Geomec-HW8

November 20, 2023

Advanced Geomechanics (PGE383) - Fall 2023

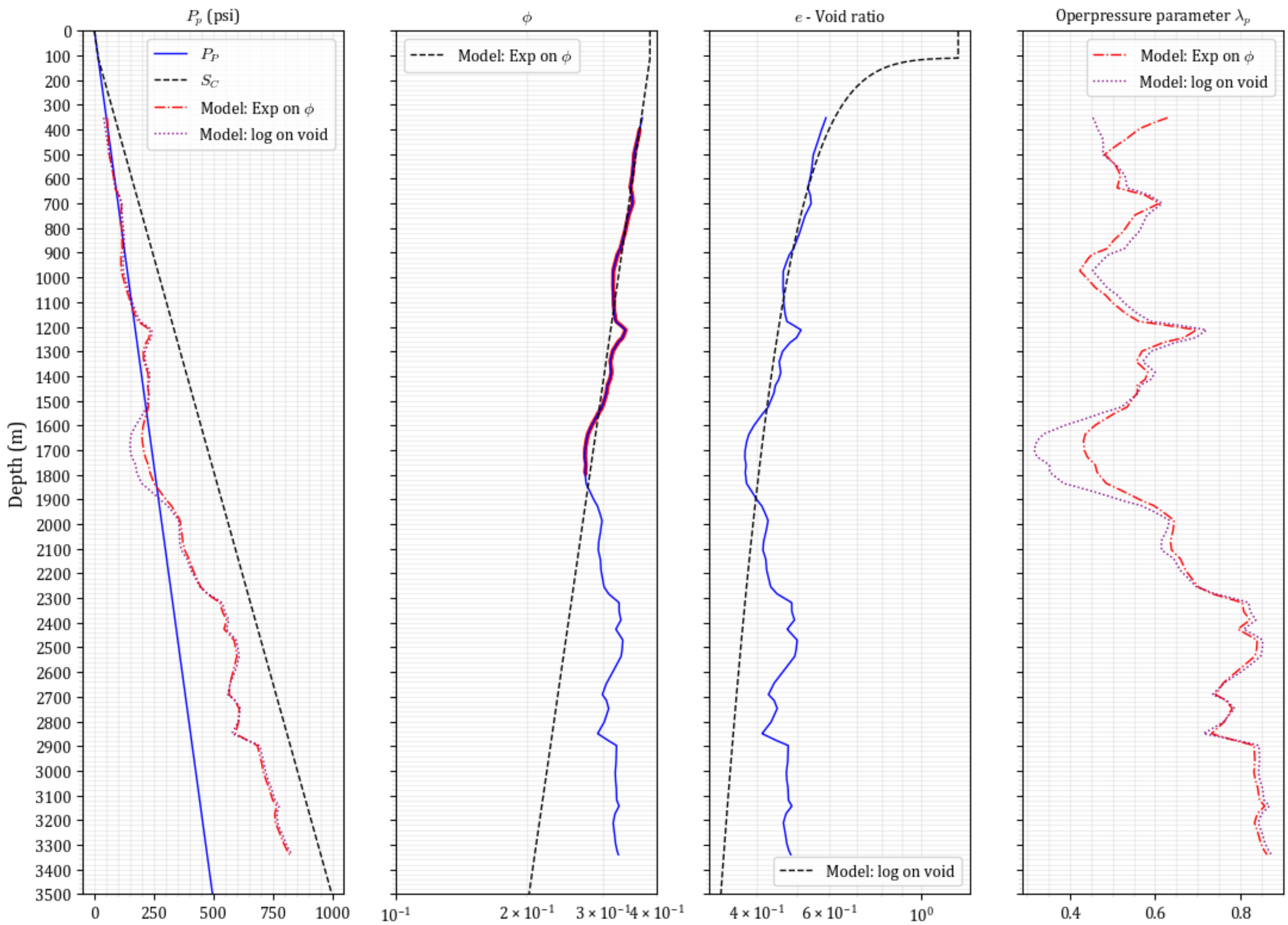
Renato Poli - rep2656

Exercise 1

1. Compute and plot pore pressure assuming a hypothetical hydrostatic pore pressure gradient $dP_p/dz = 0.465$ psi/ft.
2. Compute and plot total vertical stress assuming $dS_v/dz = 0.950$ psi/ft and pick the seafloor from the shallowest data point in "percent sand" plot.
3. Digitize shale porosity data (at least 20 equally spaced points) and fit an equation of porosity as a function of vertical effective stress from depth 400 m to 1800 m assuming hydrostatic pore pressure and models:
   - Exponential on porosity: $\phi = \phi_0 \exp{(-\beta\sigma_v)}$
   - Logarithmic on void ratio: $e = e_0 - C_c \ln\left(\frac{\sigma_v}{1\ \mathrm{MPa}}\right)$
   - Show the porosity-effective vertical stress and void ratio-effective vertical stress plots.
4. Calculate and plot actual pore pressure between the interval 1800 m to 3400 m assuming porosity is a function of vertical effective stress with the models calculated in point 3. 1; Calculate and plot overpressure parameter λ_p as a function of depth. Summarize all results with plots of:
   - (Left) Porosity (model and data) in log scale as a function of depth (y-axis)
   - (Middle) S_v and actual P_p as a function of depth (y-axis)
   - (Right) Overpressure parameter as a function of depth (y-axis)

1]:
```python
import numpy as np
from numpy import min, max
import matplotlib.pyplot as plt
import pandas as pd
from scipy.interpolate import interp1d

plt.style.use('default')   ## reset!
plt.style.use('paper.mplstyle')

DEPTH = np.arange(0, 3500, 1)
PP = 0.465 * 0.3048 * DEPTH

# Seaflor: 111.5m
GRAD = 0.950 * np.ones_like(DEPTH) #  > 111.5 m
GRAD[DEPTH<111.4] = 0.465 # See water gradient psi/ft
GRAD = GRAD * 0.3048  # to psi/m
SV = np.cumsum(GRAD)


df = pd.read_csv( 'hw8-por.csv' )
lin = interp1d( df.depth, df.por )
pormap = (DEPTH > df.depth.min()) & (DEPTH < df.depth.max())
DEPTH_POR = DEPTH[ pormap ]
POR = lin( DEPTH_POR )
VOID = POR / ( 1 - POR )
SV_POR = SV[pormap]

SIGV = SV - PP

porfit_map = (DEPTH > 400) & (DEPTH<1800)
POR_PORFIT = lin( DEPTH[porfit_map] )
VOID_PORFIT = POR_PORFIT / ( 1 - POR_PORFIT )

SIGV_PORFIT = SIGV[porfit_map]
DEPTH_PORFIT = DEPTH[porfit_map]

#  Fit porosity
from scipy.optimize import curve_fit
def expPhi(sigv, phi0, beta):
    return phi0 * np.exp( -beta * sigv )
popt, pcov = curve_fit(expPhi, SIGV_PORFIT, POR_PORFIT, [0.2,0.0001] )
mphi_phi0 = popt[0]
mphi_beta = popt[1]
POR_FIT = expPhi(SIGV, *popt)
# Extrapolate Pp
mphi_PP_POR = SV_POR + 1/mphi_beta * np.log( POR / mphi_phi0 )
mphi_lambdap = mphi_PP_POR / SV_POR # Overpressure parameter - lambda_p = pp/sv
```

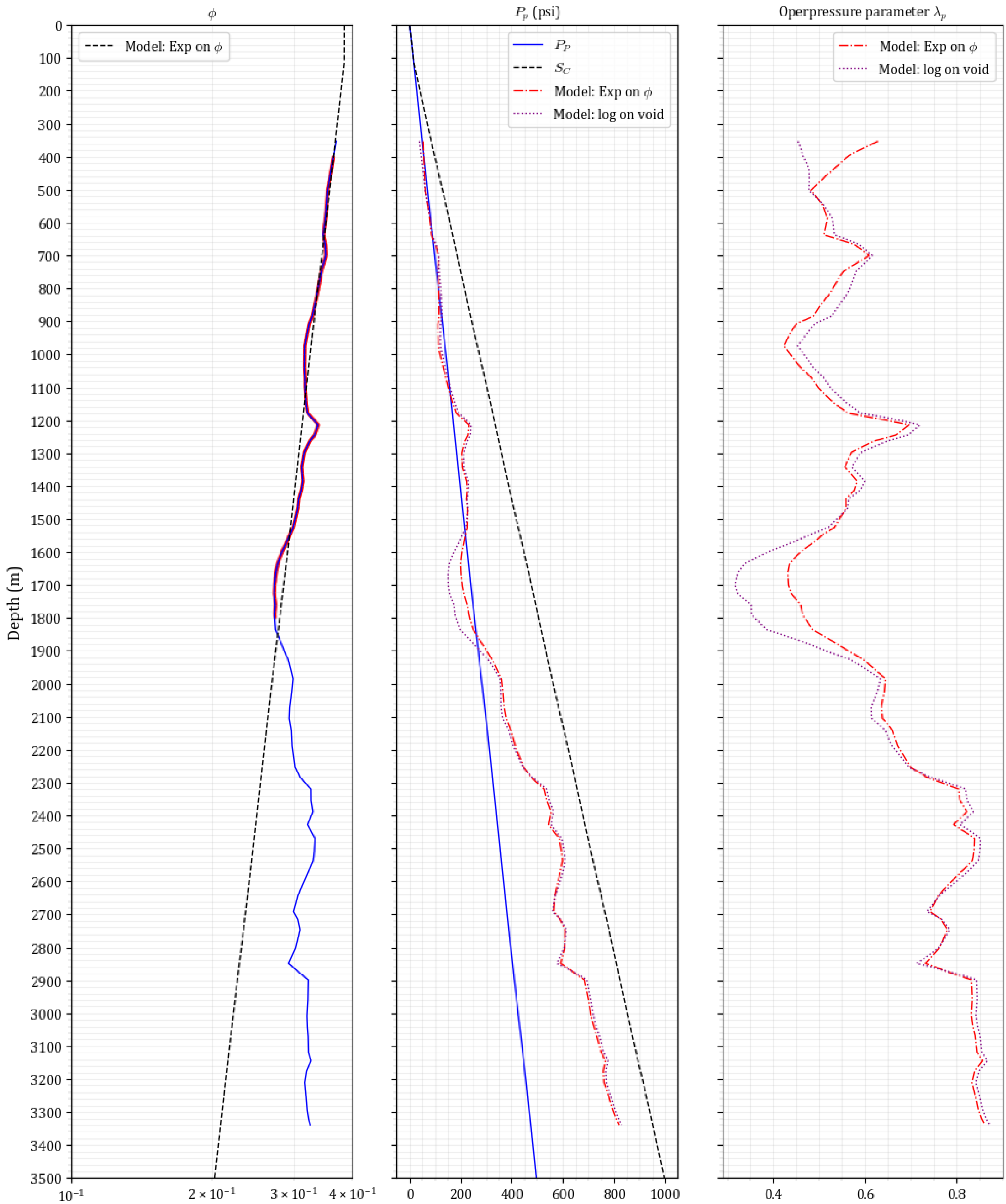
```python
# Fit void ratio
def logVoid(sigv, e0, cc):
    return e0 - cc * np.log( sigv ) # sigv is given in psi. The log argument is "times psi"
popt, pcov = curve_fit(logVoid, SIGV_PORFIT, VOID_PORFIT )
mvoid_e0 = popt[0]
mvoid_cc = popt[1]
VOID_FIT = logVoid(SIGV, *popt)
# Extrapolate Pp
mvoid_PP_POR = SV_POR - np.exp(( mvoid_e0 - VOID) / mvoid_cc )
mvoid_lambdap = mvoid_PP_POR / SV_POR # Overpressure parameter - lambda_p = pp/sv


# fig, axs = plt.subplots(1, n_crv , sharey=True)
fig, axs = plt.subplots( 1, 4, sharey=True)
fig.set_size_inches(11,8)

ax = axs[0]
ax.set_ylim( min(DEPTH), max(DEPTH) )
ax.invert_yaxis()
ax.set_ylabel(f"Depth (m)")
ax.plot( PP, DEPTH, color='b', label='$P_P$' )
ax.plot( SV, DEPTH, color='k', ls='--', label='$S_C$' )
ax.plot( mphi_PP_POR, DEPTH_POR, c='r', ls='-.', label='Model: Exp on $\phi$' )
ax.plot( mvoid_PP_POR, DEPTH_POR, c='purple', ls='dotted', label='Model: log on void' )
ax.set_title( "$P_p$ (psi)" )
ax.set_yticks( np.linspace(0,3500,36) )
ax.legend()

ax = axs[1]
ax.plot( POR, DEPTH_POR, color='b' )
ax.plot( POR_FIT, DEPTH, color='k', ls='--', label='Model: Exp on $\phi$'  )
ax.scatter( POR_PORFIT, DEPTH_PORFIT, color='r', ls='--', marker='o', s=3, alpha=0.3 )
ax.set_xscale('log')
ax.set_xlim( 0.1, 0.4 )
ax.set_title("$\phi$")
ax.legend()

ax = axs[2]
ax.plot( VOID, DEPTH_POR, color='b' )
ax.plot( VOID_FIT, DEPTH, color='k', ls='--', label='Model: log on void'  )
#ax.scatter( POR_PORFIT, DEPTH_PORFIT, color='r', ls='--', marker='o', s=3, alpha=0.3 )
ax.set_xscale('log')
#ax.set_xlim( 0.1, 0.4 )
ax.set_title("$e$ - Void ratio")
ax.legend()

ax = axs[3]
ax.plot( mphi_lambdap, DEPTH_POR, color='r', ls='-.', label='Model: Exp on $\phi$' )
ax.plot( mvoid_lambdap, DEPTH_POR, color='purple', ls='dotted', label='Model: log on void'  )
#ax.scatter( POR_PORFIT, DEPTH_PORFIT, color='r', ls='--', marker='o', s=3, alpha=0.3 )
#ax.set_xlim( 0.1, 0.4 )
ax.set_title("Operpressure parameter $\lambda_p$")
ax.legend()


fig.tight_layout()
```



2]:

```python
# fig, axs = plt.subplots(1, n_crv , sharey=True)
fig, axs = plt.subplots( 1, 3, sharey=True)
fig.set_size_inches(10,12)

ax = axs[0]
ax.plot( POR, DEPTH_POR, color='b' )
ax.plot( POR_FIT, DEPTH, color='k', ls='--', label='Model: Exp on $\phi$'  )
ax.scatter( POR_PORFIT, DEPTH_PORFIT, color='r', ls='--', marker='o', s=3, alpha=0.3 )
ax.set_xscale('log')
ax.set_xlim( 0.1, 0.4 )
ax.set_title("$\phi$")
ax.legend()
ax.set_ylabel(f"Depth (m)")

ax = axs[1]
ax.set_ylim( min(DEPTH), max(DEPTH) )
ax.invert_yaxis()
ax.plot( PP, DEPTH, color='b', label='$P_P$' )
ax.plot( SV, DEPTH, color='k', ls='--', label='$S_C$' )
ax.plot( mphi_PP_POR, DEPTH_POR, c='r', ls='-.', label='Model: Exp on $\phi$' )
ax.plot( mvoid_PP_POR, DEPTH_POR, c='purple', ls='dotted', label='Model: log on void' )
ax.set_title( "$P_p$ (psi)" )
ax.set_yticks( np.linspace(0,3500,36) )
ax.legend()

ax = axs[2]
ax.plot( mphi_lambdap, DEPTH_POR, color='r', ls='-.', label='Model: Exp on $\phi$' )
ax.plot( mvoid_lambdap, DEPTH_POR, color='purple', ls='dotted', label='Model: log on void'  )
#ax.scatter( POR_PORFIT, DEPTH_PORFIT, color='r', ls='--', marker='o', s=3, alpha=0.3 )
#ax.set_xlim( 0.1, 0.4 )
ax.set_title("Operpressure parameter $\lambda_p$")
ax.legend()
```

```
fig.tight_layout()
```



Exercise 2

Write a script that simulates a (axisymmetric) triaxial loading test ($dq = 3dp'$) for a mudrock with the following properties: - Elastic shear modulus, $G = 1$ MPa; - Pre-consolidation stress, $p'_o = 250$ [kPa] - Friction angle at critical state, $\phi_{CS} = 24°$ - Loading compressibility, $\lambda = 0.25$; - Unloading compressibility, $\kappa = 0.05$; - Initial void ratio, $e_o = 1.15$;

The initial state of stress is p'= 200 kPa; $q = 0$ kPa. Load the sample until the critical state. 1. Plot the stress path q versus p'. Plot the initial yield surface and the final yield surface. Is there hardening or softening? 1. Plot q as a function of ε_q. Why does it approximate an asymptotic value? 1. Plot void ratio e as a function of p' (with p' in logarithmic scale). Why is there a clear change of slope? 1. EXTRA: Repeat the exercise from the initial condition for a uniaxial-strain stress path approximated by $dq = 0.9\, dp'$, up to $p' = 400$ kPa). Plot the stress path q versus p' and void ratio e as a function of p' (with p' in logarithmic scale). Compare the uniaxial-strain stress-path with the triaxial deviatoric loading stress path.

Equations: Incremental elastic deformations:

$$d\varepsilon^e_{p'} = \frac{\kappa}{\upsilon}\frac{dp'}{p'};\ d\varepsilon^e_q = \frac{dq}{3G}.$$

Incremental plastic deformation:

$$\begin{bmatrix} d\varepsilon^p_{p'} \\ d\varepsilon^p_q \end{bmatrix} = \frac{\lambda-\kappa}{\upsilon p'(M^2+\eta^2)} \begin{bmatrix} M^2-\eta^2 & 2\eta \\ 2\eta & \frac{4\eta^2}{M^2-\eta^2} \end{bmatrix} \begin{bmatrix} dp' \\ dq \end{bmatrix}.$$

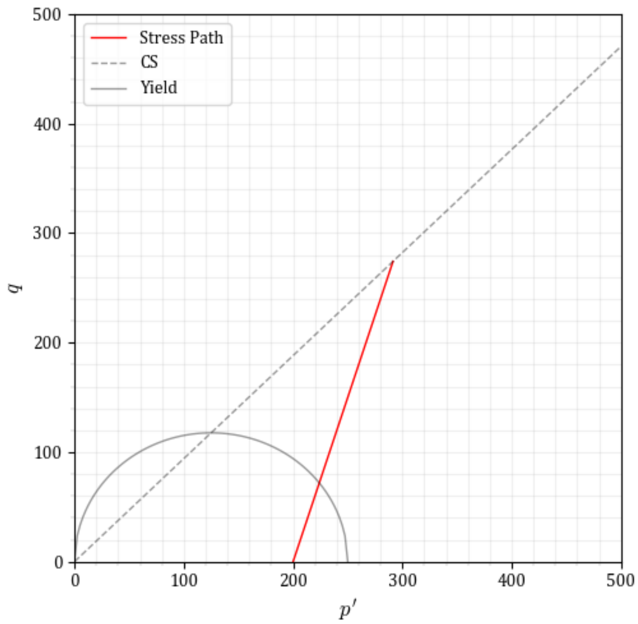
where $\upsilon = 1 + e$ is the specific volume, $\eta = q/p'$, and $de = -\upsilon d\varepsilon_p$.

The incremental change of the yield surface is: $dp'_o = d\varepsilon^p_{p'} \frac{\upsilon}{\lambda-\kappa} p'_o$.

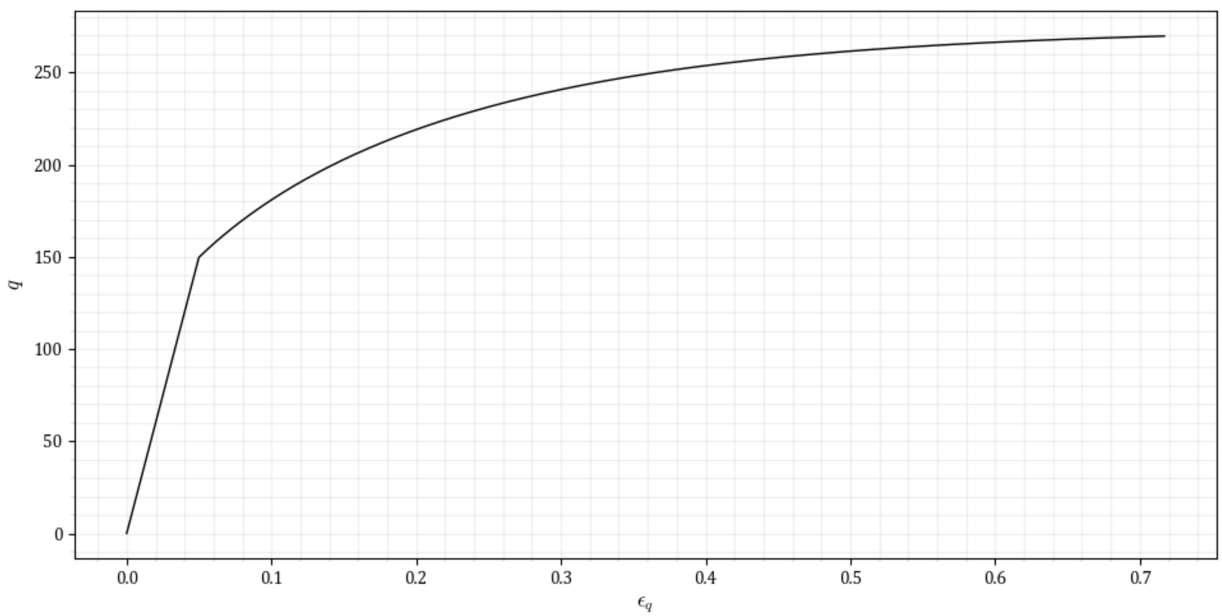
Answers: 1. Hardening. 2. Because the stress path gets to the maximum shear, on the critical state line. 3. Because of the transition from elasticity to plasticity.

```
[7]: import numpy as np
from numpy import pi, sin, cos, exp, log, min, max, sqrt, linspace
import matplotlib.pyplot as plt
plt.style.use('default')   ## reset!
plt.style.use('paper.mplstyle')

G = 1e6                      # Shear modulus [Pa]
P_0 = 250E3                  # preconsolidadtion stress [Pa]
PHIcs = 24 * pi / 180        # friction angle at critical state [rad]
LAMBDA = 0.25                 # loading compressibility
KAPPA = 0.05                 # Unloading compressibility
E0 = 1.15                    # Initial void ratio

# initial state of stress
p_ini = 200e3    # [Pa]
qini = 0         # [Pa]

# Critical state line
s = sin(PHIcs)
ss = (1+s)/(1-s)
M = 3 * (ss-1)/(2+ss)
P_cs = linspace(0, 500e3, 50 )
Qcs = M * P_cs

# Yield surface
P_y = linspace( 0, P_0, 100 )
Qy = sqrt( M**2 * P_y * (P_0-P_y) )

# Triaxial loading => dq = 3 dp_
p_max = ( qini - 3*p_ini) / ( M - 3 )
P_ = linspace( p_ini, p_max, 50 )
Q = qini + 3 * ( P_ - p_ini )

# Void ratio
P_el = linspace( p_ini, P_0, 50 )
P_pl = linspace( P_0, p_max, 50 )
VOID_EL = E0 - KAPPA * log( P_el )
VOID_PL = E0 + ( LAMBDA - KAPPA ) * log( P_0 ) - LAMBDA * log( P_pl )

# deps_q x dq
```

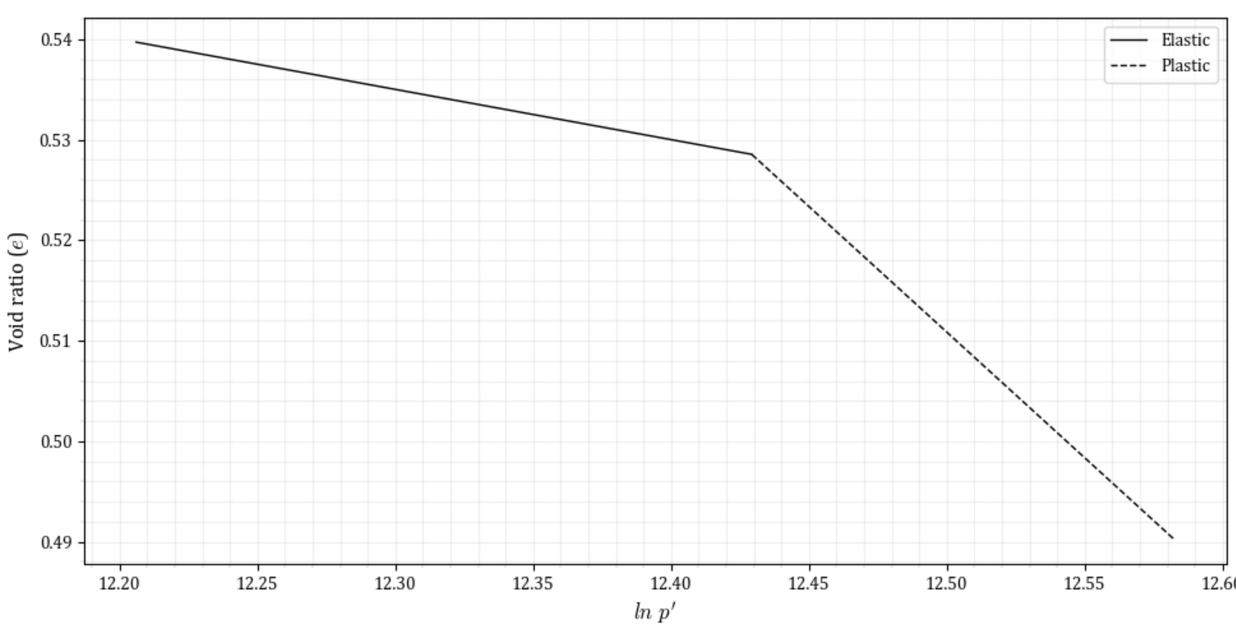
```
EPSq = np.array( [ 0 ] )
Qeps = np.array( [ 0 ] )
dp_ = 100
q = 0
p_ = p_ini
epsq = 0
for i in np.arange( 1, 900 ) :
    # Increment and continue
    dq = 3 * dp_
    p_ += dp_
    q += dq

    if ( p_ < P_0 ) :
        # Elastic
        depsq = dq / 3 / G
    else :
        # Plastic
        VOID = E0 + ( LAMBDA - KAPPA ) * log( P_0 ) - LAMBDA * log( p_ )
        eta = q / p_
        C1 = ( LAMBDA - KAPPA ) / ( 1 + VOID ) / p_ / ( M**2 + eta**2 )
        depsq =  C1 * ( ( 2 * eta ) * dp_ + 4 * eta**2 / ( M**2 - eta**2 ) * dq )

    epsq += depsq
    Qeps = np.append( Qeps,  q )
    EPSq = np.append( EPSq, epsq )


fig, axs = plt.subplots(3,1)
fig.set_size_inches(10,15)

ax=axs[0]
ax.plot( P_/1e3, Q/1e3, lw=1, c='r',label='Stress Path')
ax.plot( P_cs/1e3, Qcs/1e3, label="CS", ls='--', c='k', lw=1, alpha=.4 )
ax.plot( P_y/1e3, Qy/1e3, label="Yield", ls='-', c='k', lw=1, alpha=.4 )
ax.axis('scaled')
ax.set_xlim(0,500)
ax.set_ylim(0,500)
ax.set_ylabel("$q$")
ax.set_xlabel("$p'$")
ax.legend()

ax=axs[1]
ax.plot( EPSq, Qeps/1e3, c='k', lw=1 )
ax.set_xlabel("$\epsilon_q$")
ax.set_ylabel("$q$")

ax=axs[2]
ax.plot( log(P_el), VOID_EL, c='k', lw=1, label='Elastic' )
ax.plot( log(P_pl), VOID_PL, ls='--', lw=1, c='k', label='Plastic' )
ax.set_ylabel("Void ratio ($e$)")
ax.set_xlabel("$ln\ p'$")
ax.legend()

fig.tight_layout()
```



Stress Path
CS
Yield
q
p′



q
ϵ_q



Elastic
Plastic
Void ratio (e)
ln p′