In [ ]:
```
import pip
pip.main(["install", "openpyxl"])
pip.main(["install", "sympy"])
pip.main(["install", "xlsxwriter"])
pip.main(["install", "matplotlib"])
pip.main(["install", "numpy"])
```

In [ ]:
```
import pandas as pd

well_data = pd.ExcelFile('Well_Data.xlsx')
string_data = pd.read_excel(well_data, 'String_Data')
survey_data = pd.read_excel(well_data, 'Survey_Data')

pwd_data = pd.ExcelFile('Simulated_PWD_Data.xlsx')
pwd_table = pd.read_excel(pwd_data, 'Sheet1')
```

In [ ]:
```
pwd_table
```

Out[ ]:

	Q (gpm)	SPP (psi)	MD bit (ft)	TVD bit (ft)	simulated pressure at bit by power law model (psi)	MD Borehole (ft)	TVD Borehole (ft
0	474.031687	1402.004670	7849.07068	6355.03452	3505.652677	7849.07068	6355.0345
1	473.849437	1402.081482	7849.10349	6355.03452	3503.727704	7849.10349	6355.0345
2	473.982378	1402.155340	7849.10349	6355.03452	3501.945749	7849.10349	6355.0345
3	474.110302	1402.226411	7849.10349	6355.03452	3500.497555	7849.10349	6355.0345
4	474.233488	1402.160507	7849.13630	6355.03452	3499.547331	7849.13630	6355.0345
...	...	...	...	...	...	...	...
64759	523.593916	1661.043080	8206.37158	6355.10014	3526.933466	8206.37158	6355.1001
64760	523.496792	1660.958078	8206.37158	6355.10014	3525.193473	8206.37158	6355.1001
64761	523.396003	1660.979370	8206.37158	6355.10014	3525.833865	8206.37158	6355.1001
64762	523.599519	1661.001481	8206.37158	6355.10014	3526.897566	8206.37158	6355.1001
64763	523.811016	1661.308949	8206.37158	6355.10014	3525.346156	8206.37158	6355.1001

64764 rows × 8 columns

In [ ]:
```
L_c=2550.0
L_dp =6480.0
L_dc = 620.0


OD_c=13.725
OD_dp=5
OD_dc=8


ID_c=12.565
ID_dp=4.276
```

```
ID_dc=2.87
ID = 12.25



RHO = 8.824
PV = 12.0
YP = 12.0
Q=700
ID_list = [ID_c, ID_dp, ID_dc]
theta300=PV + YP
theta600=theta300 +PV
print('theta300',theta300)
print('theta600',theta600)
```

```
theta300 24.0
theta600 36.0
```

In [ ]:
```
import math
def pressure_drop_2 (k, n, Q, ID, PV):

    # Critical velocity
    V_c = (((5.82 * ((10.0)**4) * k) / RHO) ** (1/(2 - n))) \
        * ((1.6/ID_dp)*((3 * n+1) / 4 * n)) ** (n / (2-n))

    # Average Velocity
    V_avg = 24.5 * Q / (ID_dp **2)
    if(V_avg > V_c):  # flow is turbulent
        P = (8.91 * (10**(-5)) * (RHO ** 0.8) * (Q ** 1.8) \
             * ((PV ** 0.2) * L_dp))/(ID_dp ** 4.8)

    elif(V_avg < V_c): # flow is laminar
        P = (((1.6 * V_avg * (3*n+1))/(ID_dp * 4 * n)) ** n)\
             * (k * L_dp)/(300 * ID_dp)

    return P
```

In [ ]:
```
def pressure_drop_3(k, n, Q, ID, PV):
      #Pressure losses inside drill collars(P3)
      V_c = (((5.82 * ((10.0)**4.0) * k) / (RHO)) ** \
             (1.0/(2.0 - n))) * (((1.6*(3.0 * n+1.0))/(ID_dc*4.0*n) )) **

      V_avg = 24.5 * Q / (ID_dc**2)

      if(V_avg > V_c):  # flow is turbulent
            P3 = (((8.91 * (10**(-5)) * (RHO ** 0.8) * \
                    (Q ** 1.8) * ((PV ** 0.2) * L_dc)))/(ID_dc ** 4.8))

      elif(V_avg < V_c): # flow is laminar
            P3 = (((1.6 * V_avg * (3.0*n+1.0))\
                   /(ID_dc * 4.0 * n)) ** n) * (k * L_dc)/(300.0 * ID_dc)

      return P3
```

In [ ]:
```
def pressure_drop_4(k, n, Q, ID_dc, PV):

    #Pressure losses around drill collars(P4)
```

```
    V_c=(((3.878*(10**(4.0))*k)/RHO)**(1.0/(2.0-n)))\
        * (((2.4*((2.0*n)+1.0))/((ID-OD_dc)*(3*n)))**(n/(2.0-n)))

    V_avg = 24.5 * Q / ((ID_dc **2.0)-(OD_dc**2.0))
    if(V_avg < V_c): # flow is laminar
        P4  = ((((2.4 * V_avg * (2.0*n+1.0))\
                 /((ID - OD_dc) * 3.0 * n)) ** n)\
                      * (k * L_dc)/(300.0 * (ID - OD_dc)))
    else:
        P4 = (8.91 * (10**(-5)) * (RHO ** 0.8) * (Q ** 1.8) \
              * ((PV ** 0.2) * L_dc))/((ID-OD_dc) ** 3.0)*((ID+OD_dc) **

    return P4
```

In [ ]:
```
def pressure_drop_5(k, n, Q, ID, PV, L_c = L_c, L_dp = L_dp):
    #Annular losses (we determine in the open hole section and cased hole
    #Pressure losses around drillpipe. Cased hole section

    V_c = (((3.878 * ((10.0)**4.0) * k) / RHO) ** (1.0/(2.0 - n))) * \
        (((2.4/(ID_c-OD_dp))*((2.0 * n+1.0) / (3.0 * n))) ** (n / (2.0-n)

    V_avg = 24.5 * Q / ((ID_c **2)-(OD_dp**2))

    if(V_avg < V_c): # flow is laminar
        P_a = (((2.4 * V_avg * (2*n+1))/((ID_c-OD_dp) * 3 * n)) ** n)\
              * (k * L_c)/(300 * (ID_c-OD_dp))
    else:
        P_a = (8.91 * (10**(-5)) * (RHO ** 0.8) * (Q ** 1.8) * \
               ((PV ** 0.2) * L_c))/((ID_c-OD_dp) ** 3.0)*((ID_c+OD_dp) *

    #Open hole section


    V_c = (((3.878 * ((10.0)**4) * k) / RHO) ** (1/(2 - n))) * \
        (((2.4/(ID-OD_dp))*((2 * n+1) / (3 * n))) ** (n / (2-n)))
    V_avg = 24.5 * Q / ((ID_c **2)-(OD_dp**2))
    if(V_avg < V_c): # flow is laminar
        P_b = (((2.4 * V_avg * (2*n+1))/((ID-OD_dp) * 3 * n)) ** n) \
            * (k * L_dp)/(300 * (ID_c-OD_dp))
    else:
        P_b = (8.91 * (10**(-5)) * (RHO ** 0.8) * (Q ** 1.8) * \
               ((PV ** 0.2) * L_c))/((ID_c-OD_dp) ** 3.0)*((ID_c+OD_dp) *
    #Total pressure loss around drillpipe (P5)
    P5 = P_a +P_b

    return P5
```

In [ ]:
```
Q=700
# n is power law index
n = 3.32 * math.log(theta600/theta300,10)
print('n',n)
# consistency index
k = theta300/((511.0) ** n)
print('k',k)
list_for_table = []
header_table = ['Q', 'P1','P2','P3','P4','P5', 'P', 'ECD', 'BHP']   # to
```

```
#surface losses
P1 = 4.2*(10.0**(-5))*(RHO**(0.8))*(Q**(1.8))*(PV**(0.2))
print('P1',P1)
#pressure losses inside drillpipe(P2)
P2 = pressure_drop_2(k, n, Q, ID, PV)

#Pressure losses inside drill collars(P3)
P3 = pressure_drop_3(k, n, Q, ID, PV)

#Pressure losses around drill collars(P4)
P4 = pressure_drop_4(k, n, Q, ID, PV)


#Pressure losses around drillpipe. Cased hole section
P5 = pressure_drop_5(k, n, Q, ID, PV)

ECD = ((P5/(0.052 * pwd_table['TVD bit (ft)']))+ RHO)
BHP = (0.052 * RHO * pwd_table['TVD bit (ft)'])+ECD

P = P1 + P2 + P3 + P4 + P5

for i in range(len(ECD)):
    list_for_table.append([Q, P1, P2, P3, P4, P5, P, ECD[i], BHP[i]])

pressure_losses_table = pd.DataFrame(list_for_table, columns=header_table
```

```
n 0.5846229800648617
k 0.6263342252163234
P1 52.094804334237494
```

In [ ]:
```
#pressure_table.to_excel("pack_off_detection.xlsx", sheet_name="Pressure_
```

In [ ]:
```
D=7100
TVD_1=7050
TVD_2=6450
TVD_3=5900


list_for_table = []
header_table = ['Q', 'P_TVD_1', 'P_TVD_2','P_TVD_3','delta_P_TVD_1',\
                'delta_P_TVD_2', 'delta_P_TVD_3']   # to store all result

for i in range(150):
        Q = pwd_table['Q (gpm)'][i]

        P_TVD_1 = pressure_drop_5(k, n, Q, ID, PV, L_c=TVD_1, L_dp=TVD_1
        P_TVD_2 = pressure_drop_5(k, n, Q, ID, PV, L_c=TVD_2, L_dp=TVD_2
        P_TVD_3 = pressure_drop_5(k, n, Q, ID, PV, L_c=TVD_3, L_dp=TVD_3

      # P_TVD_1 = (0.052*TVD_p1)/P_TVD_1
        #P_TVD_2 = (0.052*TVD_p2)/P_TVD_2
       # P_TVD_3 = (0.052*TVD_p3)/ P_TVD_3

        delta_P_TVD_1=P_TVD_1- P_TVD_2
        delta_P_TVD_2=P_TVD_1- P_TVD_3
        delta_P_TVD_3=P_TVD_2-P_TVD_3


#ECD = ((P5/(0.052 * pwd_table['TVD bit (ft)']))+ RHO)
#BHP = (0.052 * RHO * pwd_table['TVD bit (ft)'])+ECD
```

```
list_for_table.append([Q, P_TVD_1, P_TVD_2, P_TVD_3,\
                       delta_P_TVD_1, delta_P_TVD_2, delta_P_TVD_3, ])
p_tvd_table = pd.DataFrame(list_for_table, columns=header_table)
#dataframe.to_excel("pack_off_detection.xlsx", sheet_name='P_TVD')
```

In [ ]:
```
def dfs_tabs(df_list, sheet_list, file_name):
    writer = pd.ExcelWriter(file_name,engine='xlsxwriter')
    for dataframe, sheet in zip(df_list, sheet_list):
        dataframe.to_excel(writer, sheet_name=sheet, startrow=0 , startco
    writer.save()

dfs = [pressure_losses_table, p_tvd_table]
sheets = ['Pressure_losses','P_TVD' ]

dfs_tabs(dfs, sheets, 'pack_off_detection.xlsx')
```

```
/tmp/ipykernel_8872/1786620611.py:5: FutureWarning: save is not part of
the public API, usage can give unexpected results and will be removed in
a future version
  writer.save()
```

Graph Part

In [ ]:
```
graph_data = pd.ExcelFile('graph_data.xlsx')
graph_table = pd.read_excel(graph_data, 'Sheet2')
graph_table
```

Out[ ]:

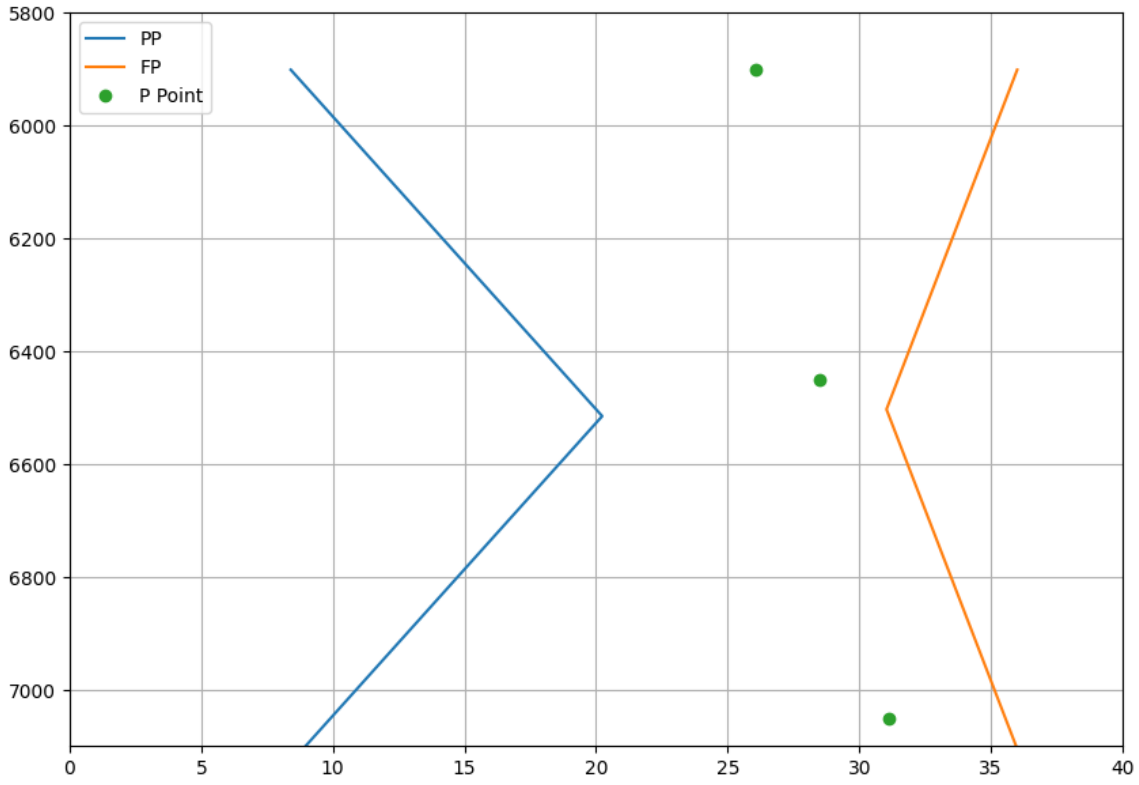
	Q	TVD	pore_pressure	frac_pressure	Unnamed: 4	Unnamed: 5	Unnamed: 6
0	700	5901.000000	8.400000	36.000000	NaN	NaN	NaN
1	700	5905.013378	8.477333	35.966887	NaN	36-31	5.000000
2	700	5909.026756	8.554667	35.933775	NaN	29-31	0.033113
3	700	5913.040134	8.632000	35.900662	NaN	NaN	NaN
4	700	5917.053512	8.709333	35.867550	NaN	NaN	NaN
...	...	...	...	...	...	...	...
296	700	7088.959867	9.173333	35.867550	NaN	NaN	NaN
297	700	7092.973245	9.096000	35.900662	NaN	NaN	NaN
298	700	7096.986623	9.018667	35.933775	NaN	NaN	NaN
299	700	7101.000001	8.941333	35.966887	NaN	NaN	NaN
300	700	7105.013379	8.864000	36.000000	NaN	NaN	NaN

301 rows × 7 columns

In [ ]:
```
D=7100
TVD_1=7050
TVD_2=6450
TVD_3=5900

for i in range(1):
        Q = pwd_table['Q (gpm)'][i]
```

```
        P_TVD_1 = pressure_drop_5(k, n, Q, ID, PV, L_c=TVD_1, L_dp=TVD_1
        P_TVD_2 = pressure_drop_5(k, n, Q, ID, PV, L_c=TVD_2, L_dp=TVD_2
        P_TVD_3 = pressure_drop_5(k, n, Q, ID, PV, L_c=TVD_3, L_dp=TVD_3

TVD = graph_table['TVD']
pore_p = graph_table['pore_pressure']
frac_p = graph_table['frac_pressure']
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(pore_p, TVD,  label = "PP"  )
plt.plot(frac_p, TVD, label = "FP")
plt.plot([P_TVD_1, P_TVD_2, P_TVD_3], [TVD_1, TVD_2, TVD_3],\
         'o', label = "P Point")
plt.legend()
plt.grid()
plt.ylim(7100.0 , 5800.0)
plt.xlim(0.0 , 40.0)

plt.rcParams["figure.figsize"] = (10,7)
plt.show()
```



In [ ]:
```
#Directional drilling

graph_data = pd.ExcelFile('graph_dadata.xlsx')
graph_table = pd.read_excel(graph_data, 'Sheet1')
graph_table['MD']
```

Out[ ]:
```
0        0.0
1        0.5
2        1.0
3        1.5
4        2.0
       ...
295    147.5
296    148.0
297    148.5
298    149.0
299    149.5
Name: MD, Length: 300, dtype: float64
```

In [ ]:
```
import numpy as np
import math
alpha = np.zeros(300)
betta = np.zeros(300)
DL = np.zeros(300)
RF = np.zeros(300)
T_V_D = np.zeros(300)
NS = np.zeros(300)
EW = np.zeros(300)
dir = np.zeros(300)
CL_D = np.zeros(300)
V_D = np.zeros(300)
dog_leg = np.zeros(300)

MD = graph_table['MD']
azimuth = graph_table['azimuth']
incl = graph_table['incl']
for i in range(1, 300):

        # alpha
        alpha[i] = incl[i]/57.2957795

        #betta
        betta[i] = azimuth[i]/57.2957795

        #DL
        DL[i] =math.acos(round(math.cos(alpha[i])*math.cos(alpha[i-1])+\
                math.sin(alpha[i])*\
                math.sin(alpha[i-1])*math.cos(betta[i]- betta[i-1]), 5))

        #RF
        if(DL[i] == 0):
                RF[i] = 1
        else:
                RF[i] = math.tan(DL[i]/2)/(DL[i]/2)

        #T_V_D
        # #for directional wells TVD and Measured depth are not same so w
        T_V_D [i]=(((MD[i]-MD[i-1])/2)*(math.cos(alpha[i])+\
                        math.cos(alpha[i-1])) *RF[i])+T_V_D[i-1]
        #print(MD[i])
        #for sections
        #North_South
        NS[i] = NS[i-1] + ((MD[i] - MD[i-1])/2) * (math.sin(alpha[i-1])\
                 * math.cos(betta[i-1]) + math.sin(alpha[i]) * math.cos(b
```

```
        #East West
        EW[i] = EW[i-1] + ((MD[i] - MD[i-1])/2) * (math.sin(alpha[i-1])\
                 * math.sin(betta[i-1]) + math.sin(alpha[i]) * math.sin(b

        # dir
        if(NS[i]==0 and EW[i] == 0):
                dir[i] = 0
        else:
                if(NS[i]>0):
                        if(EW[i]>0):
                                dir[i] = math.atan(EW[i]/NS[i]) * 57.2957
                        else:
                                dir[i] = 360 + math.atan(EW[i]/NS[i]) * 5
                else:
                        dir[i] = 180 + math.atan(EW[i]/NS[i]) * 57.295779


        #closure drift
        CL_D[i] = math.sqrt(EW[i] * EW[i] + NS[i] * NS[i])

        #vertical drift
        V_D[i] = CL_D[i] * math.cos((200-dir[i])/57.29577951)

        #dog leg
        dog_leg[i] = (math.acos(math.sin(incl[i-1]/57.3) * math.sin(incl[
                 * math.cos((azimuth[i] - azimuth[i-1])/57.3) \
                + (math.cos(incl[i-1]/57.3)*math.cos(incl[i]/57.3)))*57.3
                * (100/(MD[i] - MD[i-1]))

dataf = pd.DataFrame({'alpha': alpha,
                      'betta': betta,
                      'DL': DL,
                      'RF': RF,
                      'T_V_D': T_V_D,
                      'NS': NS,
                      'EW': EW,
                      'dir': dir,
                      'closure': CL_D,
                      'Vertical': V_D,
                      'dog_leg': dog_leg})
dataf
```

Out[ ]:

	alpha	betta	DL	RF	T_V_D	NS	EW	dir	closure	Vertical	do
0	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.0
1	0.000000	0.0	0.0	1.0	0.500000	0.000000	0.0	0.0	0.000000	-0.000000	0.0
2	0.000000	0.0	0.0	1.0	1.000000	0.000000	0.0	0.0	0.000000	-0.000000	0.0
3	0.000000	0.0	0.0	1.0	1.500000	0.000000	0.0	0.0	0.000000	-0.000000	0.0
4	0.000000	0.0	0.0	1.0	2.000000	0.000000	0.0	0.0	0.000000	-0.000000	0.0
...	...	...	...	...	...	...	...	...	...	...	
295	1.570796	0.0	0.0	1.0	100.751494	74.885235	0.0	360.0	74.885235	-70.369103	0.0
296	1.570796	0.0	0.0	1.0	100.751494	75.385235	0.0	360.0	75.385235	-70.838949	0.0
297	1.570796	0.0	0.0	1.0	100.751494	75.885235	0.0	360.0	75.885235	-71.308795	0.0
298	1.570796	0.0	0.0	1.0	100.751494	76.385235	0.0	360.0	76.385235	-71.778641	0.0
299	1.570796	0.0	0.0	1.0	100.751494	76.885235	0.0	360.0	76.885235	-72.248488	0.0

300 rows × 11 columns

In [ ]:

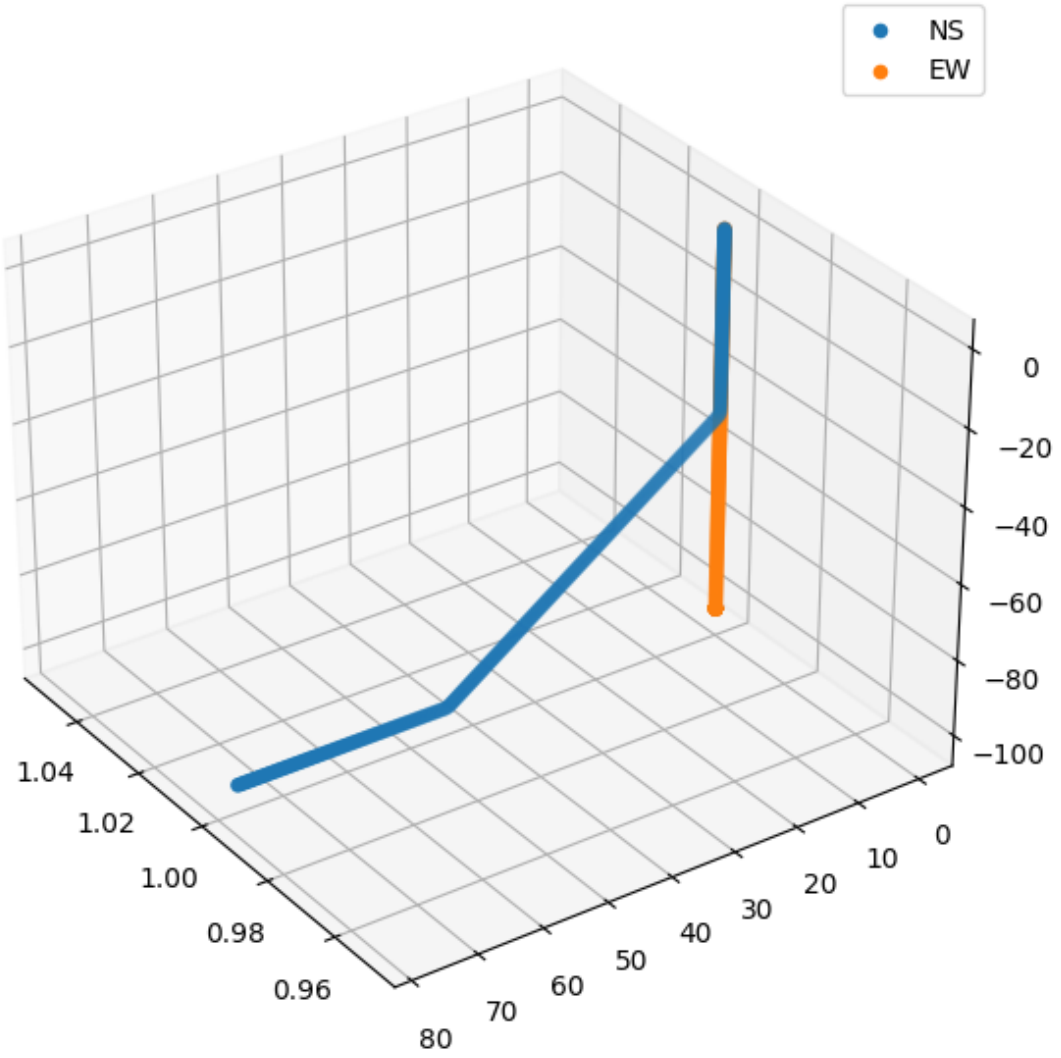
```
#pressure determination at given points
D=7830
TVD_1_new=7780
TVD_2_new=7180
TVD_3_new=6630


list_for_table = []
header_table = ['Q',
                'P_TVD_1_new',
                'P_TVD_2_new',
                'P_TVD_3_new',
                'delta_P_TVD_1_new',
                'delta_P_TVD_2_new',
                'delta_P_TVD_3_new']   # to store all results

for i in range(150):
        Q = pwd_table['Q (gpm)'][i]

        P_TVD_1_new = pressure_drop_5(k, n, Q, ID, PV, L_c=TVD_1_new, L_d
        P_TVD_2_new = pressure_drop_5(k, n, Q, ID, PV, L_c=TVD_2_new, L_d
        P_TVD_3_new = pressure_drop_5(k, n, Q, ID, PV, L_c=TVD_3_new, L_d



        delta_P_TVD_1_new=P_TVD_1_new- P_TVD_2_new
        delta_P_TVD_2_new=P_TVD_1_new- P_TVD_3_new
        delta_P_TVD_3_new=P_TVD_2_new-P_TVD_3_new



        ECD = ((P5/(0.052 * pwd_table['TVD bit (ft)']))+ RHO)
        BHP = (0.052 * RHO * pwd_table['TVD bit (ft)'])+ECD

        list_for_table.append([Q, P_TVD_1_new, P_TVD_2_new, \
                P_TVD_3_new,delta_P_TVD_1_new, delta_P_TVD_2_new, \
                        delta_P_TVD_3_new])
```

```
        p_tvd_table_new = pd.DataFrame(list_for_table, columns=header_tab
        #dataframe.to_excel("pack_off_detection.xlsx", sheet_name='P_TVD_
```

In [ ]:
```
def dfs_tabs(df_list, sheet_list, file_name):
    writer = pd.ExcelWriter(file_name,engine='xlsxwriter')
    for dataframe, sheet in zip(df_list, sheet_list):
        dataframe.to_excel(writer, sheet_name=sheet, startrow=0 , startco
    writer.save()

dfs = [pressure_losses_table, p_tvd_table, dataf, p_tvd_table_new]
sheets = ['Pressure_losses','P_TVD', 'directional_drilling', 'P_TVD_New']

dfs_tabs(dfs, sheets, 'pack_off_detection.xlsx')
```

```
/tmp/ipykernel_8872/679435215.py:5: FutureWarning: save is not part of t
he public API, usage can give unexpected results and will be removed in
a future version
  writer.save()
```

In [ ]:
```
import matplotlib.pyplot as plt


ax = plt.axes(projection='3d')

# Data for a three-dimensional line
xline = np.ones(len(NS))
yline = NS
zline = -T_V_D
ax.scatter3D(xline, yline, zline, label = "NS")

xline = np.ones(len(NS))
yline = EW
zline = -T_V_D
ax.scatter3D(xline, yline, zline, label = "EW")

# plt.plot(NS, T_V_D,  label = "NS"  )
# plt.plot(EW, T_V_D, label = "EW")
# plt.legend()
# plt.grid()
# plt.ylim(125.0 , 0.0)
# plt.xlim(-50.0 , 100.0)

ax.view_init(30, 145)
plt.legend()

plt.rcParams["figure.figsize"] = (10,7)
plt.show()
```



In [ ]:

In [ ]: