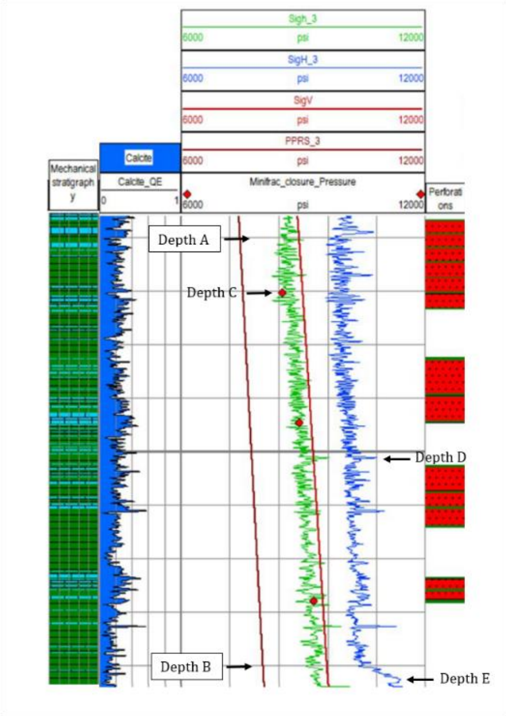
UNIVERSITY OF TEXAS AT AUSTIN
PGE 383 - ADVANCED GEOMECHANICS FALL 2023
Renato Espirito Basso Poli (rep2656) 08/30/2023

WEEKLY PROJECT #1
Subsurface Stresses, Stress Tensor and Invariants

Exercise 1: Reading a stress log



1. Calculate the average pore pressure (PPRS_3) gradient between depth A and depth B in [MPa/km] and [psi/ft]

	Pp (psi)	h (m)	h (ft)
A	7414	0	1
B	7935	200	657
DELTA	520	200	656
Pp Grad	0.79	psi/ft	

2. Calculate the average vertical stress (SigV) gradient between depth A and depth B

	Sv(psi)	H(m)	h_ft
A	8882	0	0
B	9530	200	658
DELTA	648		658
Sv Grad	0.99	psi/ft	

3. Calculate a reasonable guess for depth A

Assuming that the Sv gradient is constant		
Sv Grad	0.99	psi/ft
Sv(psi) @ A	8882	psi
H	9014	ft
	2747	m

4. Write out the principal stress tensors (as matrices 3x3) at depths A, B, C, D and E assuming vertical stress is a principal stress

The tensors below are ordered by the depth of the point.

TENSORS OF TOTAL STRESSES (psi)						
POINT A - STRIKE SLIP				POINT B - STRIKE SLIP		
9813	0	0		10345	0	0
0	8882	0		0	9530	0
0	0	8509		0	0	9053
POINT C - STRIKE SLIP				POINT E - STRIKE SLIP		
9993	0	0		11322	0	0
0	8969	0		0	9540	0
0	0	8468		0	0	9249
POINT D - REVERSE						
10742	0	0				
0	9578	0				
0	0	9205				

5. Classify A, B, C, D and E according to stress regime (Normal, Strike Slip, Reverse Faulting)

The points are ordered by depth. Total stress (S_i) readings, in psi, from the data provided.

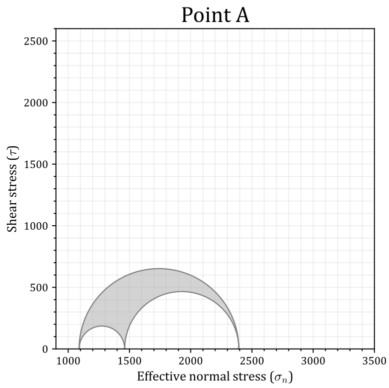
Point	H (m)	SV	SHMax	Shmin		FAULT STRESS REGIME
A	0	8882	9813	8509		**STRIKE SLIP**
C	25	8969	9993	8468		**STRIKE SLIP**
D	103	9205	10742	9578		**REVERSE**
B	200	9530	10345	9053		**STRIKE SLIP**
E	207	9540	11322	9249		**STRIKE SLIP**

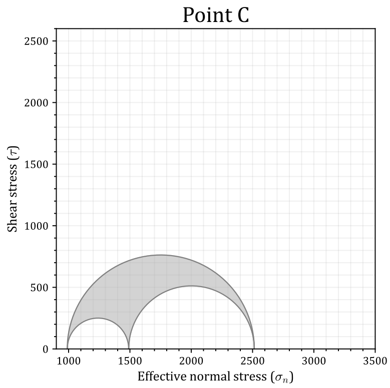
6. Plot 3D Mohr circles of effective stresses for A, B, C, D and E

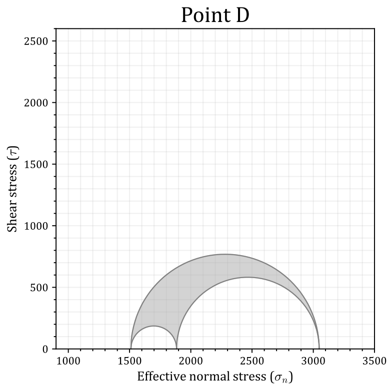
Calculation of the effective principal stresses ($\sigma_i = S_i - P$):

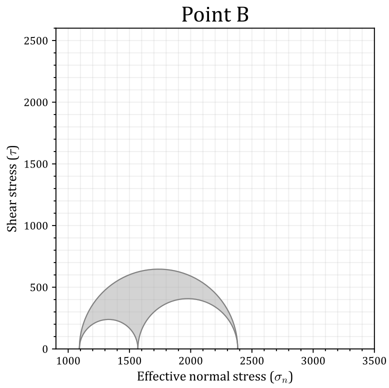
				Effective stresses (psi)		
	H(m)	**Pp(psi)**		**Sig_v**	**Sig_hmax**	**Sig_hmin**
A	0	7420		1461	9813	7048
C	25	7479		1490	9993	6978
D	100	7694		1511	10742	8067
B	200	7961		1569	10345	7484
E	207	7988		1552	11322	7697

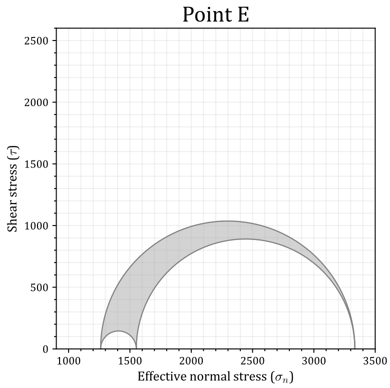
The points are ordered by depth.



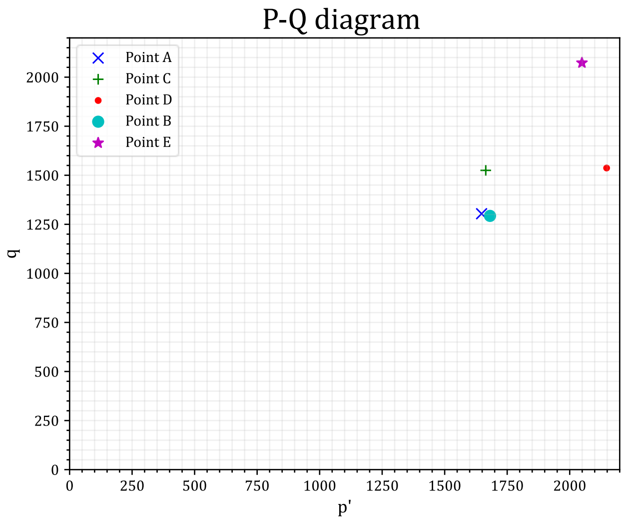




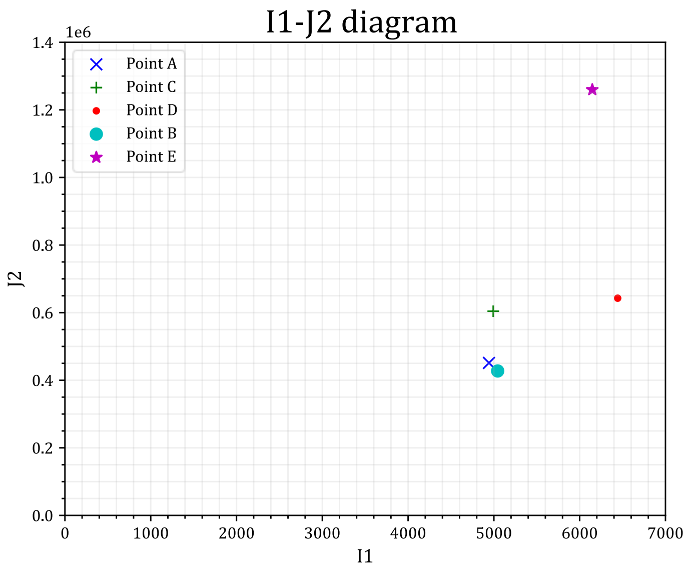




7. Plot $p - q$ points for A, B, C, D and E



8. Plot I_1-J_2 points for A, B, C, D and E

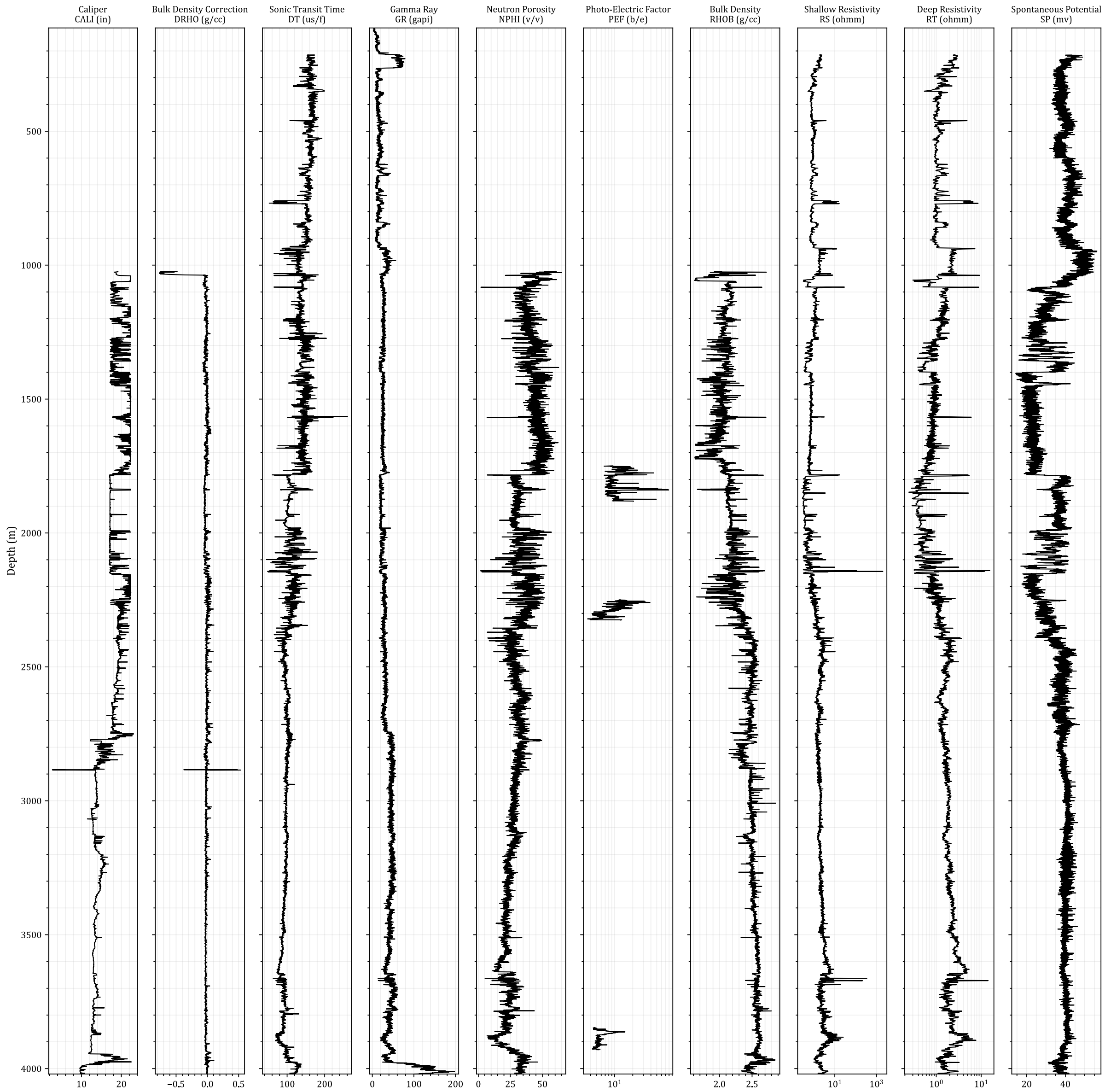


9. In which direction would a hydraulic fracture open-up in the interval under study in this formation? Justify.

A hydraulic fracture would open perpendicular to the minimum principal stress of the system. As most of the well is in strike slip faulting condition, the minimum principal stress is horizontal, and a vertical fracture is expected. Near "Point D", however, we observe a reverse faulting condition, meaning that the minimum principal stress is vertical, and it is possible for a horizontal fracture to initiate close to that position.

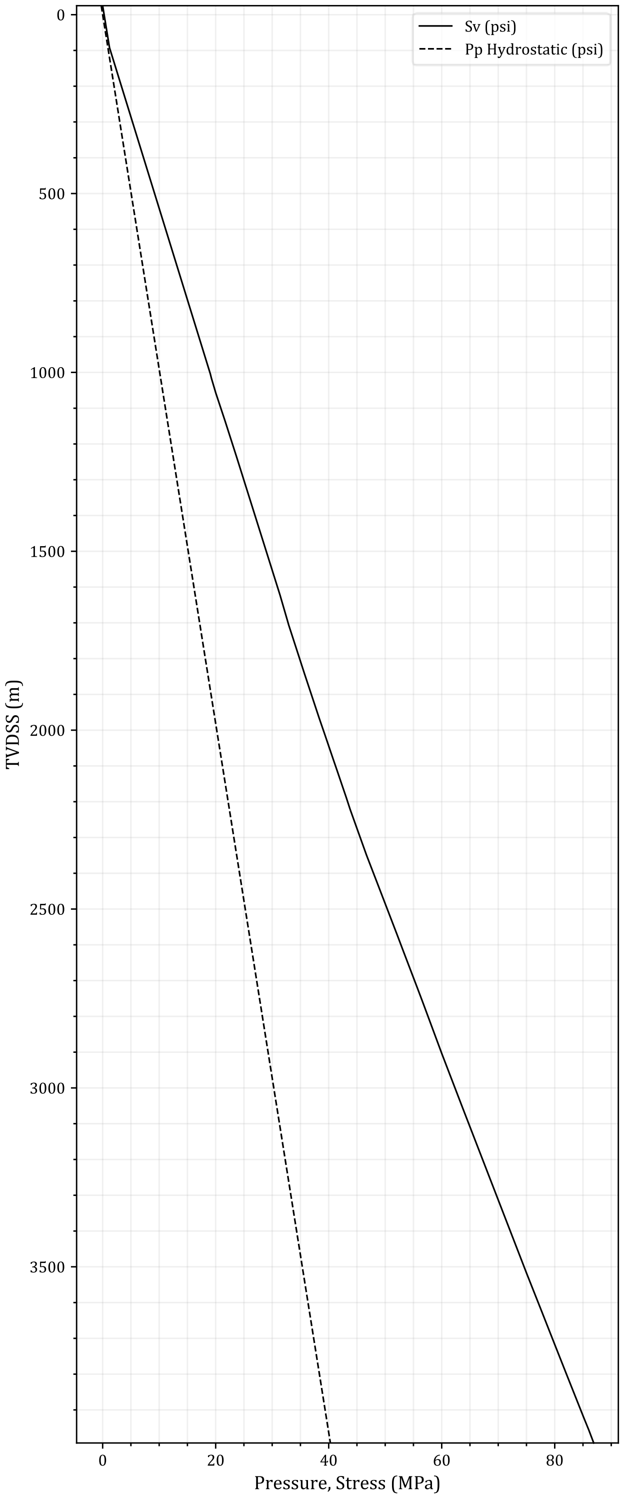
Exercise 2: Computing total vertical stress

1. Plot all available tracks with depth in the y-axis.



2. Calculate and plot vertical stress using the density log.

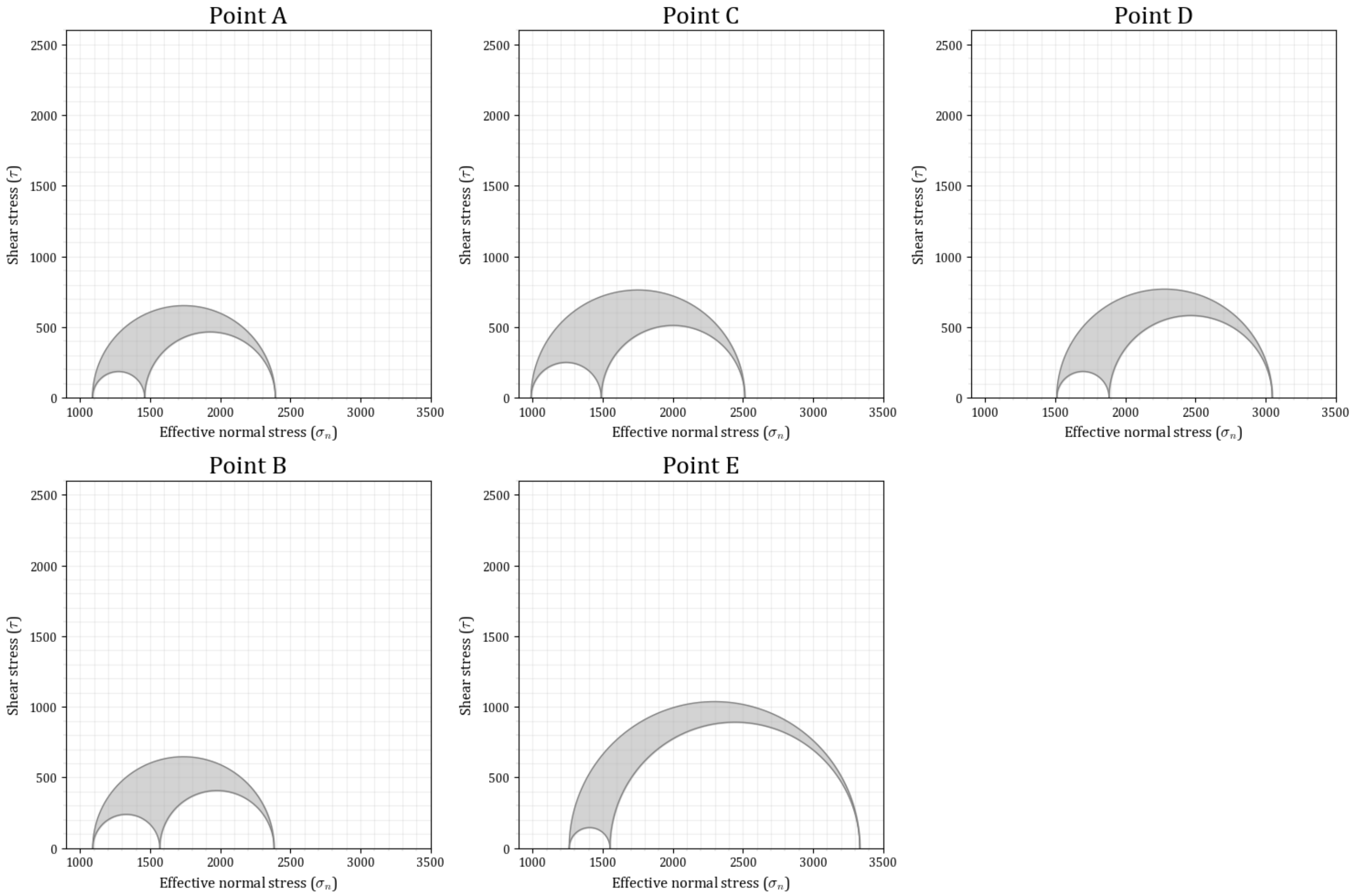
3. Calculate and plot hypothetical 'hydrostatic' pore pressure.



Python code used in this assignment:

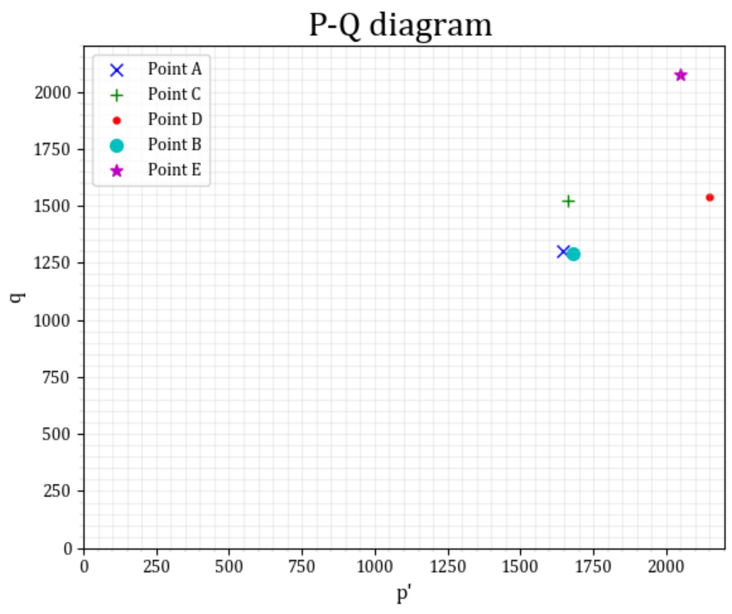
Draw Mohr diagrams

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('default')   ## reset!
plt.style.use('paper.mplstyle')

def shear( sig_n, s1, s2 ) :
    """
    Creates a shear series for given values of principal stresses and sig_n
    """
    center = (s1 + s2)/2
    radius = (s1 - s2)/2
    dx = sig_n - center
    tau_sq = radius * radius - dx * dx
    tau_sq[ tau_sq < 0 ] = None
    return np.sqrt( tau_sq )

def mohr(s1, s2, s3, ax, title="") :
    """
    Plot Mohr diagram
    """
    # Setup data
    npts = 1000

    [s1,s2,s3] = sorted([s1,s2,s3], reverse=True)
    step = (s1-s3)/npts
    sig_n = np.sort( np.append( np.arange(s3,s1,step), [s1, s2, s3]) )

    s12_tau = shear( sig_n, s1, s2 )
    s13_tau = shear( sig_n, s1, s3 )
    s23_tau = shear( sig_n, s2, s3 )

    # Do the plotting stuff
    ax.plot(sig_n, s12_tau, c='gray')
    ax.plot(sig_n, s23_tau, c='gray')
    ax.plot(sig_n, s13_tau, c='gray')

    ax.fill_between(sig_n, s12_tau, s13_tau, color='lightgray')
    ax.fill_between(sig_n, s23_tau, s13_tau, color='lightgray' )

    min, max = 900, 3500
    ax.set_xlim(min, max)
    ax.set_ylim(0, max-min)
    ax.set_title(title, fontsize=20)
    ax.set_ylabel("Shear stress ($\\tau$)")
    ax.set_xlabel("Effective normal stress ($\sigma_n$)")

    return ax

# Do the plotting
fig,[[ax11,ax12,ax13], [ax21,ax22,ax23]] = plt.subplots(2,3)
fig.set_size_inches(15,10)
ax23.remove()

mohr(1461,2393,1089, ax11, "Point A")
mohr(1490,2514,989,  ax12, "Point C")
mohr(1511,3048,1884, ax13, "Point D")
mohr(1569,2384,1091, ax21, "Point B")
mohr(1552,3334,1261, ax22, "Point E")

# Format and save.
fig.tight_layout()
fig.savefig('HW1_MOHR.svg')
```


Point A
Point C
Point D
Point B
Point E
Shear stress (τ)
Effective normal stress (σ_n)
0
500
1000
1500
2000
2500
3000
3500

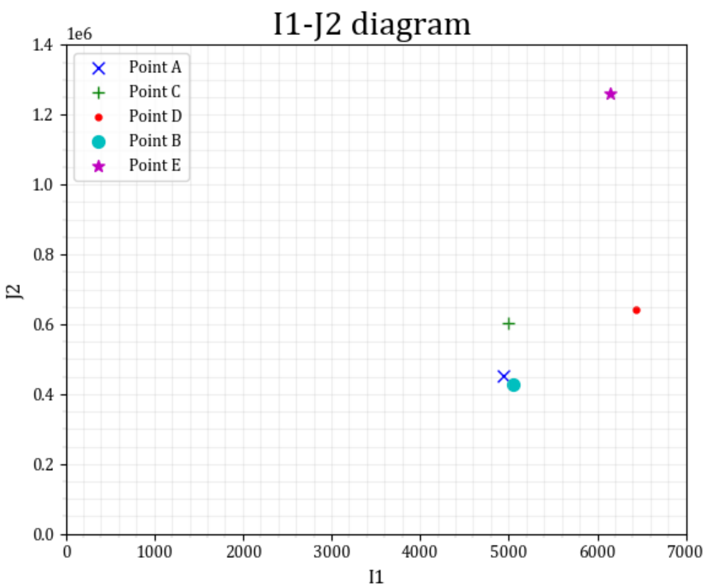
Draw p-q points

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('default')   ## reset!
plt.style.use('paper.mplstyle')
import itertools
mrk = itertools.cycle(('x', '+', '.', 'o', '*'))

def pq(s1, s2, s3, ax, title="") :
    """
    Plot PQ diagram using Pandas DF
    """
    [s1,s2,s3] = sorted([s1,s2,s3], reverse=True)

    # The math ...
    p = (s1+s2+s3)/3
    q = s1 - s3

    # The plotting
    ax.scatter(p, q, label=title, s=50, marker = next(mrk))
    ax.set_title("P-Q diagram ", fontsize=20)
    ax.set_xlim(0,2200)
    ax.set_ylim(0,2200)
    ax.set_ylabel("q")
    ax.set_xlabel("p'")
    ax.legend()
    return ax

# Do the plotting
fig,ax = plt.subplots()
fig.set_size_inches(6,5)

pq(1461,2393,1089, ax, "Point A")
pq(1490,2514,989,  ax, "Point C")
pq(1511,3048,1884, ax, "Point D")
pq(1569,2384,1091, ax, "Point B")
pq(1552,3334,1261, ax, "Point E")

# Format and save.
fig.tight_layout()
fig.savefig('HW1_PQ.svg')
```



Draw I1 – J2

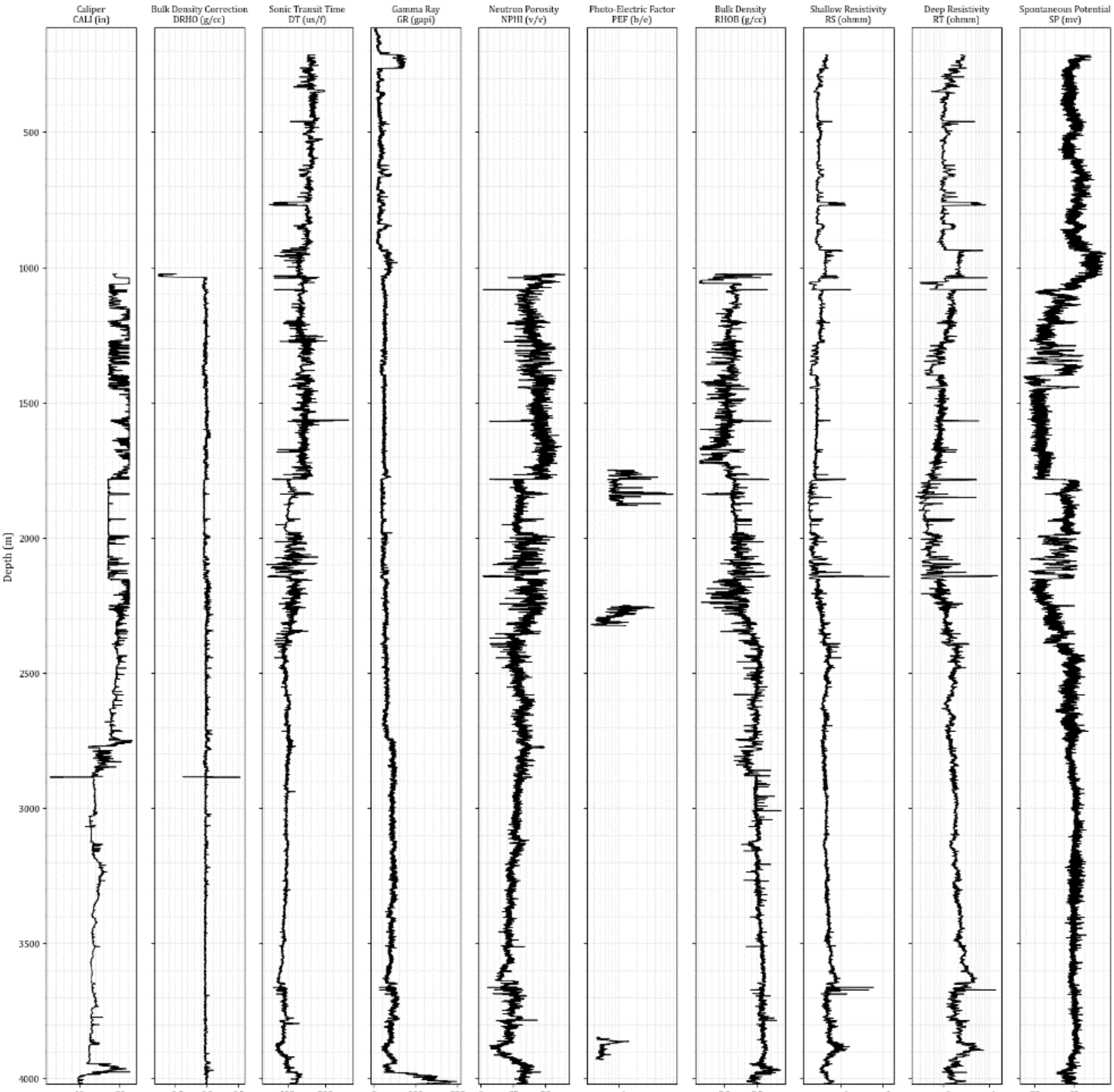
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import itertools
mrk = itertools.cycle(('x', '+', '.', 'o', '*'))
plt.style.use('default')   ## reset!
plt.style.use('paper.mplstyle')

def i1j2(s1, s2, s3, ax, title="") :
    """
    Plot I1-J2 diagram using Pandas DF
    """
    [s1,s2,s3] = sorted([s1,s2,s3], reverse=True)

    i1 = s1+s2+s3
    j2 = ( (s1-s3)**2 + (s1-s2)**2 + (s2-s3)**2 ) / 6

    ax.scatter(i1, j2, label=title, s=50, marker = next(mrk))
    ax.set_title("I1-J2 diagram ", fontsize=20)
    ax.set_xlabel("I1")
    ax.set_ylabel("J2")
    ax.set_xlim(0,7000)
    ax.set_ylim(0,1.4e6)
    ax.legend()
    return ax

# Do the plotting
fig,ax = plt.subplots()
fig.set_size_inches(6,5)

i1j2(1461,2393,1089, ax, "Point A")
i1j2(1490,2514,989, ax, "Point C")
i1j2(1511,3048,1884, ax, "Point D")
i1j2(1569,2384,1091, ax, "Point B")
i1j2(1552,3334,1261, ax, "Point E")

# Format and save.
fig.tight_layout()
fig.savefig('HW1_I1J2.svg')
```



EXERCISE 2

In [1]:

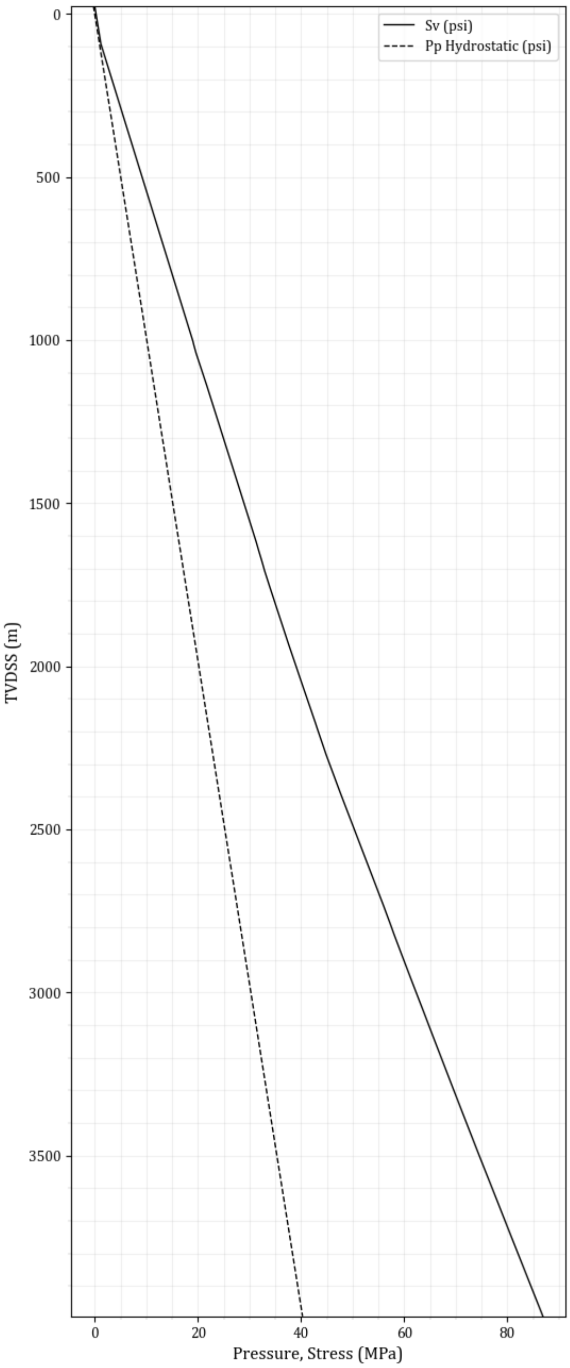
```
import matplotlib.pyplot as plt
import re
plt.style.use('default')   ## reset!
plt.style.use('paper.mplstyle')

import lasio
las = lasio.read("1_14-1_Composite.las")

cfg = {
    "CALI"  : { 'scale' : 'linear' } ,
    "DRHO"  : { 'scale' : 'linear' } ,
    "DT"    : { 'scale' : 'linear' } ,
    "GR"    : { 'scale' : 'linear' } ,
    "NPHI"  : { 'scale' : 'linear' } ,
    "PEF"   : { 'scale' : 'log' } ,
    "RHOB"  : { 'scale' : 'linear' } ,
    "RS"    : { 'scale' : 'log' } ,
    "RT"    : { 'scale' : 'log' } ,
    "RXO"   : { 'scale' : 'log' } ,
    "SP"    : { 'scale' : 'linear' } ,
}

df = las.df().dropna(axis=1, how='all')
n_crv = df.shape[1]

fig, axs = plt.subplots(1, n_crv , sharey=True)
fig.set_size_inches(20,20)
axs[0].set_ylim( df.index.min(), df.index.max() )
axs[0].invert_yaxis()
axs[0].set_ylabel(f"Depth ({las.curves['DEPTH'].unit.lower()})")

i=0
for crv_name in df.columns:
    ax = axs[i]
    descr = las.curves[crv_name].descr.replace('\t'," ")
    descr = re.sub(r"^\d+\s*", "", descr)
    title = f"{descr}\n{crv_name} ({las.curves[crv_name].unit.lower()})"

    ax.plot( df[crv_name], df.index, color='k' )
    ax.set_title( title )
    ax.set_xscale( cfg[crv_name]['scale'] )
    i=i+1

fig.savefig("HW1-LAS_tracks.svg")
```



Caliper
CALI (in)
Bulk Density Correction
DRHO (g/cc)
Sonic Transit Time
DT (us/f)
Gamma Ray
GR (gapi)
Neutron Porosity
NPHI (v/v)
Photo-Electric Factor
PEF (b/e)
Bulk Density
RHOB (g/cc)
Shallow Resistivity
RS (ohmm)
Deep Resistivity
RT (ohmm)
Spontaneous Potential
SP (mv)
Depth (m)

Calculate Sv

```
import lasio
las = lasio.read("1_14-1_Composite.las")

df = las.df().dropna(axis=1, how='all')
# Fill data as indicated in question
df["RHOB"] = df.RHOB.fillna(2)
df["DRHO"] = df.DRHO.fillna(0)
df = df[["RHOB", "DRHO"]].dropna(axis=0).reset_index() # Delete rows with NA

# Add see water rows
df = pd.concat([
        pd.DataFrame( {"DEPTH" : np.linspace(0,104,100), "RHOB": 1.03, "DRHO" : 0 } ),
        df])

df["Bulk_mass_density"] = df["RHOB"] + df["DRHO"]
df["Prev_density"] = df.Bulk_mass_density.shift().fillna(1.03)

#                      g/cc                g[m/s2]      to Pa/m    to MPa
df["SvGrad"] = ( df.Bulk_mass_density/2 + df.Prev_density/2 ) *    9.8      *    1E3    *  1e-6

# Load the deviation model and interpolate into DF
dev_df = pd.read_csv("1_14-1_deviation_mod.dev", sep="\t")
df["TVDSS"] = np.interp(df.DEPTH, dev_df.DEPTH, dev_df.TVDSS )

df['DeltaDepth'] = df.DEPTH.diff().shift(-1)

df['dSv_psi'] = df.DeltaDepth * df.SvGrad
df['Sv'] = df.dSv_psi.shift().cumsum().fillna(0)

# Water gradient and Hydro Pressure
#        g     sea water density (g/m3)     MPa/m
wgrad = 9.8   * 1030               * 1E-6
df["Pp_hyd"] = wgrad * df["TVDSS"]

fig, ax = plt.subplots(1, 1 , sharey=True)
fig.set_size_inches(6,16)
ax.set_ylim( df.TVDSS.min(), df.TVDSS.max() )
ax.invert_yaxis()
ax.set_ylabel(f"TVDSS (m)")
ax.set_xlabel("Pressure, Stress (MPa)")

ax.plot(df.Sv, df.TVDSS, color="k", ls='-', label="Sv (psi)")
ax.plot(df.Pp_hyd, df.TVDSS, color="k", ls='--', label="Pp Hydrostatic (psi)")

ax.legend()

fig.savefig("HW1_Sv_Pp.svg")
```



Sv (psi)
Pp Hydrostatic (psi)
TVDSS (m)
0
500
1000
1500
2000
2500
3000
3500
0
20
40
60
80
Pressure, Stress (MPa)