.get(5000, profile='J', kop=2000, eob=3000, build_angle=85, set_start={
↪'east':2000})        # GET WELL 2 --> North: 0 m, East: 2000 m
>>> well_3 = wp.load('trajectory2.xlsx', set_start={'north':-3000})        # LOAD WELL 3␣
↪--> North: -3000 m, East: 0 m
>>> well_1.plot(add_well=[well_2, well_3],
>>>              names=['first well name',
>>>                     'second well name',
>>>                     'third well name']).show()        # Generate 3D plot for well 1␣
↪including wells 2 and 3
```



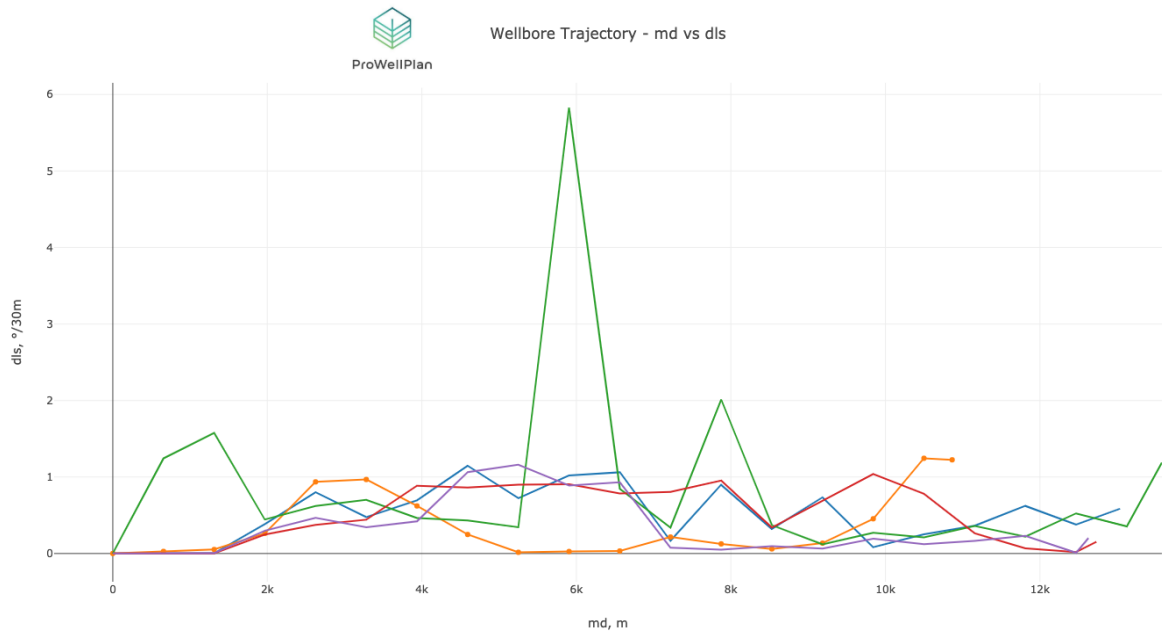
6.3 Top view

```
>>> import well_profile as wp
>>> w1 = wp.load('wellbore1.xlsx')
>>> w2 = wp.load('wellbore2.xlsx')
>>> w3 = wp.load('wellbore3.xlsx')
>>> w4 = wp.load('wellbore4.xlsx')
>>> w5 = wp.load('wellbore5.xlsx')
>>> w1.plot(add_well=[w2, w3, w4], plot_type='top').show()
```



6.4 Versus view

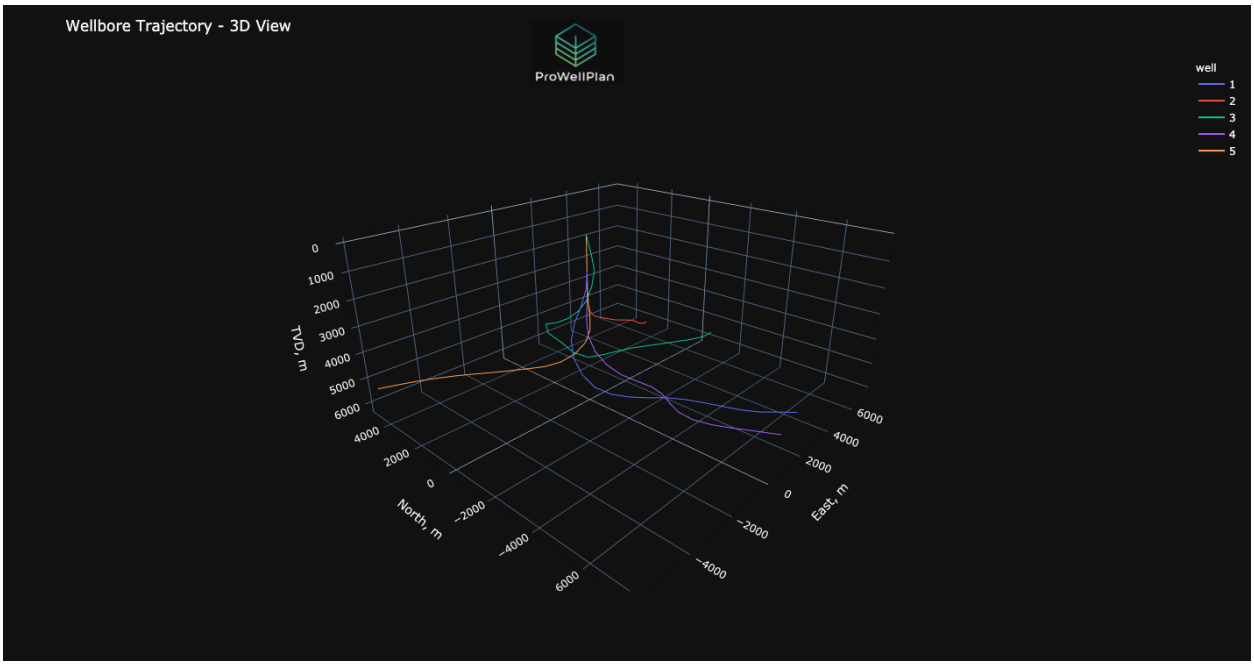
```
>>> import well_profile as wp
>>> w1 = wp.load('wellbore1.xlsx')
>>> w2 = wp.load('wellbore2.xlsx')
>>> w3 = wp.load('wellbore3.xlsx')
>>> w4 = wp.load('wellbore4.xlsx')
>>> w5 = wp.load('wellbore5.xlsx')
>>> w1.plot(add_well=[w2, w3, w4, w5], plot_type='vs', x_axis='md', y_axis='dls').show()
```



6.5 Dark mode

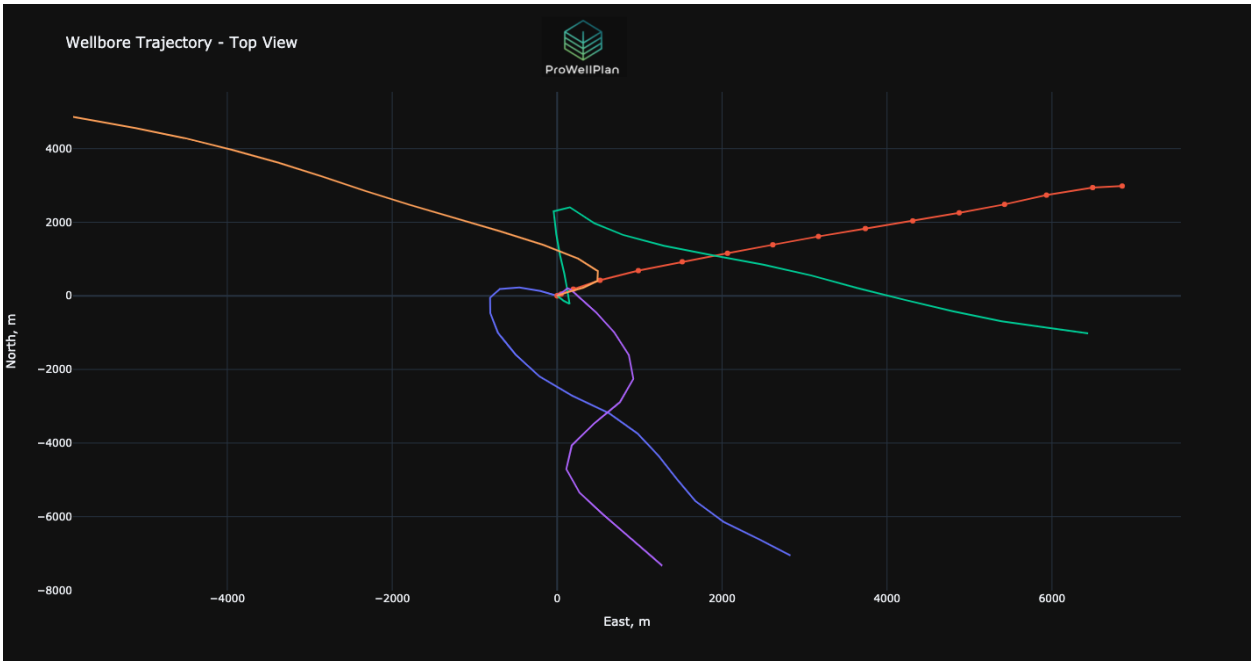
6.5.1 on 3D view

```
>>> import well_profile as wp
>>> w1 = wp.load('wellbore1.xlsx')
>>> w2 = wp.load('wellbore2.xlsx')
>>> w3 = wp.load('wellbore3.xlsx')
>>> w4 = wp.load('wellbore4.xlsx')
>>> w5 = wp.load('wellbore5.xlsx')
>>> w1.plot(add_well=[w2, w3, w4, w5], style={'darkMode': True}).show()
```



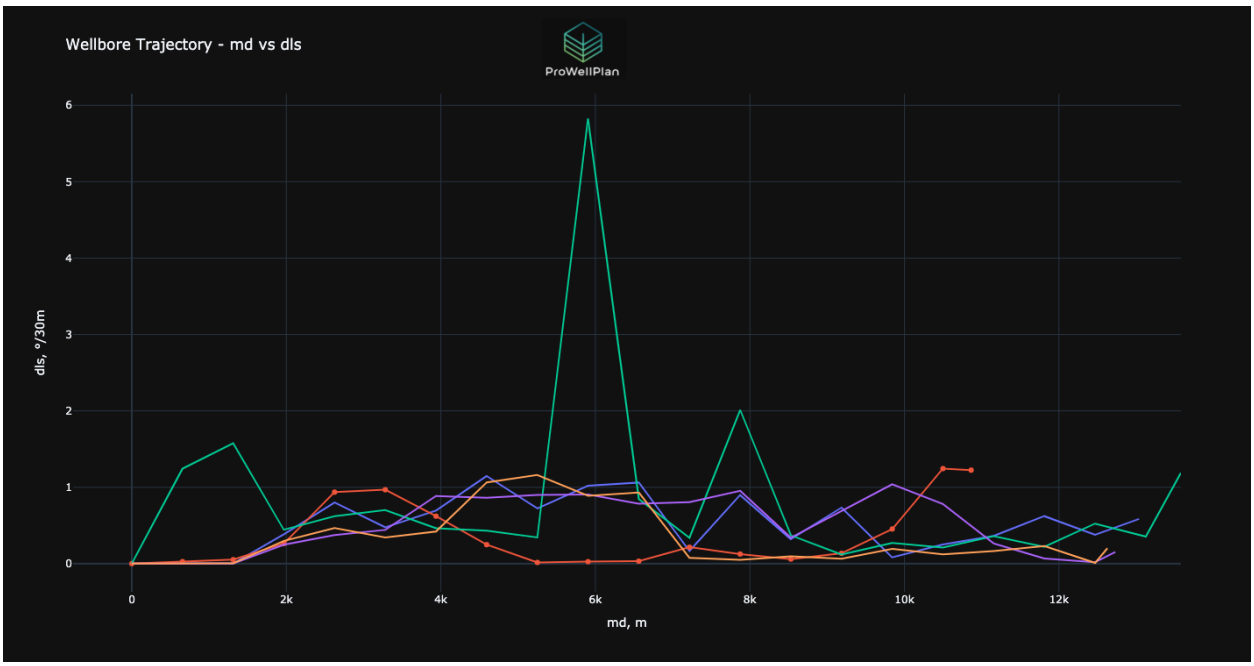
6.5.2 on Top view

```
>>> w1.plot(add_well=[w2, w3, w4, w5], plot_type='top', style={'darkMode': True}).show()
```



6.5.3 on Versus view

```
>>> w1.plot(add_well=[w2, w3, w4, w5], plot_type='vs', x_axis='md', y_axis='dls', style={
↪'darkMode': True}).show()
```



CHAPTER

SEVEN

ABOUT PRO WELL PLAN

Pro Well Plan offers an easy and effective well planning platform for the entire team. Check out our website to know more about us.

INDEX

G

L

T