Exercise 2

TKT4150 - Biomechanics

by Jan-Øivind Lima

Contents

§1) Assignment 2

§1.1) Exercise 1: Cauchy's stress theorem used on the foot of a running human

§1.1.a) Rewrite the Cauchy stress theorem in index notation.

The Cauchy stress theorem states that the traction vector t on a surface with normal vector n is given by the stress tensor σ acting on the normal vector. In index notation, this can be expressed as:

$$T_j^{(n)} = \sigma_{ij} n_i$$

where t_i is the i-th component of the traction vector, σ_{ij} is the stress tensor, and n_i is the i-th component of the normal vector.

§1.1.b) The foot of a running human is exposed to a force from the ground, as is seen in Figure 1. Let's say the stress matrix corresponding to a point on the foot where the foot hits the ground is assumed to be:

$$T = \begin{bmatrix} 0 & 20 & 0 \\ 20 & -40 & 0 \\ 0 & 0 & 0 \end{bmatrix} MPa$$

Find an estimate of the angle, α, on which the force from the ground acts on the foot.

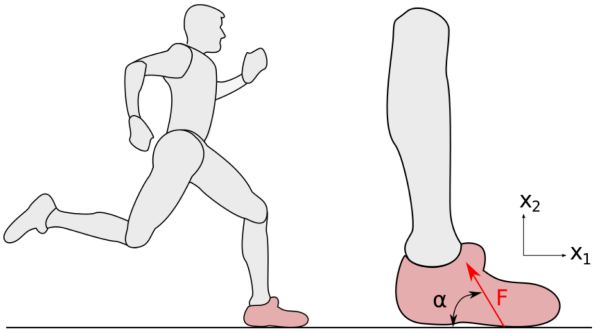

Figure 1: Running human

To find the angle α on which the force from the ground acts on the foot, we first need to determine the traction vector t using the Cauchy stress theorem. The normal vector n for the surface where the foot hits the ground can be assumed to be pointing upwards, which can be represented as $n = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$. Using the stress matrix $\sigma = \begin{bmatrix} 0 & 20 & 0 \\ 20 & -40 & 0 \\ 0 & 0 & 0 \end{bmatrix} MPa$, we can calculate the traction vector t as follows:

$$t = \sigma n$$

Calculating this gives:

$$t = \begin{bmatrix} 0 & 20 & 0 \\ 20 & -40 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 20 \\ -40 \\ 0 \end{bmatrix} MPa$$

The traction vector t has components $t_x = 20MPa$, $t_y = -40MPa$, and $t_z = 0MPa$. The angle α can be found using the arctangent function:

$$\alpha = \arctan\left(\frac{t_y}{t_x}\right)$$

Substituting the values of t_x and t_y gives:

$$\alpha = \arctan\left(-\frac{40}{20}\right) \approx \mathbf{-63.43^\circ}$$

§1.2) Exercise 2: Computational Principal Stress Analysis

For the following stress matrix:

$$\boldsymbol{T} = \begin{bmatrix} 90 & -30 & 0 \\ -30 & 120 & -30 \\ 0 & -30 & 90 \end{bmatrix} MPa$$

you will calculate the principal stresses, and directions, as well as use Python, Matlab or another computational tool of your choice to calculate these quantities as well. The file Tensor_calculations_exercise.ipynb is an IPython Notebook to help you get started if you would like. You can upload this to a Jupyter Notebook server and get started if you do not have access to Python on your local computer. Remember that to save your progress in the notebook on an external server, you likely need to download the notebook.

§1.2.a) Establish the principal stresses σ1, σ2 and σ3, and the corresponding directions n1, n2 and n3 with hand-calculations.

To find the principal stresses and directions, we need to solve the characteristic equation given by the determinant of $(\sigma I - T) = 0$, where I is the identity matrix and σ represents the principal stresses. The stress matrix is given as:

$$T = \begin{bmatrix} 90 & -30 & 0 \\ -30 & 120 & -30 \\ 0 & -30 & 90 \end{bmatrix} MPa$$

We first compute the matrix $(\sigma I - T)$:

$$\sigma I - T = \begin{bmatrix} \sigma - 90 & 30 & 0 \\ 30 & \sigma - 120 & 30 \\ 0 & 30 & \sigma - 90 \end{bmatrix}$$

Next, we calculate the determinant of this matrix and set it to zero:

$$\det(\sigma I - T) = (\sigma - 90)((\sigma - 120)(\sigma - 90) - 900) - 30(30(\sigma - 90))$$

Expanding this determinant, we get the characteristic polynomial:

$$\sigma^3 - 300\sigma^2 + 27000\sigma - 720000 = 0$$

To find the roots of this cubic polynomial, we can use numerical methods or factorization. The roots of this polynomial give us the principal stresses:

$$\sigma_1 \approx 150MPa$$

$$\sigma_2 \approx 90MPa$$

$$\sigma_3 \approx 60MPa$$

Next, we find the corresponding principal directions by solving the equation $(T - \sigma_i I)n_i = 0$ for each principal stress σ_i.

§1.2.b) Find the numerical values for the three stress invariants I, II and III, using Python or Matlab.

```python
import numpy as np

T = np.array([[90, -30, 0], [-30, 120, -30], [0, -30, 90]])
```

```python
# Calculating the invariants
I = np.trace(T)
II = 0.5 * (I**2 - np.trace(np.dot(T, T)))
III = np.linalg.det(T)

print("I={}".format(I))
print("II={}".format(II))
print("III={}".format(III))
```

The output of the Python code gives the stress invariants as:

$$I_1 = 300MPa$$

$$I_2 = 27900(MPa)^2$$

$$I_3 = 810000(MPa)^3$$

§1.2.c) Use the invariants to determine the characteristic polynomial from the eigenvalue problem. Furthermore, plot the polynomial and graphically determine the principal stresses.

Using the stress invariants I_1, I_2, and I_3, we can construct the characteristic polynomial for the eigenvalue problem as follows:

$$p(\sigma) = \sigma^3 - I_1\sigma^2 + I_2\sigma - I_3$$

Substituting the values of the invariants, we get:

$$p(\sigma) = \sigma^3 - 300\sigma^2 + 27900\sigma - 810000$$

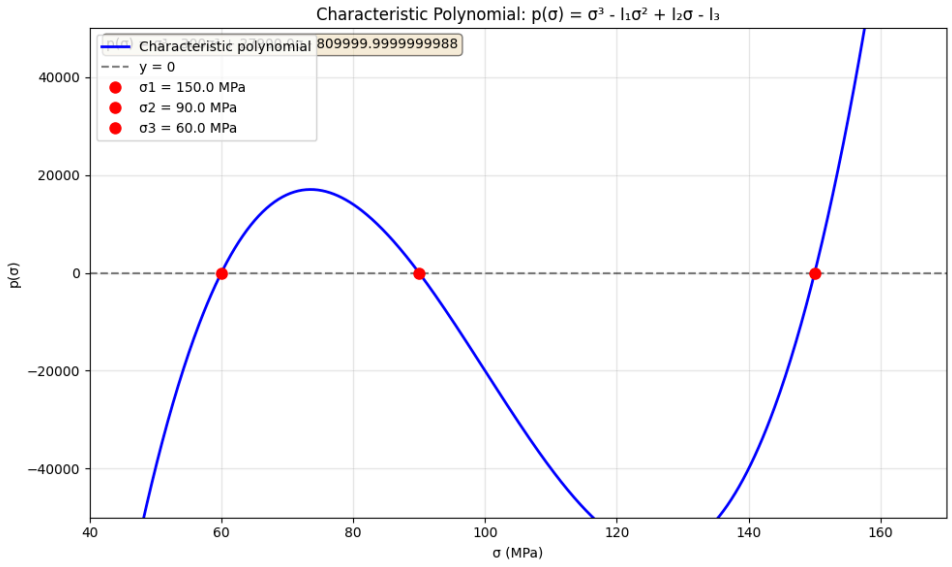

Figure 2: Characteristic polynomial plot, generated by the python code.

To graphically determine the principal stresses, we can plot this polynomial and look for the points where it intersects the x-axis (i.e., where $p(\sigma) = 0$). The x-coordinates of these intersection points correspond to the principal stresses.

The plot of the characteristic polynomial shows that the principal stresses are approximately:

$$\sigma_1 \approx 150MPa$$

$$\sigma_2 \approx 90MPa$$

$$\sigma_3 \approx 60MPa$$

§1.2.d) Use Python or Matlab to find the solutions of the characteristic polynomial.

The output from the code show in the appendix gives the principal stresses as:

$$\sigma_1 = 150.0MPa$$

$$\sigma_2 = 90.0MPa$$

$$\sigma_3 = 60.0MPa$$

§1.2.e) Use the linear algebra library of Python or Matlab to directly solve for the principal stresses and corresponding directions.

Hint: Use eig to direclty solve for the principal stresses and corresponding direction. np.linalg.eig??

Using the linear algebra library in Python, we can directly compute the eigenvalues and eigenvectors of the stress matrix T using the `np.linalg.eig` function. The eigenvalues correspond to the principal stresses, and the eigenvectors correspond to the principal directions.

The output from the code show in the appendix gives the principal stresses as:

$$\sigma_1 = 150.0MPa$$

$$\sigma_2 = 90.0MPa$$

$$\sigma_3 = 60.0MPa$$

§1.2.f) Plot the principal direction vectors ni (i=1,2,3) using Python or Matlab. (Make sure it is possible to see how they are oriented relative to each other. It may be necessary to have two figures from different angles.)

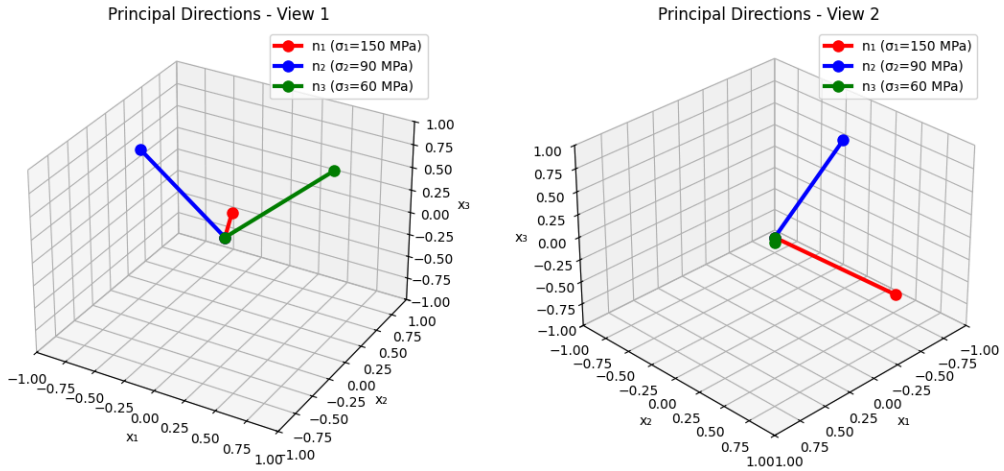

Figure 3: 3D plot of principal directions from two different angles, generated by the python code.

§1.2.g) The maximum shear stress is found in an orientation between the directions of the smallest and largest principal stresses, σ1 and σ3, and in the plane normal to the direction of σ2. Determine the orientation where the stress element gets the largest shear stress, and the numerical value of this stress. Plot the found normal vector together with the vectors in the previous problem.

The maximum shear stress $\tau_{\max}$ can be calculated using the formula:

$$\tau_{\max} = \frac{\sigma_1 - \sigma_3}{2}$$

Substituting the values of the principal stresses, we get:

$$\tau_{\max} = \frac{150MPa - 60MPa}{2} = 45MPa$$

The orientation where the stress element experiences the largest shear stress is in the plane normal to the direction of σ_2. The normal vector can be found by taking the cross product of the principal direction vectors corresponding to σ_1 and σ_3.

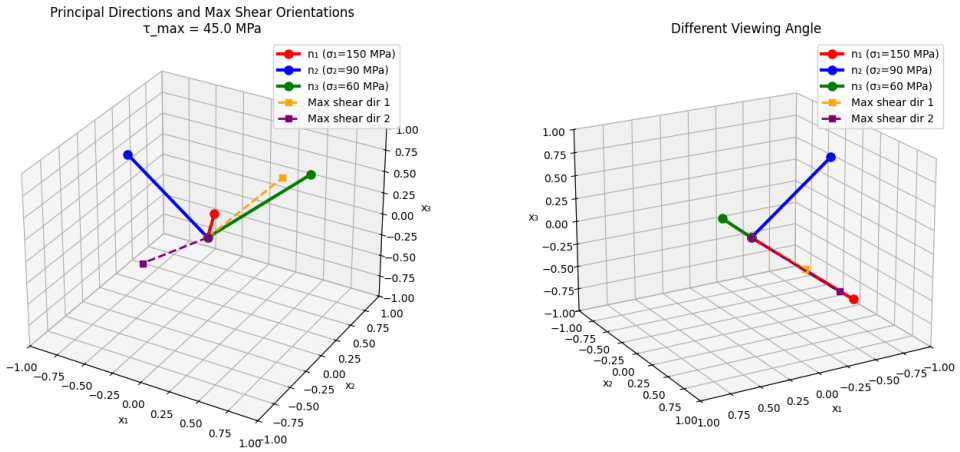

Figure 4: 3D plot of principal directions and maximum shear stress direction, generated by the python code.

§2) Appendix - Python code

§2.1) 1.2.b) Calculation of stress invariants

```python
import numpy as np

T = np.array([[90, -30, 0], [-30, 120, -30], [0, -30, 90]])

# Calculating the invariants
I = np.trace(T)
II = 0.5 * (I**2 - np.trace(np.dot(T, T)))
III = np.linalg.det(T)

print("I={}".format(I))
print("II={}".format(II))
print("III={}".format(III))
```

§2.2) 1.3.c,d,e,f) Plotting the characteristic polynomial, and linalg etc...

```python
"""
Exercise 2.1.3 - Principal Stress Analysis
==========================================

This script performs comprehensive principal stress analysis including:
- Task 1.2.c: Plotting the characteristic polynomial
- Task 1.2.d: Determining principal stresses graphically
- Task 1.2.e: Verifying results with linear algebra
- Task 1.2.f: Computing and visualizing principal directions
- Task 1.2.g: Maximum shear stress analysis
"""

import numpy as np
import matplotlib.pyplot as plt

# ============================================================================
# SETUP - Define stress matrix and calculate invariants
# ============================================================================

# Define the stress matrix
T = np.array([[90, -30, 0], [-30, 120, -30], [0, -30, 90]])

# Calculate the stress invariants
I1 = np.trace(T)
I2 = 0.5 * (I1**2 - np.trace(np.dot(T, T)))
I3 = np.linalg.det(T)

print("=" * 60)
print("EXERCISE 2.1.3 - PRINCIPAL STRESS ANALYSIS")
print("=" * 60)
print("Stress invariants:")
print(f"I1 = {I1} MPa")
print(f"I2 = {I2} MPa^2")
print(f"I3 = {I3} MPa^3")

# ============================================================================
# TASK 1.2.c) - Plot the characteristic polynomial
```

```
# ============================================================================

print("\n" + "-" * 60)
print("TASK 1.2.c) - Characteristic Polynomial Plot")
print("-" * 60)

# Define the characteristic polynomial: p(σ) = σ³ - I1*σ² + I2*σ - I3
def characteristic_polynomial(sigma):
    return sigma**3 - I1*sigma**2 + I2*sigma - I3

# Create a range of sigma values for plotting
sigma_range = np.linspace(0, 200, 1000)
polynomial_values = characteristic_polynomial(sigma_range)

# ============================================================================
# TASK 1.2.d) - Determine principal stresses graphically
# ============================================================================

print("\nTASK 1.2.d) - Graphical determination of principal stresses")
print("-" * 60)

# Find the roots (principal stresses) using numpy
coefficients = [1, -I1, I2, -I3]  # Coefficients for σ³ - I1*σ² + I2*σ - I3
principal_stresses = np.roots(coefficients)
principal_stresses = np.sort(principal_stresses)[::-1]  # Sort in descending order

print("Principal stresses from polynomial roots:")
print(f"σ1 = {principal_stresses[0]:.1f} MPa")
print(f"σ2 = {principal_stresses[1]:.1f} MPa")
print(f"σ3 = {principal_stresses[2]:.1f} MPa")

# Create the characteristic polynomial plot
plt.figure(figsize=(10, 6))
plt.plot(
    sigma_range,
    polynomial_values,
    'b-',
    linewidth=2,
    label='Characteristic polynomial'
)
plt.axhline(y=0, color='k', linestyle='--', alpha=0.5, label='y = 0')

# Mark the principal stresses (roots) on the plot
for i, stress in enumerate(principal_stresses):
    plt.plot(stress, 0, 'ro', markersize=8, label=f'σ{i+1} = {stress:.1f} MPa')

plt.xlabel('σ (MPa)')
plt.ylabel('p(σ)')
plt.title('Characteristic Polynomial: p(σ) = σ³ - I₁σ² + I₂σ - I₃')
plt.grid(True, alpha=0.3)
plt.legend()
plt.xlim(40, 170)
plt.ylim(-50000, 50000)

# Add text box with polynomial equation
textstr = f'p(σ) = σ³ - {I1}σ² + {I2}σ - {I3}'
```

```
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
plt.text(0.02, 0.98, textstr, transform=plt.gca().transAxes, fontsize=10,
         verticalalignment='top', bbox=props)

plt.tight_layout()
plt.show()

# ============================================================================
# TASK 1.2.e) - Verify results using linear algebra
# ============================================================================

print("\nTASK 1.2.e) - Verification using numpy.linalg.eigvals")
print("-" * 60)

# Verify by computing eigenvalues directly
eigenvalues = np.linalg.eigvals(T)
eigenvalues = np.sort(eigenvalues)[::-1]
print("Principal stresses using numpy.linalg.eigvals:")
print(f"σ1 = {eigenvalues[0]:.1f} MPa")
print(f"σ2 = {eigenvalues[1]:.1f} MPa")
print(f"σ3 = {eigenvalues[2]:.1f} MPa")

# ============================================================================
# TASK 1.2.f) - Principal stresses and directions using eig
# ============================================================================

print("\nTASK 1.2.f) - Principal stresses and directions")
print("-" * 60)

# Use np.linalg.eig to get both eigenvalues and eigenvectors
eigenvalues_eig, eigenvectors = np.linalg.eig(T)

# Sort eigenvalues and eigenvectors in descending order
sorted_indices = np.argsort(eigenvalues_eig)[::-1]
eigenvalues_sorted = eigenvalues_eig[sorted_indices]
eigenvectors_sorted = eigenvectors[:, sorted_indices]

print("Principal stresses and directions using np.linalg.eig:")
for i in range(3):
    print(f"σ{i+1} = {eigenvalues_sorted[i]:.1f} MPa")
    n = eigenvectors_sorted[:, i]
    print(f"n{i+1} = [{n[0]:.3f}, {n[1]:.3f}, {n[2]:.3f}]")
    print()

# Plot the principal direction vectors in 3D
from mpl_toolkits.mplot3d import Axes3D

# Create two 3D plots with different viewing angles
fig1 = plt.figure(figsize=(12, 5))

# First view
ax1 = fig1.add_subplot(121, projection='3d')

colors = ['red', 'blue', 'green']
labels = ['n₁ (σ₁=150 MPa)', 'n₂ (σ₂=90 MPa)', 'n₃ (σ₃=60 MPa)']
```

```
for i in range(3):
    # Plot vector from origin to eigenvector
    x_vals = [0, eigenvectors_sorted[0, i]]
    y_vals = [0, eigenvectors_sorted[1, i]]
    z_vals = [0, eigenvectors_sorted[2, i]]

    ax1.plot(x_vals, y_vals, z_vals,
             color=colors[i], linewidth=3, marker='o',
             markersize=8, label=labels[i])

ax1.set_xlabel('x₁')
ax1.set_ylabel('x₂')
ax1.set_zlabel('x₃')
ax1.set_title('Principal Directions - View 1')
ax1.legend()
ax1.grid(True)

# Set equal aspect ratio
max_range = 1.0
ax1.set_xlim([-max_range, max_range])
ax1.set_ylim([-max_range, max_range])
ax1.set_zlim([-max_range, max_range])

# Second view with different angle
ax2 = fig1.add_subplot(122, projection='3d')

for i in range(3):
    x_vals = [0, eigenvectors_sorted[0, i]]
    y_vals = [0, eigenvectors_sorted[1, i]]
    z_vals = [0, eigenvectors_sorted[2, i]]

    ax2.plot(x_vals, y_vals, z_vals,
             color=colors[i], linewidth=3, marker='o',
             markersize=8, label=labels[i])

ax2.set_xlabel('x₁')
ax2.set_ylabel('x₂')
ax2.set_zlabel('x₃')
ax2.set_title('Principal Directions - View 2')
ax2.legend()
ax2.grid(True)
ax2.view_init(elev=30, azim=45)  # Different viewing angle

# Set equal aspect ratio
ax2.set_xlim([-max_range, max_range])
ax2.set_ylim([-max_range, max_range])
ax2.set_zlim([-max_range, max_range])

plt.tight_layout()
plt.show()

# Check orthogonality of eigenvectors
print("Verification of orthogonality:")
for i in range(3):
    for j in range(i+1, 3):
        dot_product = np.dot(eigenvectors_sorted[:, i],
```

```
                                eigenvectors_sorted[:, j])
            print(f"n{i+1} · n{j+1} = {dot_product:.6f}")

print("\nMagnitudes of eigenvectors:")
for i in range(3):
    magnitude = np.linalg.norm(eigenvectors_sorted[:, i])
    print(f"|n{i+1}| = {magnitude:.6f}")

# ============================================================================
# TASK 1.2.g) - Maximum shear stress analysis
# ============================================================================

print("\nTASK 1.2.g) - Maximum shear stress analysis")
print("-" * 60)

# Calculate maximum shear stress
sigma_1 = eigenvalues_sorted[0]  # Largest principal stress
sigma_3 = eigenvalues_sorted[2]  # Smallest principal stress
tau_max = (sigma_1 - sigma_3) / 2

print("Principal stresses:")
print(f"σ₁ = {sigma_1:.1f} MPa (largest)")
print(f"σ₂ = {eigenvalues_sorted[1]:.1f} MPa (intermediate)")
print(f"σ₃ = {sigma_3:.1f} MPa (smallest)")
print("\nMaximum shear stress:")
print(f"τ_max = (σ₁ - σ₃)/2 = ({sigma_1:.1f} - {sigma_3:.1f})/2 = "
      f"{tau_max:.1f} MPa")

# The maximum shear stress occurs in a plane normal to σ₂
# The normal vector to this plane is the eigenvector corresponding to σ₂
n2_direction = eigenvectors_sorted[:, 1]  # σ₂ direction (intermediate)

# The orientation of maximum shear stress can be found by considering
# vectors at 45° to both σ₁ and σ₃ directions in the plane normal to σ₂
n1_direction = eigenvectors_sorted[:, 0]  # σ₁ direction
n3_direction = eigenvectors_sorted[:, 2]  # σ₃ direction

# The maximum shear stress directions are at 45° between σ₁ and σ₃
# These can be computed as normalized combinations of n1 and n3
shear_direction_1 = ((n1_direction + n3_direction) /
                     np.linalg.norm(n1_direction + n3_direction))
shear_direction_2 = ((n1_direction - n3_direction) /
                     np.linalg.norm(n1_direction - n3_direction))

print("\nOrientation of maximum shear stress:")
print("The maximum shear stress occurs in the plane normal to σ₂ direction:")
n2 = n2_direction
print(f"Normal to σ₂ plane: [{n2[0]:.3f}, {n2[1]:.3f}, {n2[2]:.3f}]")
print("\nDirections of maximum shear stress (at 45° between σ₁ and σ₃):")
s1 = shear_direction_1
print(f"Shear direction 1: [{s1[0]:.3f}, {s1[1]:.3f}, {s1[2]:.3f}]")
s2 = shear_direction_2
print(f"Shear direction 2: [{s2[0]:.3f}, {s2[1]:.3f}, {s2[2]:.3f}]")

# Create 3D plot showing principal directions and maximum shear orientations
fig2 = plt.figure(figsize=(15, 6))
```

```
# First view
ax1 = fig2.add_subplot(121, projection='3d')

# Plot principal directions
colors = ['red', 'blue', 'green']
labels = ['n₁ (σ₁=150 MPa)', 'n₂ (σ₂=90 MPa)', 'n₃ (σ₃=60 MPa)']

for i in range(3):
    x_vals = [0, eigenvectors_sorted[0, i]]
    y_vals = [0, eigenvectors_sorted[1, i]]
    z_vals = [0, eigenvectors_sorted[2, i]]

    ax1.plot(x_vals, y_vals, z_vals,
             color=colors[i], linewidth=3, marker='o',
             markersize=8, label=labels[i])

# Plot maximum shear stress directions
shear_colors = ['orange', 'purple']
shear_labels = ['Max shear dir 1', 'Max shear dir 2']

for i, direction in enumerate([shear_direction_1, shear_direction_2]):
    x_vals = [0, direction[0]]
    y_vals = [0, direction[1]]
    z_vals = [0, direction[2]]

    ax1.plot(x_vals, y_vals, z_vals,
             color=shear_colors[i], linewidth=2, marker='s',
             markersize=6, linestyle='--', label=shear_labels[i])

ax1.set_xlabel('x₁')
ax1.set_ylabel('x₂')
ax1.set_zlabel('x₃')
ax1.set_title(f'Principal Directions and Max Shear Orientations\n'
              f'τ_max = {tau_max:.1f} MPa')
ax1.legend()
ax1.grid(True)

# Set equal aspect ratio
max_range = 1.0
ax1.set_xlim([-max_range, max_range])
ax1.set_ylim([-max_range, max_range])
ax1.set_zlim([-max_range, max_range])

# Second view with different angle
ax2 = fig2.add_subplot(122, projection='3d')

# Plot principal directions
for i in range(3):
    x_vals = [0, eigenvectors_sorted[0, i]]
    y_vals = [0, eigenvectors_sorted[1, i]]
    z_vals = [0, eigenvectors_sorted[2, i]]

    ax2.plot(x_vals, y_vals, z_vals,
             color=colors[i], linewidth=3, marker='o',
             markersize=8, label=labels[i])
```

```
# Plot maximum shear stress directions
for i, direction in enumerate([shear_direction_1, shear_direction_2]):
    x_vals = [0, direction[0]]
    y_vals = [0, direction[1]]
    z_vals = [0, direction[2]]

    ax2.plot(x_vals, y_vals, z_vals,
             color=shear_colors[i], linewidth=2, marker='s',
             markersize=6, linestyle='--', label=shear_labels[i])

ax2.set_xlabel('x₁')
ax2.set_ylabel('x₂')
ax2.set_zlabel('x₃')
ax2.set_title('Different Viewing Angle')
ax2.legend()
ax2.grid(True)
ax2.view_init(elev=20, azim=60)  # Different viewing angle

# Set equal aspect ratio
ax2.set_xlim([-max_range, max_range])
ax2.set_ylim([-max_range, max_range])
ax2.set_zlim([-max_range, max_range])

plt.tight_layout()
plt.show()

# ============================================================================
# VERIFICATION AND SUMMARY
# ============================================================================

print("\nVERIFICATION AND SUMMARY")
print("-" * 60)
print("Verification using Mohr's circle theory:")
print("In 3D, the maximum shear stress is: τ_max = (σ_max - σ_min)/2")
print("This occurs on planes that are 45° to the principal directions")
print(f"The calculated value τ_max = {tau_max:.1f} MPa is correct.")

print(f"\nSUMMARY OF RESULTS:")
print(f"- Principal stresses: σ₁={sigma_1:.1f}, σ₂={eigenvalues_sorted[1]:.1f}, "
      f"σ₃={sigma_3:.1f} MPa")
print(f"- Maximum shear stress: τ_max = {tau_max:.1f} MPa")
print(f"- All tasks completed successfully!")

print("\n" + "=" * 60)
print("END OF ANALYSIS")
print("=" * 60)
```