Homework 3 Solutions

Problem 1

Twelve triaxial tests on cylindrical plugs of Berea sandstone are reported below (Bernabe and Brace, 1990):

a) Plot all data points in a σ_1 VS σ_3 plot and draw respective Mohr Circles (in Matlab, Python or Excel).

σ_3 (the effective minimum principal stress) is calculated using

$$\sigma_3 = P_c - P_p$$

σ_1 (the effective maximum principal stress) is calculated using

$$\sigma_1 = \sigma_3 + \sigma_d$$

Where P_c is the confining pressure, P_p is the pore pressure, and σ_d is the deviatoric stress at failure

Note deviatoric stress is the axial stress that a load cell inside the pressure vessel measures ($S_1 - S_3$). For example, the value would be zero for hydrostatic loading (since $S_1 = S_3$, $S_1 - S_3 = 0$) even though axial stress measured with the load frame is not zero $S_1 \neq 0$.

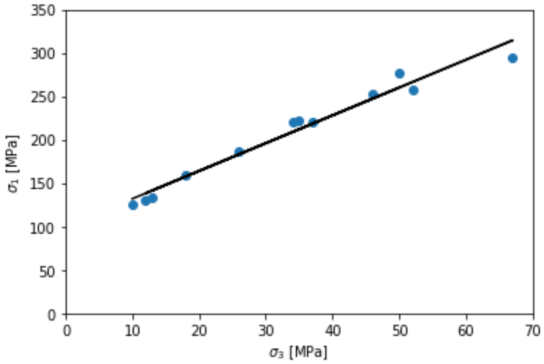
In [1]:
```
import pandas as pd

excel_file = 'HW_3.xlsx'
DataQ1 = pd.read_excel(excel_file, sheet_name=0)
DataQ1.head(12)
```

Out[1]:

	Confining Pressure [MPa]	Pore Pressure [MPa]	Peak Deviatoric Stress [MPa]	Sigma_3 [MPa]	Sigma_1 [MPa]
0	10	0	116	10	126
1	50	0	227	50	277
2	20	8	119	12	131
3	45	8	183	37	220
4	60	8	206	52	258
5	75	8	228	67	295
6	50	37	120	13	133
7	50	32	141	18	159
8	90	64	161	26	187
9	90	55	187	35	222
10	130	96	186	34	220
11	130	84	207	46	253

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt

Pc = DataQ1['Confining Pressure [MPa]']
Pp = DataQ1['Pore Pressure [MPa]']
sigma_d = DataQ1['Peak Deviatoric Stress [MPa]']
sigma_3 = DataQ1['Sigma_3 [MPa]']
sigma_1 = DataQ1['Sigma_1 [MPa]']

# plot data
plt.scatter(sigma_3,sigma_1)
# calculate the simple linear regression fit
coefficients = np.polyfit(sigma_3, sigma_1, 1)
yy = np.poly1d(coefficients)
# plot trendline
plt.plot(sigma_3,yy(sigma_3),"-k")
# print trendline
print ('The simple linear regression fit is:')
print ("y=%.6fx+%.6f"%(coefficients[0],coefficients[1]))
# plot labels
plt.xlabel('$\sigma_3$ [MPa]')
plt.ylabel('$\sigma_1$ [MPa]')
# axis range
plt.xlim([0, 70])
plt.ylim([0, 350])
plt.show()
```

```
The simple linear regression fit is:
y=3.200073x+100.080890
```



Coulomb's failure criterion can be written as

$$\sigma_1 = UCS + q\ \sigma_3$$

Where σ_1 is the effective maximum principal stress at failure, σ_3 is the effective minimum principal stress and q is the friction parameter function of the friction angle (warning: this is not the same q from the $p' - q$ or $\sigma_m - q$ space)

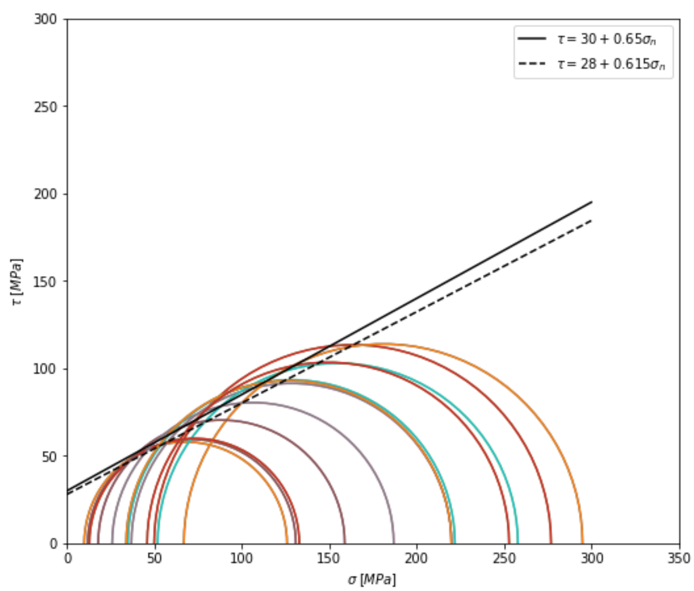
From the above plot, the slope of the linear regression fit through σ_1 VS σ_3 is $q = 3.2$. The intercept is $UCS = 100[MPa]$.

So, Coulomb's failure criterion can be written as

$$\sigma_1 = 100 + 3.2\ \sigma_3$$

In [3]:
```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Function for Mohr's Circle
# Code adapted from John D'Angelo
def plotMohr3D(sig3,sig2,sig1):

    circle1X=[]
    circle1Y=[]
    circle2X=[]
    circle2Y=[]
    circle3X=[]
    circle3Y=[]

    for i in np.linspace(0,np.pi):
        circle1X.append((sig2-sig3)/2*np.cos(i) + (sig3+(sig2-sig3)/2))
        circle2X.append((sig1-sig2)/2*np.cos(i) + (sig2+(sig1-sig2)/2))
        circle3X.append((sig1-sig3)/2*np.cos(i) + (sig3+(sig1-sig3)/2))
        circle1Y.append((sig2-sig3)/2*np.sin(i) )
        circle2Y.append((sig1-sig2)/2*np.sin(i) )
        circle3Y.append((sig1-sig3)/2*np.sin(i) )
    plt.plot(circle2X,circle2Y)
    # plot labels
    plt.xlabel(r'$\sigma \; [MPa]$')
    plt.ylabel(r'$\tau \; [MPa]$')
    # plot layout
    plt.axis('square')
    plt.tight_layout()
    # axis range
    plt.xlim([0, 350])
    plt.ylim([0, 300])

# Plot 2d Mohr Circles
for i in range(0,len(sigma_1)):
    plotMohr3D(0,sigma_3[i],sigma_1[i])
    plotMohr3D(0,sigma_3[i],sigma_1[i])

# Plot shear failure envelope 1
mu = 0.65 #friction coefficient
plt.plot([0,300],[30, 300 * mu],'k-',label=r'$\tau = 30 + 0.65 \sigma_n$')

# Plot shear failure envelope 2
mu = 0.615 #friction coefficient
plt.plot([0,300],[28, 300 * mu],'k--',label=r'$\tau = 28 + 0.615 \sigma_n$')

# Change plot size
fig = plt.gcf()
fig.set_size_inches(7, 7)

# Plot legend
plt.legend()
```

Out[3]:



b) Fit the data to Mohr-Coulomb criterion to compute unconfined compressive strength UCS and the parameter q through a linear regression. Then, calculate the cohesive strength S_0 and internal friction coefficient μ_i

From part a) the Coulomb's failure criterion was

$$\sigma_1 = UCS + q\,\sigma_3$$

$$\sigma_1 = 100 + 3.2\,\sigma_3$$

Where σ_1 is the effective maximum principal stress at failure, σ_3 is the effective minimum principal stress and q is the friction parameter function of the friction angle (warning: this is not the same q from the $p' - q$ or $\sigma_m - q$ space)

Coulomb failure criterion can also be written as:

$$\tau = S_0 + \mu_i \sigma_n$$

Where the maximum shear stress τ will be a function of the rock cohesive strength S_0, the internal friction coefficient μ_i, and the applied normal effective compressive stress σ_n

q (friction parameter function of the friction angle) is related to φ (friction angle) through the equation

$$q = \frac{1 + sin\varphi}{1 - sin\varphi}$$

$$3.2 = \frac{1 + sin\varphi}{1 - sin\varphi}$$

$$\varphi = 31^\circ$$

UCS is a function of S_0 (rock cohesive strength) and φ (friction angle)

$$UCS = 2S_0\sqrt{\frac{1 + \sin\varphi}{1 - \sin\varphi}}$$

$$100 = 2S_0\sqrt{3.2}$$

$$S_0 = 28[MPa]$$

q (friction parameter function of the friction angle) is related to μ_i (internal friction coefficient) through the equation

$$q = (\sqrt{\mu_i^2 + 1} + \mu_i)^2$$

$$\mu_i = \frac{q - 1}{2\sqrt{q}} = \frac{3.2 - 1}{2\sqrt{3.2}} = 0.615$$

So, in summary

$$\tau = S_0 + \mu_i \sigma_n$$

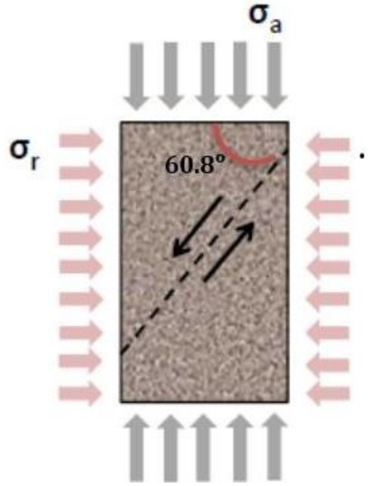
$$\tau = 28 + 0.615\sigma_n$$

From the Mohr Circles plot, a μ_i (internal friction coefficient) of 0.65 and S_0 (rock cohesive strength) of 30 seems to give a better fit.

c) Based on this information, compute the failure angle of the shear fracture you would expect to see in this sample after failure. Draw a sketch indicating the orientation with respect to the axial and radial stress.

The failure angle is

$$\alpha = 45^{\circ} + \frac{\varphi}{2} = 60.8^{\circ}$$



d) Did pore pressure significantly change the effective stress failure criterion?

Rock strength and shear failure depends on effective stress, so pore pressure doesn't change effective stress failure line in the effective stress space.

Problem 2

The file "Triaxial-1500psi-raw.xlsx" in the 'Homework' folder contains data from a triaxial test performed on a sandstone in dry conditions, $P_p = 0\ psi$. $P_c = 1500\ psi$ is the confining pressure, SigD is the deviatoric stress ($S_1 - S_3$), Ex is the axial strain, and Ey is the radial strain.

In [4]:
```python
import pandas as pd
excel_file = 'HW_3.xlsx'
DataQ2 = pd.read_excel(excel_file, sheet_name=1)
DataQ2.head(2427)
```

Out[4]:

	Time [s]	SigD [psi]	Axial Strain, Ex	Radial Strain, Ey	Volumetric Strain, Evol
0	924.3000	12.313760	0.001796	0.002105	0.006005
1	925.3000	13.134120	0.001795	0.002106	0.006006
2	926.3000	12.071360	0.001796	0.002106	0.006008
3	927.3000	11.906290	0.001796	0.002107	0.006010
4	928.3000	11.021060	0.001797	0.002108	0.006013
5	929.3000	9.342437	0.001797	0.002108	0.006014
6	930.3000	10.996250	0.001796	0.002109	0.006014
7	931.3000	13.646980	0.001796	0.002110	0.006016
8	932.3000	10.222380	0.001797	0.002111	0.006019
9	933.3000	11.427650	0.001797	0.002112	0.006020
10	934.3000	12.700250	0.001797	0.002113	0.006023
11	935.3000	13.318190	0.001798	0.002113	0.006025
12	936.3000	12.372090	0.001797	0.002114	0.006025
13	937.3000	11.999220	0.001798	0.002115	0.006028
14	938.3000	15.766300	0.001797	0.002115	0.006027
15	939.3000	12.364130	0.001798	0.002116	0.006030
16	940.3000	11.488130	0.001799	0.002117	0.006032
17	941.3000	12.023170	0.001797	0.002118	0.006033
18	942.3000	12.016570	0.001799	0.002118	0.006035
19	943.3000	7.637846	0.001798	0.002119	0.006036
20	944.3000	12.537750	0.001799	0.002119	0.006037
21	945.3000	11.179910	0.001798	0.002119	0.006036
22	946.3000	12.430140	0.001798	0.002121	0.006039
23	946.7001	12.151980	0.001799	0.002121	0.006041
24	947.7001	14.420810	0.001801	0.002124	0.006048
25	948.7001	14.158290	0.001804	0.002124	0.006052
26	949.7001	12.227550	0.001806	0.002124	0.006054
27	950.7001	15.016960	0.001807	0.002124	0.006056
28	951.7001	12.295560	0.001807	0.002125	0.006057
29	952.7001	20.204210	0.001809	0.002126	0.006061
...	...	...	...	...	...
2397	3318.4300	16578.270000	0.010800	-0.005069	0.000661
2398	3319.4300	16568.850000	0.010804	-0.005081	0.000642
2399	3320.4300	16551.110000	0.010808	-0.005092	0.000624
2400	3321.4300	16525.290000	0.010812	-0.005102	0.000607
2401	3322.4300	16509.330000	0.010821	-0.005114	0.000593
2402	3323.4300	16502.900000	0.010821	-0.005128	0.000565
2403	3324.4300	16483.040000	0.010824	-0.005140	0.000544
2404	3325.4300	16466.790000	0.010828	-0.005154	0.000520
2405	3326.4300	16445.140000	0.010833	-0.005166	0.000500
2406	3327.4300	16425.490000	0.010834	-0.005179	0.000476
2407	3328.4300	16401.880000	0.010839	-0.005191	0.000457
2408	3329.4300	16384.780000	0.010845	-0.005205	0.000435
2409	3330.4300	16362.550000	0.010845	-0.005217	0.000411

2410	3331.4300	16344.970000	0.010852	-0.005233	0.000386
2411	3332.4300	16319.620000	0.010857	-0.005246	0.000364
2412	3333.4300	16313.600000	0.010859	-0.005264	0.000331
2413	3334.4300	16278.440000	0.010861	-0.005277	0.000308
2414	3335.4300	16246.080000	0.010867	-0.005293	0.000282
2415	3336.4300	16223.740000	0.010874	-0.005309	0.000256
2416	3337.4300	16191.880000	0.010878	-0.005326	0.000226
2417	3338.4300	16154.570000	0.010880	-0.005343	0.000194
2418	3339.4300	16127.780000	0.010885	-0.005363	0.000159
2419	3340.4300	16083.020000	0.010888	-0.005383	0.000122
2420	3341.4300	16036.500000	0.010894	-0.005404	0.000086
2421	3342.4300	16002.070000	0.010896	-0.005429	0.000038
2422	3343.4300	15935.310000	0.010899	-0.005455	-0.000011
2423	3344.4300	15883.010000	0.010901	-0.005488	-0.000076
2424	3345.4300	15816.840000	0.010906	-0.005522	-0.000139
2425	3346.4300	15714.220000	0.010909	-0.005560	-0.000211
2426	3347.4300	15610.060000	0.010917	-0.005624	-0.000330

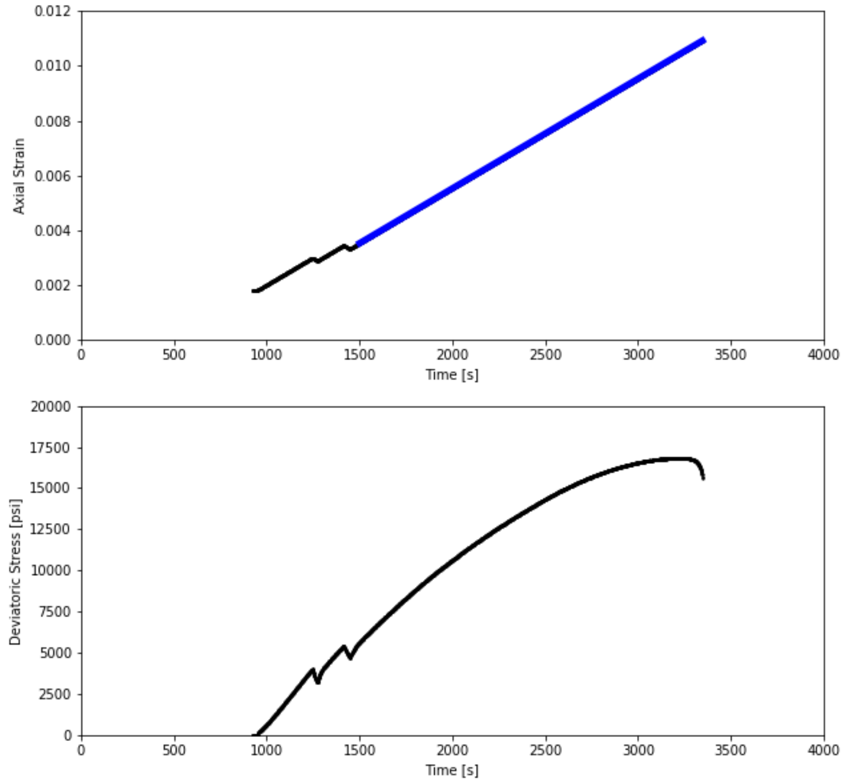
2427 rows × 5 columns

a) Plot deviatoric stress and strains as a function of time (two plots). Mechanical experiments are usually performed at constant strain rate or constant stress rate. Which case is this? What is the rate?

The axial strain vs time plot shows that the experiment is constant strain rate loading. The loading strain rate is equal to $4 \times 10^{-6}(\frac{1}{s})$

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt

Time = DataQ2['Time [s]']
SigD = DataQ2['SigD [psi]']
Eaxial = DataQ2['Axial Strain, Ex']
Eradial = DataQ2['Radial Strain, Ey']
Evol = DataQ2['Volumetric Strain, Evol']

# Axial strain vs time
plt.subplot(2, 1, 1)
plt.scatter(Time,Eaxial,color="k",s=2)
# plot labels
plt.xlabel('Time [s]')
plt.ylabel('Axial Strain')
# axis range
plt.xlim([0, 4000])
plt.ylim([0, 0.012])

# Segment of the data set for calculating loading strain rate
Time1 = Time.iloc[580:2427]
Eaxial1 = Eaxial.iloc[580:2427]
# Calculate the simple linear regression fit to find loading strain rate
coefficients = np.polyfit(Time1, Eaxial1, 1)
yy = np.poly1d(coefficients)
# plot trendline
plt.plot(Time1,yy(Time1),"-b",linewidth=5)
# print trendline
print ("Linear regression fit through axial strain vs time data gives")
print ("y=%.6fx+%.6f"%(coefficients[0],coefficients[1]))
print ("The slope is the loading rate")

# Deviatoric stress vs time
plt.subplot(2, 1, 2)
plt.scatter(Time,SigD,color="k",s=2)
# plot labels
plt.xlabel('Time [s]')
plt.ylabel('Deviatoric Stress [psi]')
# axis range
plt.xlim([0, 4000])
plt.ylim([0, 20000])

# Change plot size
fig = plt.gcf()
fig.set_size_inches(10, 10)
```

```
Linear regression fit through axial strain vs time data gives
y=0.000004x+-0.002474
The slope is the loading rate
```



b) Plot deviatoric stress as a function of axial strain. Compute loading Young modulus at 25% of peak stress and the unloading Young moduli for the two unloading cycles. Comment on the difference.

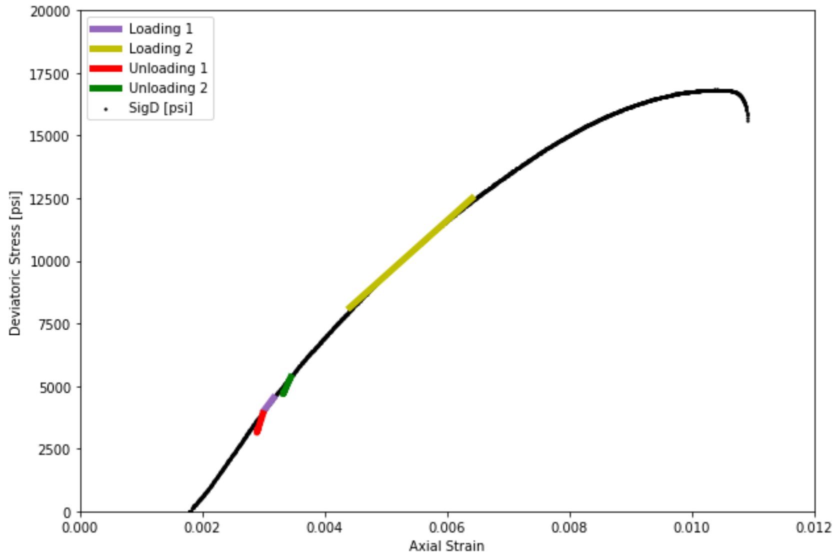
Loading Young's modulus at 25% of peak stress is 3270 [Mpsi]
Loading Young's modulus at 50% of max strain is 2180 [Mpsi]
Unloading Young's modulus for the first cyle is 7560 [Mpsi]
Unloading Young's modulus for the second cyle is 5340 [Mpsi]

In [6]:

```
import numpy as np
import matplotlib.pyplot as plt

Time = DataQ2['Time [s]']
SigD = DataQ2['SigD [psi]']
Eaxial = DataQ2['Axial Strain, Ex']
Eradial = DataQ2['Radial Strain, Ey']
Evol = DataQ2['Volumetric Strain, Evol']

# Deviatoric stress vs axial strain
plt.scatter(Eaxial,SigD,color="k",s=2)
# plot labels
plt.xlabel('Axial Strain')
plt.ylabel('Deviatoric Stress [psi]')
# axis range
plt.xlim([0, 0.012])
plt.ylim([0, 20000])

# Segment of the data set for calculating loading Young's modulus at 25% of peak stress (~16800 psi)
Eaxial_load1 = Eaxial.iloc[380:420]
SigD_load1 = SigD.iloc[380:420]
# Calculate the simple linear regression fit to find loading Young's modulus at 25% of peak stress
coefficients = np.polyfit(Eaxial_load1, SigD_load1, 1)
yy = np.poly1d(coefficients)
# plot trendline
plt.plot(Eaxial_load1,yy(Eaxial_load1),"tab:purple",label='Loading 1',linewidth=5)
# print trendline
print ("Linear regression fit through the loading phase at 25% of peak stress gives")
print ("y=%.6fx+%.6f"%(coefficients[0],coefficients[1]))
print ("The slope is the Young's modulus")

# Segment of the data set for calculating loading Young's modulus at 50% of max strain
Eaxial_load2 = Eaxial.iloc[800:1300]
SigD_load2 = SigD.iloc[800:1300]
# Calculate the simple linear regression fit to find loading Young's modulus at 50% of max strain
coefficients = np.polyfit(Eaxial_load2, SigD_load2, 1)
yy = np.poly1d(coefficients)
# plot trendline
plt.plot(Eaxial_load2,yy(Eaxial_load2),"-y",label='Loading 2',linewidth=5)
# print trendline
print ("Linear regression fit through the loading phase at 50% of max strain gives")
print ("y=%.6fx+%.6f"%(coefficients[0],coefficients[1]))
# Segment of the data set for calculating second unloading Young's modulus
Eaxial_unload2 = Eaxial.iloc[495:529]
SigD_unload2 = SigD.iloc[495:529]
# Calculate the simple linear regression fit to find second unloading Young's modulus
coefficients = np.polyfit(Eaxial_unload2, SigD_unload2, 1)
yy = np.poly1d(coefficients)
# plot trendline
plt.plot(Eaxial_unload2,yy(Eaxial_unload2),"-g",label='Unloading 2',linewidth=5)
# print trendline
print ("Linear regression fit through the second unloading phase gives")
print ("y=%.6fx+%.6f"%(coefficients[0],coefficients[1]))

# Change plot size
fig = plt.gcf()
fig.set_size_inches(10, 7)
# Legend
plt.legend()
```

```
Linear regression fit through the loading phase at 25% of peak stress gives
y=3274902.890458x+-5829.303546
The slope is the Young's modulus
Linear regression fit through the loading phase at 50% of max strain gives
y=2183282.780543x+-1492.591370
Linear regression fit through the first unloading phase gives
y=7560143.023729x+-18713.050167
Linear regression fit through the second unloading phase gives
y=5338099.301622x+-13037.001663
```

Out[6]:



c) Plot radial strain VS axial strain and compute loading Poisson ratio.

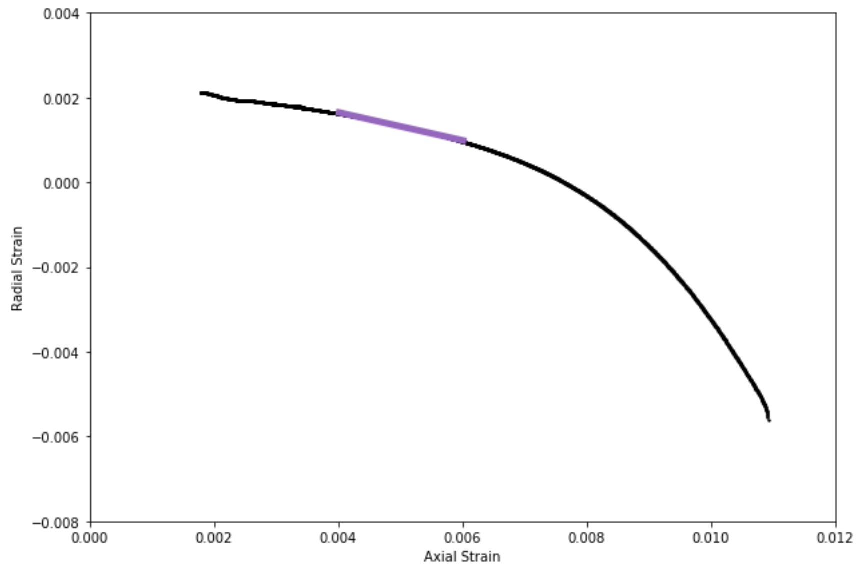
Poisson's ratio:

$$\frac{\epsilon_r}{\epsilon_a} = -\nu = -0.371$$

In [7]:

```
import numpy as np
import matplotlib.pyplot as plt

Time = DataQ2['Time [s]']
SigD = DataQ2['SigD [psi]']
Eaxial = DataQ2['Axial Strain, Ex']
Eradial = DataQ2['Radial Strain, Ey']
Evol = DataQ2['Volumetric Strain, Evol']

# Radial strain vs axial strain
plt.scatter(Eaxial,Eradial,color="k",s=2)
# plot labels
plt.xlabel('Axial Strain')
plt.ylabel('Radial Strain')
# axis range
plt.xlim([0, 0.012])
plt.ylim([-0.008, 0.004])

# Segment of the data set for calculating Poisson's ratio
Eaxial_poisson = Eaxial.iloc[700:1200]
Eradial_poisson = Eradial.iloc[700:1200]
# Calculate the simple linear regression fit to find Poisson's ratio
coefficients = np.polyfit(Eaxial_poisson, Eradial_poisson, 1)
yy = np.poly1d(coefficients)
# plot trendline
plt.plot(Eaxial_poisson,yy(Eaxial_poisson),"tab:purple",linewidth=5)
# print trendline
print ("Linear regression fit through the portion of the data used to calculate Poisson's ratio gives")
print ("y=%.6fx+%.6f"%(coefficients[0],coefficients[1]))
print ("The slope is the Poisson's ratio")

# Change plot size
fig = plt.gcf()
fig.set_size_inches(10, 7)
```

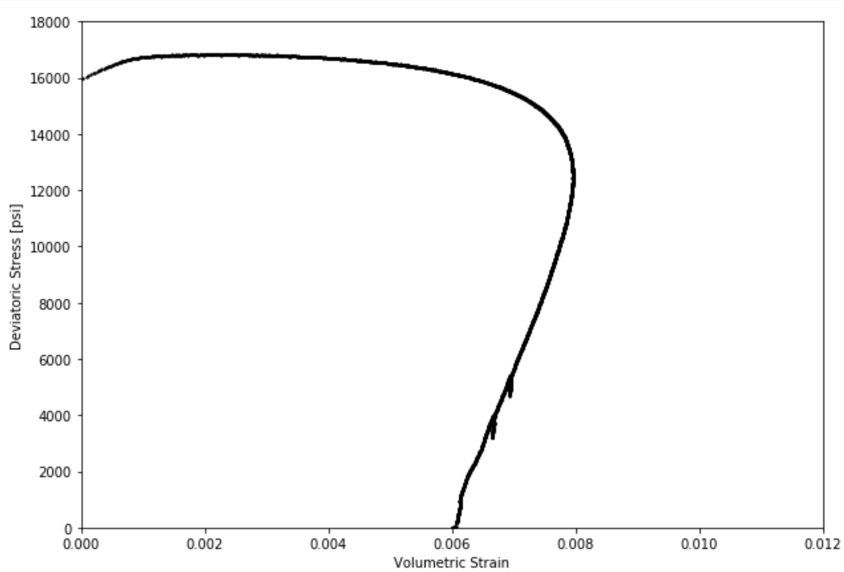
```
Linear regression fit through the portion of the data used to calculate Poisson's ratio gives
y=-0.329833x+0.002966
The slope is the Poisson's ratio
```



d) Plot deviatoric stress VS volumetric strain. Does the sample contract, dilate, or both? Explain.

Sample first contracts and then dilates.

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt

Time = DataQ2['Time [s]']
SigD = DataQ2['SigD [psi]']
Eaxial = DataQ2['Axial Strain, Ex']
Eradial = DataQ2['Radial Strain, Ey']
Evol = DataQ2['Volumetric Strain, Evol']

# Radial strain vs axial strain
plt.scatter(Evol,SigD,color="k",s=2)
# plot labels
plt.xlabel('Volumetric Strain')
plt.ylabel('Deviatoric Stress [psi]')
# axis range
plt.xlim([0, 0.012])
plt.ylim([0, 18000])

# Change plot size
fig = plt.gcf()
fig.set_size_inches(10, 7)
```



e) If q=5.3, what is the UCS of this rock?

Confining pressure or the effective minimum principal stress (σ_3) is given to be 1500 [psi]

From part b), we plotted deviatoric stress (σ_d) as a function of axial strain. The peak stress was ~16800 [psi]

Recall σ_1 (the effective maximum principal stress) is calculated using

$$\sigma_1 = \sigma_3 + \sigma_d$$

$$\sigma_1 = 1500 + 16800 = 18300[psi]$$

Recall Coulomb's failure criterion can be written as

$$\sigma_1 = UCS + q\ \sigma_3$$

$$18300 = UCS + 5.3 \times 1500$$

$$UCS = 10350[psi]$$