Lined Circular Tunnel in an Elastic Medium with Anisotropic Stresses

A circular tunnel with a radius of 5 m is located at 30 m depth in a soft elastic soil. The in-situ stresses are 600 kPa vertical and 300 kPa horizontal. The tunnel is supported by a 125 mm thick shotcrete liner. The support is installed simultaneously with the excavation in a preexisting anisotropic biaxial stress field. The support displacements and stress resultants (in terms of thrust and moment), and the interface contact stresses are computed under plane-strain conditions for the two limiting conditions of no-slip (no relative shear displacement) and full-slip (no shear stress transmission) at the ground-support interface. These computed values are compared with the analytical solution of Einstein and Schwartz (1979).

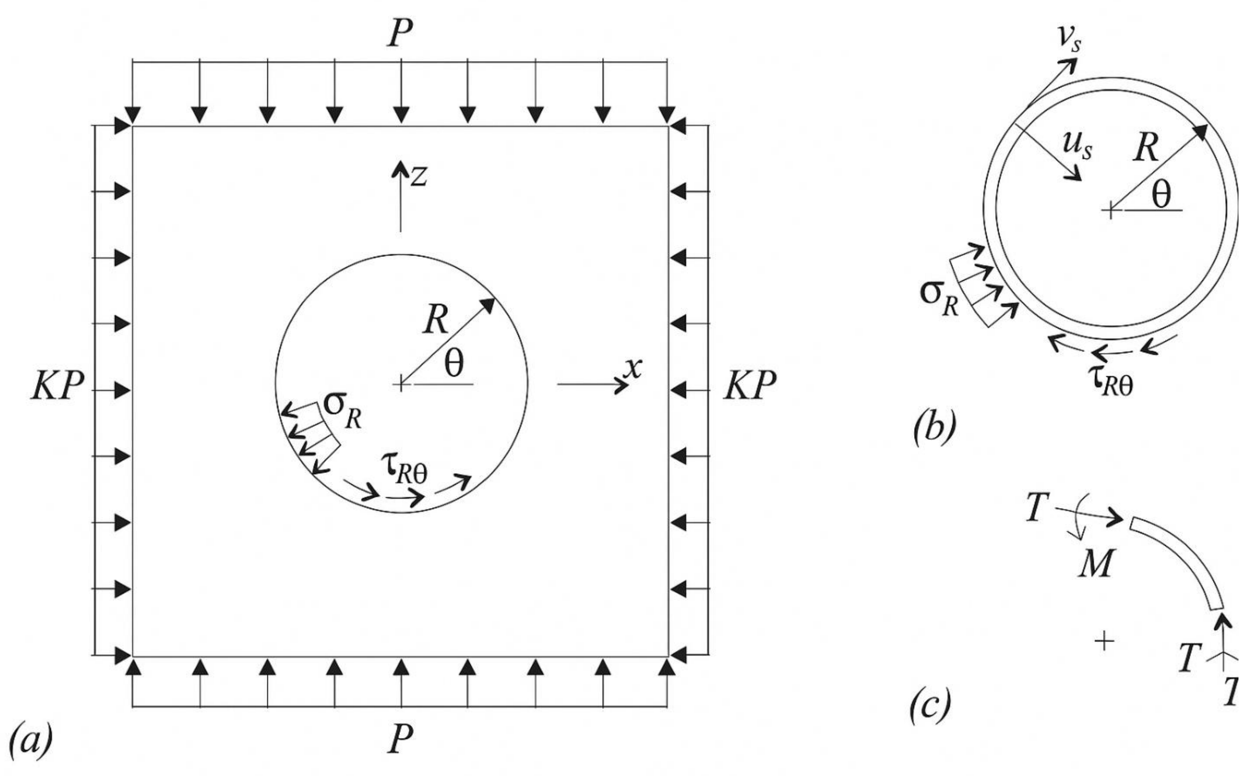
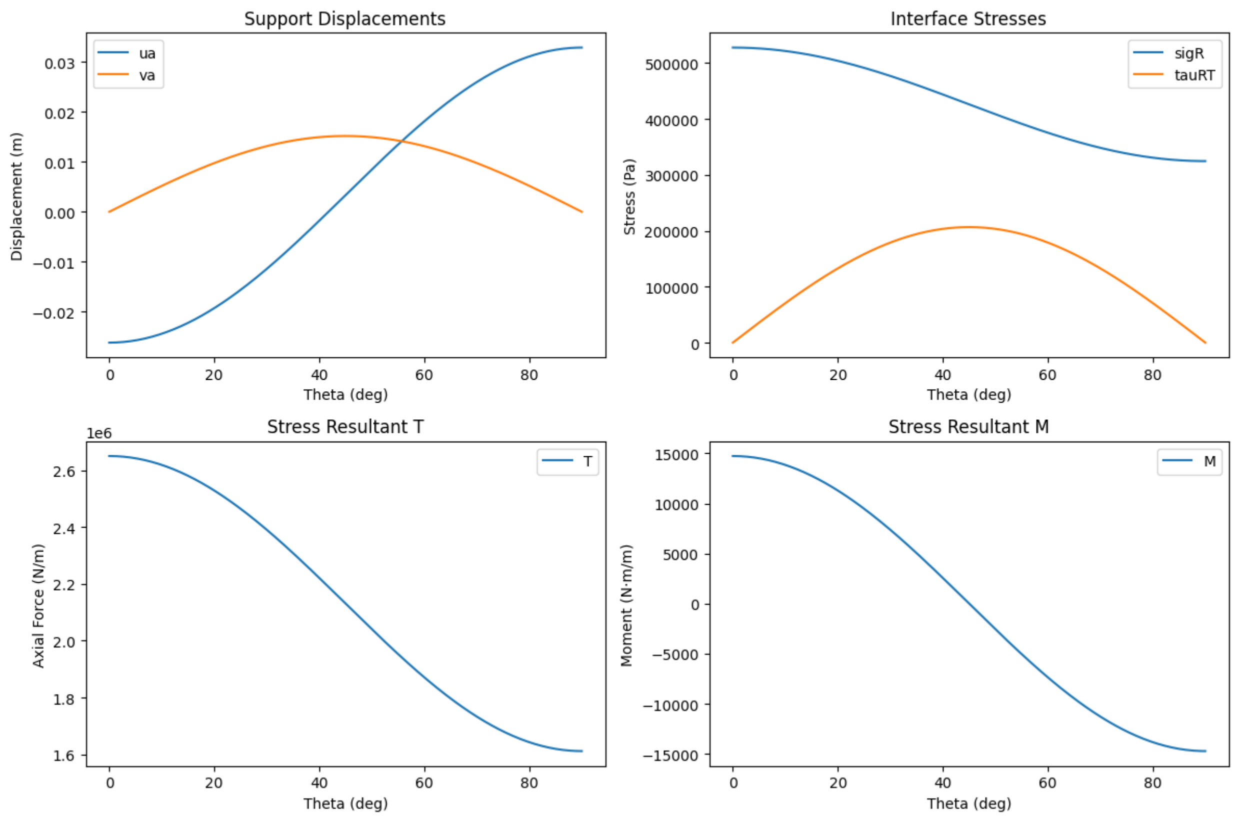
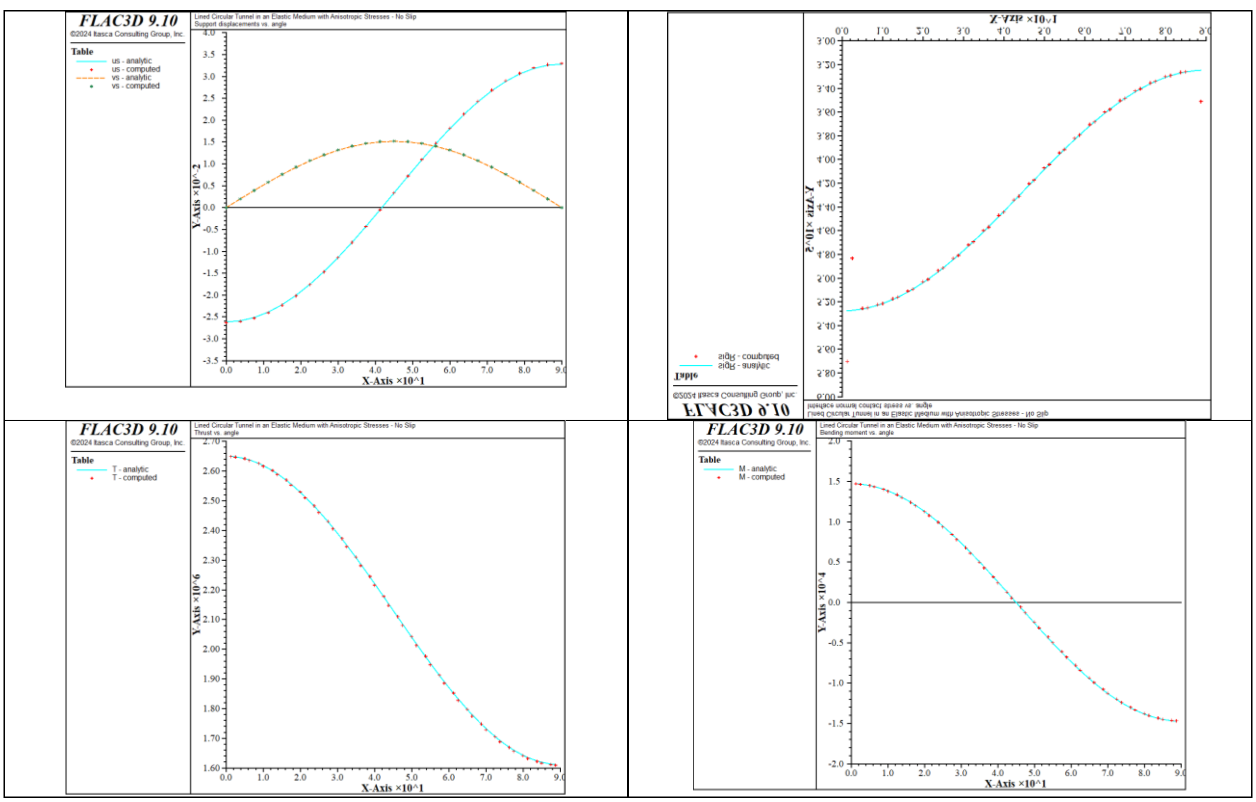


Figure 1: Notation for analytical solution: (a) ground medium; (b) tunnel liner; and (c) positive sense of internal stress resultants in liner.


Support Displacements
Displacement (m)
Theta (deg)
ua
va
Interface Stresses
Stress (Pa)
Theta (deg)
sigR
tauRT
Stress Resultant T
1e6
Axial Force (N/m)
Theta (deg)
T
Stress Resultant M
Moment (N·m/m)
Theta (deg)
M


FLAC3D 9.10
©2024 Itasca Consulting Group, Inc.
Lined Circular Tunnel in an Elastic Medium with Anisotropic Stresses - No Slip
Support displacements vs. angle
Table
us - analytic
us - computed
vs - analytic
vs - computed
Y-Axis ×10^-2
X-Axis ×10^1
FLAC3D 9.10
©2024 Itasca Consulting Group, Inc.
Lined Circular Tunnel in an Elastic Medium with Anisotropic Stresses - No Slip
Thrust vs. angle
Table
T - analytic
T - computed
Y-Axis ×10^6
X-Axis ×10^1
FLAC3D 9.10
©2024 Itasca Consulting Group, Inc.
Lined Circular Tunnel in an Elastic Medium with Anisotropic Stresses - No Slip
Bending moment vs. angle
Table
M - analytic
M - computed
Y-Axis ×10^4
X-Axis ×10^1

```
#Lined Circular Tunnel
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Parameters
def responses(no_slip=True):
    R = 5.0
    K = 0.5
    P = 6e5
    bulk = 5e7
    shear = 1.791e7
    E = 9.0 * bulk * shear / (3.0 * bulk + shear)
    nu = 0.5 * (3.0 * bulk - 2.0 * shear) / (3.0 * bulk +
shear)
    Es = 2.5e10
    nus = 0.15
    As = 0.125
    Is = As**3 / 12.0
    Cs = E * R * (1.0 - nus**2) / (Es * As * (1.0 -
nu**2))
    Fs = E * R**3 * (1.0 - nus**2) / (Es * Is * (1.0 -
nu**2))

    if no_slip:
        Beta = (6.0 + Fs) * Cs * (1.0 - nu) + 2.0 * Fs *
nu
        Beta = Beta / (3.0 * Fs + 3.0 * Cs + 2.0 * Cs *
Fs * (1.0 - nu))
        bs2 = 2.0 * Cs * (1.0 - nu) + 8.0 * nu - 12.0 *
Beta - 6.0 * Beta * Cs * (1.0 - nu)
        bs2 = Cs * (1.0 - nu) / bs2
        as0 = Cs * Fs * (1.0 - nu) / (Cs + Fs + Cs * Fs *
(1.0 - nu))
        as2 = Beta * bs2
    else:
        as0 = Cs * Fs * (1.0 - nu) / (Cs + Fs + Cs * Fs *
(1.0 - nu))
        as2 = (Fs + 6.0) * (1.0 - nu) / (2.0 * Fs * (1.0
- nu) + 6.0 * (5.0 - 6.0 * nu))

    if no_slip:
```

```
        U1 = 0.5 * (1.0 + K) * as0
        U2 = 0.5 * (1.0 - K) * (4.0 * (1.0 - nu) * bs2 - 2.0 * as2)
        V2 = (-1.0 + K) * (as2 + (1.0 - 2.0 * nu) * bs2)
        T1 = (1.0 + K) * (1.0 - as0)
        T2 = (1.0 - K) * (1.0 + 2.0 * as2)
        M2 = 0.5 * (1.0 - K) * (1.0 - 2.0 * as2 + 2.0 * bs2)
        sigR1 = (1.0 + K) * (1.0 - as0)
        sigR2 = (-1.0 + K) * (1.0 - 6.0 * as2 + 4.0 * bs2)
        tauRT2 = (1.0 - K) * (1.0 + 6.0 * as2 - 2.0 * bs2)
    else:
        U1 = 0.5 * (1.0 + K) * as0
        U2 = (-1.0 + K) * ((5.0 - 6.0 * nu) * as2 - (1.0 - nu))
        V2 = 0.5 * (1.0 - K) * ((5.0 - 6.0 * nu) * as2 - (1.0 - nu))
        T1 = (1.0 + K) * (1.0 - as0)
        T2 = (1.0 + K) * (1.0 - 2.0 * as2)
        M2 = (1.0 - K) * (1.0 - 2.0 * as2)
        sigR1 = (1.0 + K) * (1.0 - as0)
        sigR2 = (-1.0 + K) * (3.0 - 6.0 * as2)
        tauRT2 = 0.0

    theta_deg = np.linspace(0, 90, 200)
    theta = np.deg2rad(theta_deg)

    ua = (P * R * (1.0 + nu) / E) * (U1 + U2 * np.cos(2.0 * theta))
    va = (P * R * (1.0 + nu) / E) * (V2 * np.sin(2.0 * theta))

    sigRa = 0.5 * P * (sigR1 + sigR2 * np.cos(2.0 * theta))
    tauRTa = 0.5 * P * tauRT2 * np.sin(2.0 * theta)

    Ta = 0.5 * P * R * (T1 + T2 * np.cos(2.0 * theta))
    Ma = 0.5 * P * R**2 * M2 * np.cos(2.0 * theta)

    df = pd.DataFrame({
```

```
        'theta_deg': theta_deg,
        'ua': ua,
        'va': va,
        'sigR': sigRa,
        'tauRT': tauRTa,
        'T': Ta,
        'M': Ma
    })

    df.to_csv('analytic_results_0_90.csv', index=False)

    # --- Plot results ---
    plt.figure(figsize=(12, 8))
    plt.subplot(2, 2, 1)
    plt.plot(theta_deg, ua, label='ua')
    plt.plot(theta_deg, va, label='va')
    plt.xlabel('Theta (deg)')
    plt.ylabel('Displacement (m)')
    plt.title('Support Displacements')
    plt.legend()

    plt.subplot(2, 2, 2)
    plt.plot(theta_deg, sigRa, label='sigR')
    plt.plot(theta_deg, tauRTa, label='tauRT')
    plt.xlabel('Theta (deg)')
    plt.ylabel('Stress (Pa)')
    plt.title('Interface Stresses')
    plt.legend()

    plt.subplot(2, 2, 3)
    plt.plot(theta_deg, Ta, label='T')
    plt.xlabel('Theta (deg)')
    plt.ylabel('Axial Force (N/m)')
    plt.title('Stress Resultant T')
    plt.legend()

    plt.subplot(2, 2, 4)
    plt.plot(theta_deg, Ma, label='M')
    plt.xlabel('Theta (deg)')
    plt.ylabel('Moment (N·m/m)')
    plt.title('Stress Resultant M')
    plt.legend()
```

```
    plt.tight_layout()
    plt.show()

    print('Results saved to analytic_results_0_90.csv')

responses(no_slip=True)
```

Reference

Einstein, H. H., and C. W. Schwartz. "*Simplified Analysis for Tunnel Supports*," J. Geotech. Engr. Div., 105(GT4): 499-518 (1979).