

Homework 4 Solutions

Problem 1

Let's compare reservoirs with normal and reverse faulting regimes (onshore). For both regimes, plot the limits of S_1 and S_3 versus depth from 0 to 5km assuming hydrostatic pore pressure and a state of stress limited by frictional strength of faults. Assume sliding friction coefficient is $\mu = 0.6$. For both regimes, how does changing μ from 0.6 to 1.0 change the maximum possible differential stress $S_1 - S_3$ at a fixed depth of $z = 5\text{km}$? Explain what that friction coefficient change physically means.

Normal faulting occurs in tectonically passive or extensional environments, such that $S_V > S_{Hmax} > S_{Hmin}$, where $S_1 = S_V$, $S_2 = S_{Hmax}$, $S_3 = S_{Hmin}$. Hydraulic fractures in this environment are vertical.

Reverse faulting occurs in "strong" tectonically compressive environments, such that $S_{Hmax} > S_{Hmin} > S_V$, where $S_1 = S_{Hmax}$, $S_2 = S_{Hmin}$, $S_3 = S_V$. Hydraulic fractures in this environment are horizontal.

We assume the pore pressure gradient is 10 MPa/km and the overburden gradient is 23 MPa/km.

q (friction parameter function of the friction angle) is related to μ (friction coefficient) through the equation:

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

From the below graphs, the higher the friction coefficient (μ), the larger $\sigma_1 - \sigma_3$ is. Physically this means a stronger fault.

In general, σ_3 (minimum effective stress) is related to σ_1 (maximum effective stress) through the equation:

$$\sigma_1 = \sigma_o + q\sigma_3$$

Because there is a lack of cohesive strength in the Earth's crust at the large scale we have:

$$\sigma_1 = q\sigma_3$$

$$\sigma_3 = \frac{\sigma_1}{q}$$

Where $\sigma_1 = S_1 - P_p$ and $\sigma_3 = S_3 - P_p$

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

# Pressure Gradients Assumed
PpGradient = 10 # dPp/dz = 10 MPa/km
SvGradient = 23 # dSv/dz = 23 MPa/km

## Case when friction coefficient is 0.6
mu1 = 0.6 # define mu, friction coefficient
q1 = ((mu1**2+1)**0.5+mu1)**2 # calculates q, friction parameter function of the friction angle
print ("When the friction coefficient u=0.6, q is calculated to be",round(q1,2))

# Normal Faulting
plt.subplot(3, 2, 1)
depth = np.linspace(0, 5, 5) # depth in km
plt.plot(depth*PpGradient, depth, '-b', label='$P_p$') # Pp (Data)
plt.plot(depth*SvGradient, depth, '-r', label='$S_1=S_v$') # S1 = Sv (Data)
plt.plot((depth*SvGradient-depth*PpGradient)/q1+depth*PpGradient, depth, '--g', label='$S_3=S_{hmin}$') # S3 = Shmin (Calculated)
# Plot Labels
plt.xlabel('Pressure [MPa]')
plt.ylabel('Depth [km]')
plt.title("Normal Faulting ($\mu=0.6$)")
plt.legend()
# Axis range
plt.xlim([0, 140])
plt.ylim([5, 0])

# Reverse Faulting
plt.subplot(3, 2, 2)
depth = np.linspace(0, 5, 5) # depth in km
plt.plot(depth*PpGradient, depth, '-b', label='$P_p$') # Pp (Data)
plt.plot((depth*SvGradient-depth*PpGradient)*q1+depth*PpGradient, depth, '--g', label='$S_1=S_{Hmax}$') # S1 = SHmax (Calculated)
plt.plot(depth*SvGradient, depth, '-r', label='$S_3=S_v$') # S3 = Sv (Data)
# Plot Labels
plt.xlabel('Pressure [MPa]')
plt.ylabel('Depth [km]')
plt.title("Reverse Faulting ($\mu=0.6$)")
plt.legend()
# Axis range
plt.xlim([0, 500])
plt.ylim([5, 0])
```

```

## Case when friction coefficient is 1
mu2 = 1 # define mu, friction coefficient
q2=((mu2**2+1)**0.5+mu2)**2 # calculates q, friction parameter function of the friction angle
print ("When the friction coefficient u=1, q is calculated to be",round(q2,2))

# Normal Faulting
plt.subplot(3, 2, 3)
depth = np.linspace(0, 5, 5) # depth in km
plt.plot(depth*PpGradient, depth, '-b', label='$P_p$',) # Pp (Data)
plt.plot(depth*SvGradient, depth, '-r', label='$S_1=S_v$',) # S1 = Sv (Data)
plt.plot((depth*SvGradient-depth*PpGradient)/q1+depth*PpGradient, depth, '--g', label='$S_3=S_{hmin}$') # S3 = Shmin (Calculated)
# Plot Labels
plt.xlabel('Pressure [MPa]')
plt.ylabel('Depth [km]')
plt.title("Normal Faulting ($\mu=1$)")
plt.legend()
# Axis range
plt.xlim([0, 140])
plt.ylim([5, 0])

# Reverse Faulting
plt.subplot(3, 2, 4)
depth = np.linspace(0, 5, 5) # depth in km
plt.plot(depth*PpGradient, depth, '-b', label='$P_p$',) # Pp (Data)
plt.plot((depth*SvGradient-depth*PpGradient)*q1+depth*PpGradient, depth, '--g', label='$S_1=S_{Hmax}$') # S1 = SHmax (Calculated)
plt.plot(depth*SvGradient, depth, '-r', label='$S_3=S_v$',) # S3 = Sv (Data)
# Plot Labels
plt.xlabel('Pressure [MPa]')
plt.ylabel('Depth [km]')
plt.title("Reverse Faulting ($\mu=1$)")
plt.legend()
# Axis range
plt.xlim([0, 500])
plt.ylim([5, 0])

## Comparing S1-S3

# Normal Faulting
plt.subplot(3, 2, 5)
depth = np.linspace(0, 5, 5) # depth in km
S1 = depth*SvGradient
S3_1 = (depth*SvGradient-depth*PpGradient)/q1+depth*PpGradient
S3_2 = (depth*SvGradient-depth*PpGradient)*q2+depth*PpGradient
plt.plot(S1-S3_1, depth, label='$\mu=0.6$',) # S1-S3 (friction coefficient is 0.6)
plt.plot(S1-S3_2, depth, label='$\mu=1$',) # S1-S3 (friction coefficient is 1)
# Plot Labels
plt.xlabel('Pressure [MPa]')
plt.ylabel('Depth [km]')
plt.title("Normal Faulting ($S_1-S_3$)")
plt.legend()
# Axis range
plt.xlim([0, 140])
plt.ylim([5, 0])

# Reverse Faulting
plt.subplot(3, 2, 6)
depth = np.linspace(0, 5, 5) # depth in km
S1_1 = (depth*SvGradient-depth*PpGradient)*q1+depth*PpGradient
S1_2 = (depth*SvGradient-depth*PpGradient)*q2+depth*PpGradient
S3 = depth*SvGradient
plt.plot(S1_1-S3, depth, label='$\mu=0.6$',) # S1-S3 (friction coefficient is 0.6)
plt.plot(S1_2-S3, depth, label='$\mu=1$',) # S1-S3 (friction coefficient is 1)
# Plot Labels
plt.xlabel('Pressure [MPa]')
plt.ylabel('Depth [km]')
plt.title("Reverse Faulting ($S_1-S_3$)")
plt.legend()
# Axis range
plt.xlim([0, 500])
plt.ylim([5, 0])

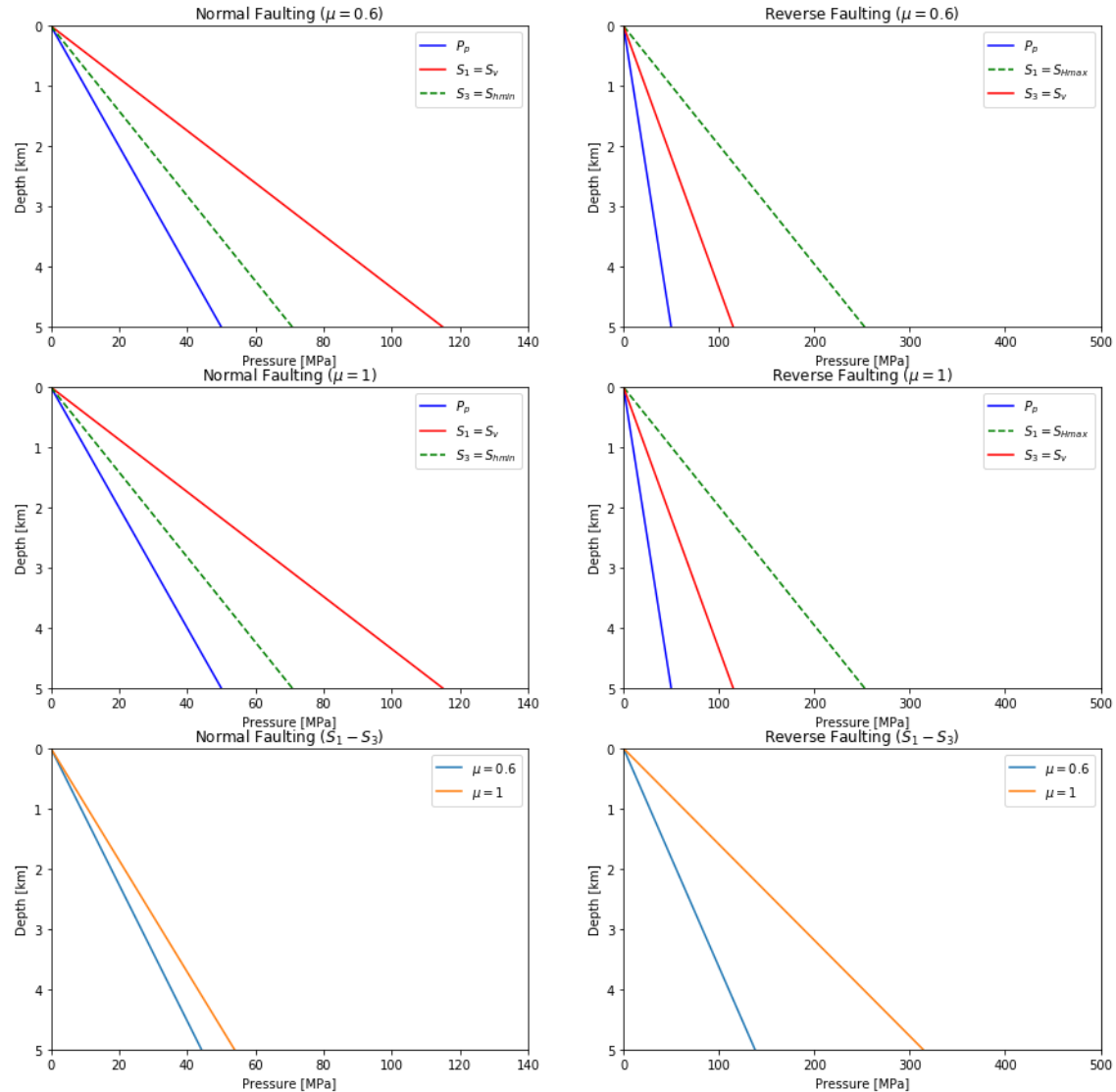
# Change plot size
fig = plt.gcf()
fig.set_size_inches(15, 15)
print("In the below graphs, dashed line signifies stresses that were calculated while solid line indicates stresses that were")
print("directly calculated from assumed pressure gradients")

```

When the friction coefficient $\mu=0.6$, q is calculated to be 3.12

When the friction coefficient $\mu=1$, q is calculated to be 5.83

In the below graphs, dashed line signifies stresses that were calculated while solid line indicates stresses that were directly calculated from assumed pressure gradients



```
In [3]: import pandas as pd

excel_file = 'HW4.xlsx'
DataQ2 = pd.read_excel(excel_file, sheet_name=0)
DataQ2.head()
```

Out[3]:

	Normal Faulting	Depth [km]	Pp [MPa]	Sv [MPa]	σ_v [MPa]	Shmin [MPa]	σ_{Hmin} [MPa]	S1-S3 [MPa]	Reverse Faulting	SHmax [MPa]	σ_{Hmax} [MPa]	S1-S3 [MPa].1	~.1	~.2	~.3	~.4
0	Sv > SHmax > Shmin	0	0	0	0	0.00000	0.00000	0.00000	SHmax > Shmin > Sv	0.00000	0.00000	0.00000	-	Pore Pressure Gradient	dPp/dz	10 [Mpa/km]
1	-	1	10	23	13	14.16743	4.16743	8.83257	-	50.55257	40.55257	27.55257	-	Overburden Gradient	dSv/dz	23 [Mpa/km]
2	-	2	20	46	26	28.33486	8.33486	17.66514	-	101.10514	81.10514	55.10514	-	Friction Coefficient	μ	0.6 [-]
3	-	3	30	69	39	42.50229	12.50229	26.49771	-	151.65771	121.65771	82.65771	-	Friction Parameter Function	q	3.11943 [-]
4	-	4	40	92	52	56.66972	16.66972	35.33028	-	202.21028	162.21028	110.21028	-	-	-	-

Problem 2

A given site offshore (sea floor at 500ft) is subjected to a Normal Faulting stress regime. Over-pressure is first detected at 1500 ft (TVD) and at 2000ft overpressure is $\lambda_p = 0.78$. Calculate the total minimum horizontal stress S_{hmin} at this depth assuming frictional equilibrium of faults and friction angle 30° . Draw the corresponding Mohr circle with a solid line and draw an additional Mohr circle with a dashed line assuming hydrostatic pressure. (Assume typical water and lithostatic gradients)

We assume the pore pressure gradient is 0.44 psi/ft and the overburden gradient is 1 psi/ft.

The overburden at 2000 ft is the summation of the overlying hydrostatic head of water and the remaining pressure head to that depth:

$$S_v = 500 \times 0.44 + 1500 \times 1 = 1720psi$$

The actual pore pressure at any depth beyond 1500 ft TVD where overpressure was first detected can be calculated since the overpressure parameter λ_p of 0.78 is given. The actual pore pressure at 2000 ft would be:

$$P_p = S_v \times \lambda_p = 1720 \times 0.78 = 1342psi$$

So the effective vertical stress is:

$$\sigma_v = S_v - P_p = 1720 - 1342 = 378psi$$

In general, σ_3 (minimum effective stress) is related to σ_1 (maximum effective stress) through the equation:

$$\sigma_1 = S_o + q\sigma_3$$

Because there is a lack of cohesive strength in the Earth's crust at the large scale we have:

$$\sigma_1 = q\sigma_3$$

$$\sigma_3 = \frac{\sigma_1}{q}$$

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

$$q = \frac{1 + \sin\varphi}{1 - \sin\varphi} = \frac{1 + \sin(30^\circ)}{1 - \sin(30^\circ)} = 3$$

For a normal faulting regime, $\sigma_v > \sigma_{Hmax} > \sigma_{hmin}$, where $\sigma_1 = \sigma_v$, $\sigma_2 = \sigma_{Hmax}$, $\sigma_3 = \sigma_{hmin}$, we can calculate σ_{hmin} to be:

$$\sigma_{hmin} = \frac{\sigma_v}{q} = \frac{378}{3} = 126psi$$

The total minimum horizontal stress is:

$$S_{hmin} = \sigma_{hmin} + P_p = 126 + 1342 = 1468psi$$

This procedure can be repeated for hydrostatic pressure (overpressure parameter λ_p of 0.44) to give:

$$\sigma_v = S_v - P_p = 1720 - 880 = 840psi$$

$$\sigma_{hmin} = \frac{\sigma_v}{q} = \frac{840}{3} = 280psi$$

```
In [4]: import pandas as pd

excel_file = 'HW4.xlsx'
DataQ2 = pd.read_excel(excel_file, sheet_name=1)
DataQ2.head()
```

Out[4]:

	Location	Depth [ft]	Actual Pp [psi]	Normal Pp [psi]	Sv [psi]	Actual ov [psi]	Normal ov [psi]	Actual Shmin [psi]	Normal Shmin [psi]	Actual ohmin [psi]	Normal ohmin [psi]	-	Pore Pressure Gradient	dPp/dz	0.44 [psi/ft]	
0	Surface	0	0.0	0	0	0.0	0	0.000000	0.000000	0.000000	0.000000	-	Overburden Gradient	dSv/dz	1.00 [psi/ft]	
1	Seafloor	500	220.0	220	220	0.0	0	220.000000	220.000000	0.000000	0.000000	-	Overpressure Parameter	λ_p	0.78	[-]
2	Rock	1500	660.0	660	1220	560.0	560	846.666667	846.666667	186.666667	186.666667	-	Friction Angle	φ	30.00	[°]
3	TVD	2000	1341.6	880	1720	378.4	840	1467.733333	1160.000000	126.133333	280.000000	-	Friction Parameter Function	q	3.00	[-]

```

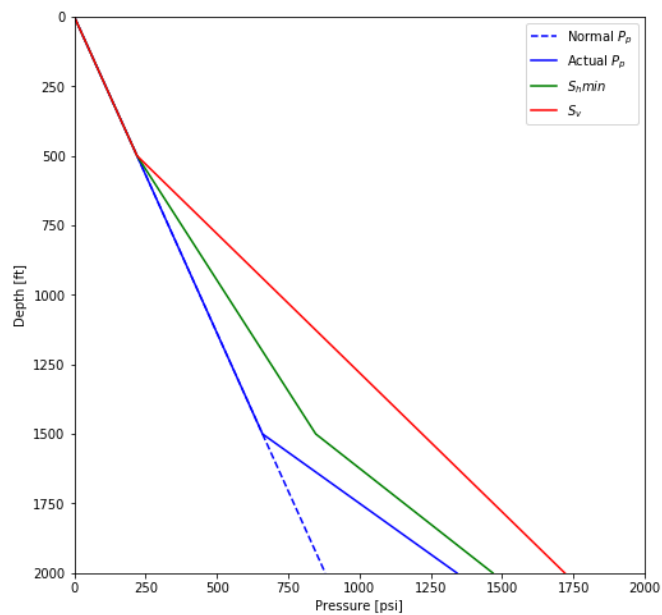
In [5]: import numpy as np
import matplotlib.pyplot as plt

Depth = DataQ2['Depth [ft]']
Pp_normal = DataQ2['Normal Pp [psi]']
Pp_actual = DataQ2['Actual Pp [psi]']
Shmin_normal = DataQ2['Normal Shmin [psi]']
Shmin_actual = DataQ2['Actual Shmin [psi]']
Sv = DataQ2['Sv [psi]']

# Change plot size
fig = plt.gcf()
fig.set_size_inches(8, 8)

# Plot data
plt.plot(Pp_normal, Depth, '--b', label='Normal $P_p$')
plt.plot(Pp_actual, Depth, '-b', label='Actual $P_p$')
plt.plot(Shmin_actual, Depth, '-g', label='$S_{hmin}$')
plt.plot(Sv, Depth, '-r', label='$S_v$')
# Plot labels
plt.xlabel('Pressure [psi]')
plt.ylabel('Depth [ft]')
plt.legend()
# Axis range
plt.xlim([0, 2000])
plt.ylim([2000, 0])
plt.show()

```



```

In [6]: import numpy as np
import matplotlib.pyplot as plt

# Function for Mohr's Circle
def plotMohr3D1(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))
        circle1Y.append((sig1-sig2)/2*np.cos(i) + (sig2+(sig1-sig2)/2))
        circle2X.append((sig1-sig3)/2*np.cos(i) + (sig3+(sig1-sig3)/2))
        circle2Y.append((sig2-sig3)/2*np.sin(i) )
        circle3X.append((sig1-sig2)/2*np.sin(i) )
        circle3Y.append((sig1-sig3)/2*np.sin(i) )
    plt.plot(circle2X,circle2Y,'-',label=r'$\lambda_p=0.78$')
    # 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, 1000])
    plt.ylim([0, 400])

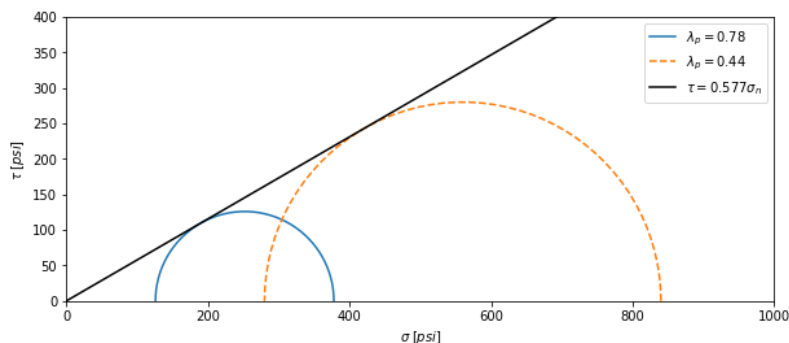
def plotMohr3D2(sig3,sig2,sig1):
    circle1X=[]
    circle1Y=[]
    circle2X=[]
    circle2Y=[]
    circle3X=[]
    circle3Y=[]
    for i in np.linspace(0,np.pi):
        circle1X.append((sig2-sig3)/2*np.cos(i) + (sig3+(sig2-sig3)/2))
        circle2X.append((sig1-sig2)/2*np.cos(i) + (sig2+(sig1-sig2)/2))
        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,'--',label=r'$\lambda_p=0.44$')
    # Plot Labels
    plt.xlabel(r'$\sigma \ ; \ [psi] $')
    plt.ylabel(r'$\tau \ ; \ [psi] $')
    # Plot Layout
    plt.axis('square')
    plt.tight_layout()
    # Axis range
    plt.xlim([0, 1000])
    plt.ylim([0, 400])

# Plot 2d Mohr Circles
plotMohr3D1(0,126,378) # overpressure parameter of 0.78
plotMohr3D2(0,280,840) # overpressure parameter of 0.44
# Plot shear failure envelope
phi = (30 / 180) * np.pi # calculate friction angle
q = (1+np.sin(phi))/(1-np.sin(phi)) # calculate friction parameter function of the friction angle
mu = (q-1)/(2*(q)**0.5) # calculate friction coefficient
plt.plot([0,1000],[0, 1000 * mu],'k-',label=r'$\tau = 0.577 \sigma_n$')

# Change plot size
fig = plt.gcf()
fig.set_size_inches(8, 6)
# Plot Legend
plt.legend()

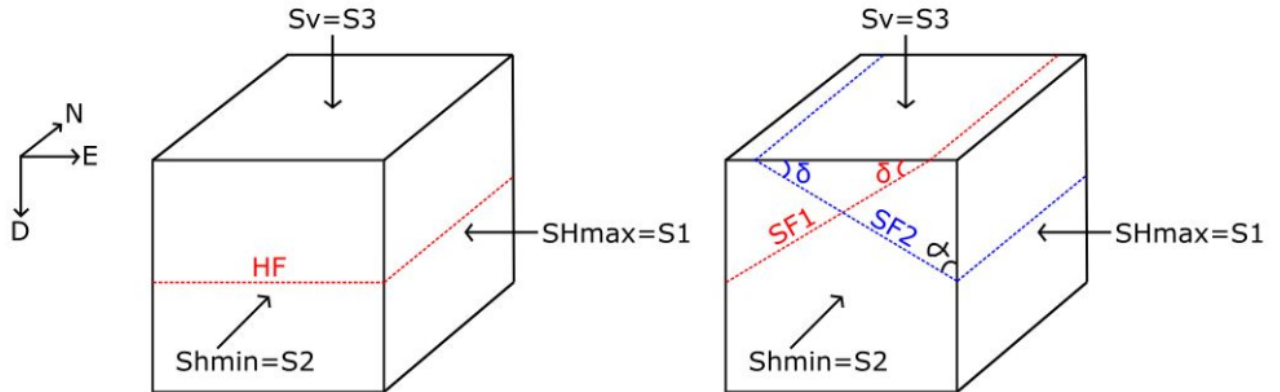
```

Out[6]: <matplotlib.legend.Legend at 0xc200350>



Problem 3

Find out the orientations of (a) a hydraulic fracture and (b) a shear fracture both created in a site subjected to stresses $S_v < S_{hmin} < S_{Hmax}$ where S_{Hmax} strikes E-W and $\mu = 0.8$.



The strike line is a line representing the intersection of the feature of interest with a horizontal plane. The magnitude of the strike (ϕ) is the angle between the strike line and north. Strike can be given in quadrant convention (N45°E) or in azimuth convention (a single three digit number ranging from 0 to 360° measured from the North in clockwise direction (045°)).

The dip (δ) is the angle between a horizontal plane and the feature under consideration. Dip is given by the number (0°-90°) as well as a letter (N,S,E,W) with rough direction in which the bed is dipping downwards.

We have a reverse fault since $S_v < S_{hmin} < S_{Hmax}$.

a) For hydraulic fracture,

Hydraulic fracture will propagate along a plane perpendicular to the least principal stress. In this case, the hydraulic fracture would propagate in the horizontal direction perpendicular to S_v .

Strike (ϕ) = 090° (E-W)

Dip (δ) = 90°

b) For shear fracture,

Orientation of shear fracture (α) is expressed in degrees from a vertical plane perpendicular to S_{Hmax} :

$$\phi = \tan^{-1}(\mu) = \tan^{-1}(0.8) = 38.7^\circ$$

$$\alpha = 45^\circ + \frac{\phi}{2} = 45 + \frac{38.7^\circ}{2} = 64.3^\circ$$

The hanging wall block will move up along the direction of S_{Hmax} relative to the foot wall.

For shear fracture 1 (SF1),

Strike (ϕ) = 000° (N-S)

Dip (δ) = 25.7°W

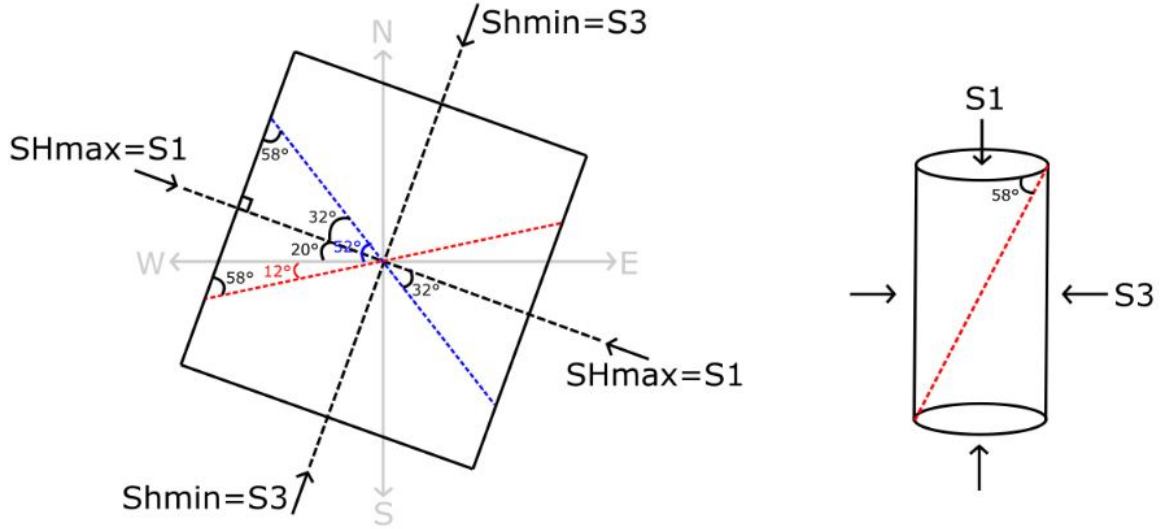
For shear fracture 2 (SF2),

Strike (ϕ) = 000° (N-S)

Dip (δ) = 25.7°E

Problem 4

Find out the orientations of (a) a hydraulic fracture and (b) a shear fracture both created in a site subjected to stresses $S_{hmin} < S_v < S_{Hmax}$ where S_{Hmax} strikes W20°N and $\mu = 0.5$.



We have a strike-slip fault since $S_{hmin} < S_v < S_{Hmax}$

a) For hydraulic fracture,

Hydraulic fracture will propagate along a plane perpendicular to the least principal stress. In this case, the hydraulic fracture would propagate in a direction perpendicular to S_{Hmax} .

Strike (ϕ) = 110° (W20°N or E20°S)

Dip (δ) = 90°

b) For shear fracture,

Orientation of shear fracture (β) is expressed in degrees from a vertical plane perpendicular to S_{Hmax} :

$$\varphi = \tan^{-1}(\mu) = \tan^{-1}(0.5) = 26^\circ$$

$$\beta = 45^\circ + \frac{\varphi}{2} = 45^\circ + \frac{26^\circ}{2} = 58^\circ$$

For shear fracture 1 (SF1),

Strike (ϕ) = 078° (W12°S)

Dip (δ) = 90°

For shear fracture 2 (SF2),

Strike (ϕ) = 142° (W52°N)

Dip (δ) = 90°

Problem 5

For the following faults: (i) draw the corresponding points in a stereonet (lower hemisphere projection) map, (ii) convert to strike to azimuth convention, and (iii) draw equivalent geological map symbols.

a) N55°E, 45°SE

b) E20°S, 60°NE

c) N20°W, 25°SW

d) S10°W, 60°NW

```
In [7]: import matplotlib.pyplot as plt
import matplotlib.transforms as mtransforms
import numpy as np
import mplstereonet

# Basic quadrant and strike/dip parsing (https://github.com/joferkington/mplstereonet)
print('Parse quadrant azimuth measurements')
for original in ['N55E', 'E20S', 'N20W', 'S10W']:
    azi = mplstereonet.parse_quadrant_measurement(original)
    print("{}" --> {:.1f}'.format(original, azi))

print('\nParse quadrant strike/dip measurements.')
def parse_sd(original, separator):
    strike, dip = mplstereonet.parse_strike_dip(*original.split(separator))
    print("{}" --> Strike: {:.1f}, Dip: {:.1f}'.format(original, strike, dip))

parse_sd('N55E/45SE', '/')
parse_sd('E20S/60NE', '/')
parse_sd('N20W/25SW', '/')
parse_sd('S10W/60NW', '/')

# Strike and dip data to be plotted
strike = np.array([55, 290, 160, 190])
dip = np.array([45, 60, 25, 60])
legend = ['a) N55°E/45°SE', 'b) E20°S/60°NE', 'c) N20°W/25°SW', 'd) S10°W/60°NW']

# Convert data to polar coordinates
theta = (strike + 90)/180 * np.pi
r = dip

# Polar plot
ax = plt.subplot(111, projection='polar')
# Change plot size
fig = plt.gcf()
fig.set_size_inches(10, 10)
# Plot size
fig = plt.figure(figsize=(15, 20))
# Polar plot properties
ax.set_theta_zero_location("N")
ax.set_theta_direction(-1)
ax.set_xticklabels(['$0°$ (N)', '$45°$', '$90°$ (E)', '$135°$', '$180°$ (S)', '$225°$', '$270°$ (W)', '$315°$'])
ax.set_rlabel_position(70)
# Axis range
ax.set_ylim(0, 90)

for i in range(len(strike)):
    ax.scatter(theta[i], r[i], label=legend[i])
    ax.text(theta[i] * (1 + 0), r[i] * (1 + 0), '%d, %d' % (strike[i], r[i]))

ax.legend(loc="center right", bbox_to_anchor=(0.8, 0, 0.5, 1))
plt.show()
```

Parse quadrant azimuth measurements

"N55E" --> 55.0

"E20S" --> 110.0

"N20W" --> 340.0

"S10W" --> 190.0

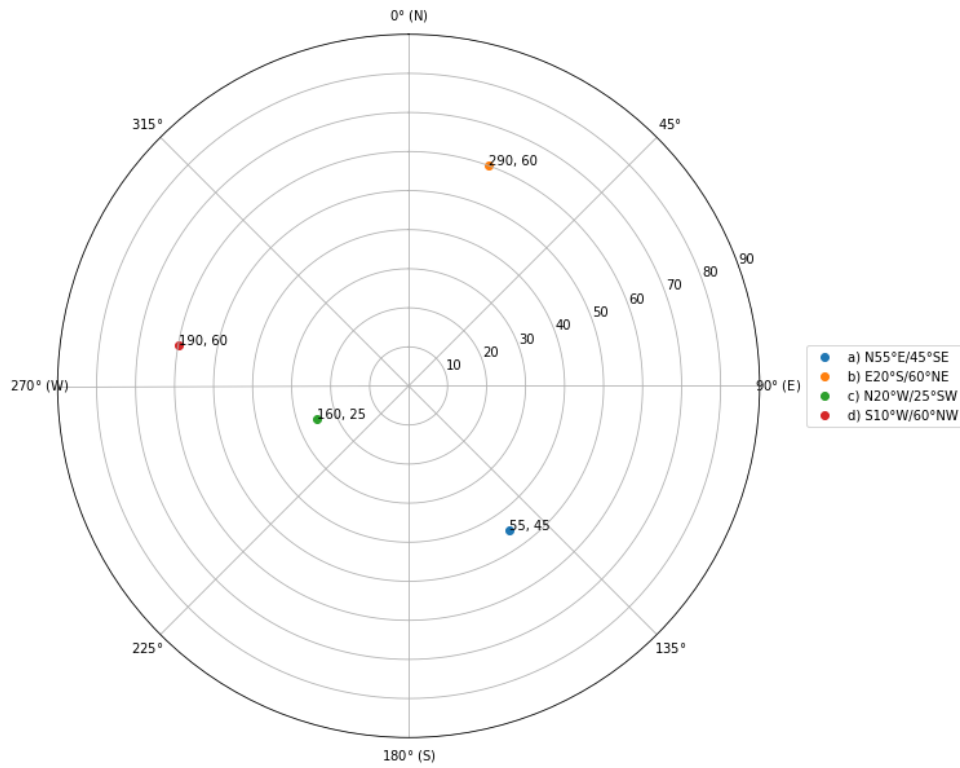
Parse quadrant strike/dip measurements.

"N55E/45SE" --> Strike: 55.0, Dip: 45.0

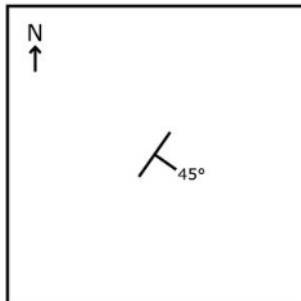
"E20S/60NE" --> Strike: 290.0, Dip: 60.0

"N20W/25SW" --> Strike: 160.0, Dip: 25.0

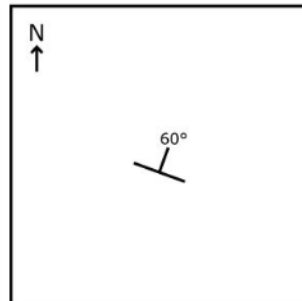
"S10W/60NW" --> Strike: 190.0, Dip: 60.0



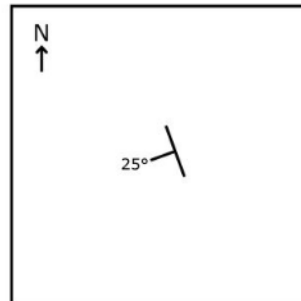
a) N55°E, 45°SE



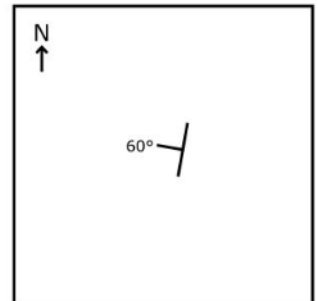
b) E20°S, 60°NE



a) N20°W, 25°SW



a) S10°W, 60°NW



Problem 6

In a given reservoir under study stresses are $S_{hmin} = 40MPa$, $S_{Hmax} = 60MPa$, $S_v = 45MPa$, $P_p = 20MPa$, and S_{hmin} acts E-W. For each of the faults below, calculate (using the Mohr Circle method if possible) the normal and shear stress on the plane of the fault and then determine if the fault is prone to slip ($\mu = 0.6$).

With $P_p = 20MPa$, we have:

$$\sigma_{hmin} = \sigma_3 = 20MPa, \sigma_{Hmax} = \sigma_1 = 40MPa, \sigma_v = \sigma_2 = 25MPa$$

In general, the shear and normal effective stresses are given by:

$$\sigma_n = C + R \times \cos(2 \times \alpha)$$

$$\tau = R \times \sin(2 \times \alpha)$$

a) Fault with strike north-south, dip 65° to the east

With $S_v > S_{hmin}$ we have dip-slip normal

$$C = \frac{\sigma_2 + \sigma_3}{2} = \frac{25 + 20}{2} = 22.5MPa$$

$$R = \frac{\sigma_2 - \sigma_3}{2} = \frac{25 - 20}{2} = 2.5MPa$$

$$\alpha = 65^\circ$$

$$\tau = 2.5 \times \sin(2 \times 65^\circ) = 1.91MPa$$

$$\sigma_n = 22.5 + 2.5 \times \cos(2 \times 65^\circ) = 20.9MPa$$

$$\frac{\tau}{\sigma_n} = 0.09 < 0.6 \text{ implies no slip}$$

b) Fault with strike east-west, dip 35° to the north

With $S_{Hmax} > S_v$ we have dip-slip reverse

$$C = \frac{\sigma_1 + \sigma_2}{2} = \frac{40 + 25}{2} = 32.5MPa$$

$$R = \frac{\sigma_1 - \sigma_2}{2} = \frac{40 - 25}{2} = 7.5MPa$$

$$\alpha = 55^\circ$$

$$\tau = 7.5 \times \sin(2 \times 55^\circ) = 7.0MPa$$

$$\sigma_n = 32.5 + 7.5 \times \cos(2 \times 55^\circ) = 29.9MPa$$

$$\frac{\tau}{\sigma_n} = 0.23 < 0.6 \text{ implies no slip}$$

c) Fault with strike 060°, dip 90°

With $S_{Hmax} > S_{hmin}$ we have strike-slip left-lateral

$$C = \frac{\sigma_1 + \sigma_3}{2} = \frac{40 + 20}{2} = 30MPa$$

$$R = \frac{\sigma_1 - \sigma_3}{2} = \frac{40 - 20}{2} = 10MPa$$

$$\alpha = 30^\circ$$

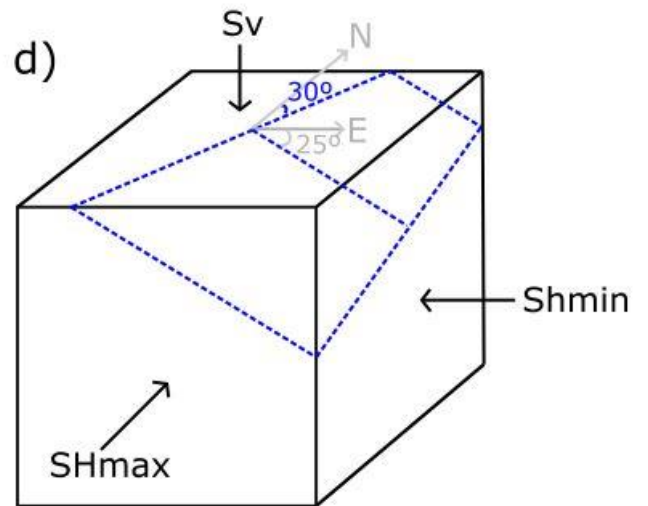
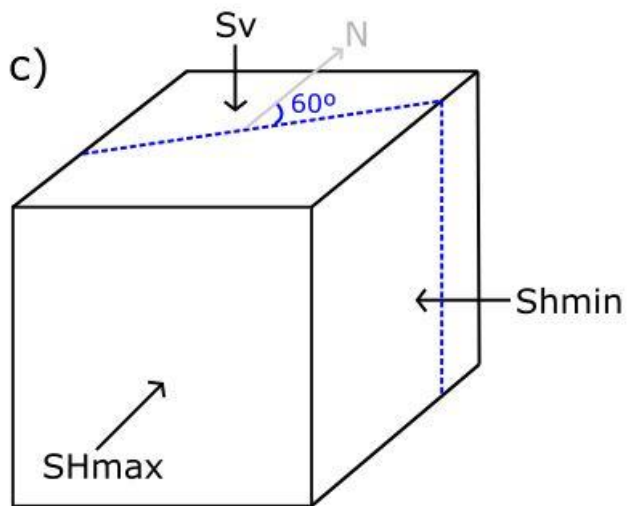
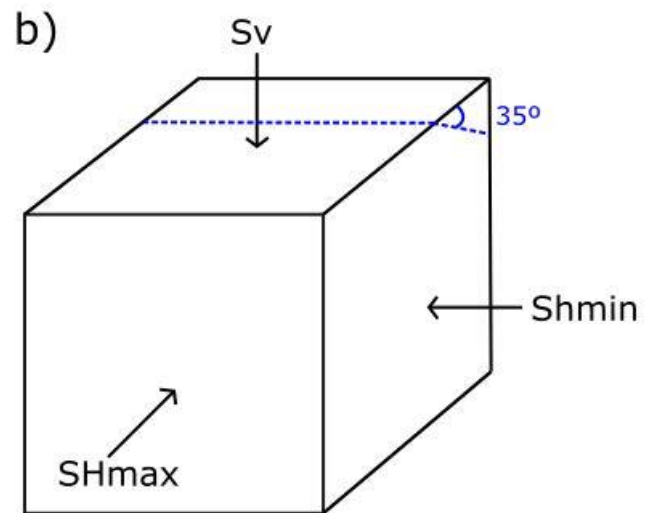
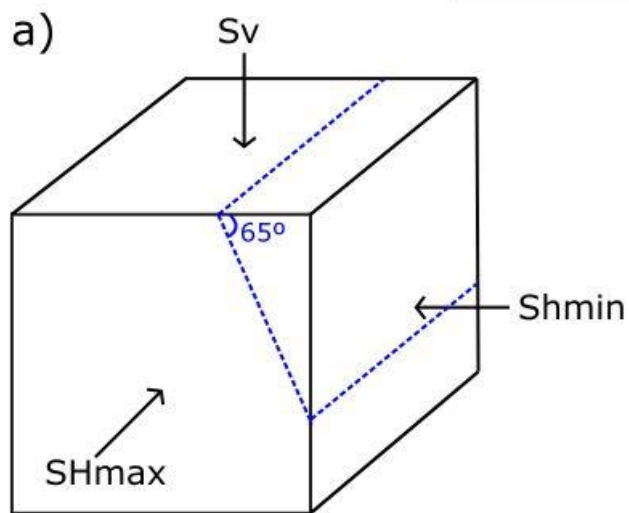
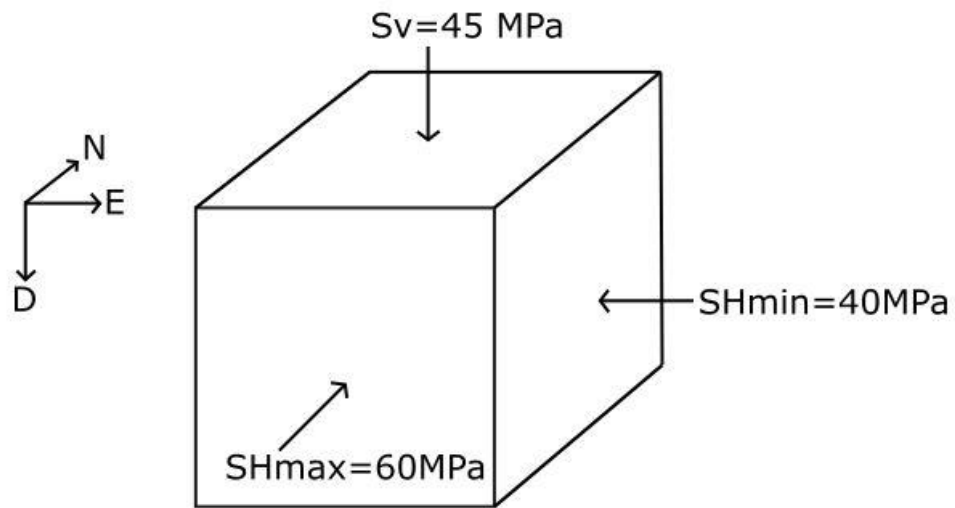
$$\tau = 10 \times \sin(2 \times 30^\circ) = 8.7MPa$$

$$\sigma_n = 30 + 10 \times \cos(2 \times 30^\circ) = 35MPa$$

$$\frac{\tau}{\sigma_n} = 0.24 < 0.6 \text{ implies no slip}$$

d) Fault with strike 030°, dip 25°

Oblique can't solve so easily...



```

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

# Function for Mohr's Circle
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))
        circle1Y.append((sig2-sig3)/2*np.sin(i) )
        circle2X.append((sig1-sig2)/2*np.cos(i) + (sig2+(sig1-sig2)/2))
        circle2Y.append((sig1-sig2)/2*np.sin(i) )
        circle3X.append((sig1-sig3)/2*np.cos(i) + (sig3+(sig1-sig3)/2))
        circle3Y.append((sig1-sig3)/2*np.sin(i) )
    plt.plot(circle1X,circle1Y,'r')
    plt.plot(circle2X,circle2Y,'g')
    plt.plot(circle3X,circle3Y,'b')
    # 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, 50])
    plt.ylim([0, 16])

# Plot 2d Mohr Circles
plotMohr3D(20,25,40) # overpressure parameter of 0.78

# Change plot size
fig = plt.gcf()
fig.set_size_inches(10, 15)

```

