Innovative Pedagogical Tool:

Python-Based Problem Solving and Analysis of Principal Stresses using Generative AI.

Course: Mechanics of Deformable Solids

Faculty:

Syed Muhammed Fahd
Associate Professor

Innovative Pedagogical Tool: Python-Based Problem Solving and Analysis of Principal Stresses using Generative AI.

1. Background and Motivation

Understanding the **three-dimensional state of stress** and the concept of **principal stresses** is a cornerstone of the *Mechanics of Deformable Solids* course. Traditionally, this topic is taught using manual calculations as an eigen value problem and/or Mohr's circles, which often remain abstract for many students. Recognizing this gap, the present assignment was conceived as an **innovative pedagogical tool** that merges analytical mechanics with computational tools.

The activity directs students to solve a 3D stress problem using **Python programming** utilising Gen AI, wherein they compute the **principal stresses, directions, and maximum shear stress** by evaluating the **eigenvalues and eigenvectors** of the stress tensor. This bridges mathematical formulation with modern engineering computation.

The assignment aligns directly with **Course Outcome (CO1)** — "Determine the principal stresses using tensorial and graphical approach" — and is mapped to **PO1 (Engineering Knowledge)**, **PO2 (Problem Analysis)**, **PO5 (Modern Tool Usage)**, and **PO9 (Teamwork)**.

2. Explanation of the Innovative Pedagogical Activity

The idea was conceived to move beyond traditional problem-solving exercises and immerse students in an **applied, exploratory learning process**. The design process involved the following pedagogical intentions:

- **Bridging Theory and Computation:** Students were guided to translate the mathematical stress tensor formulation into Python code. This computational step compels them to understand the underlying structure of the stress components and how eigenvalue analysis yields principal stresses.
- **Encouraging Self-Learning and Exploration:** Instead of a step-by-step handout, students were provided with a **YouTube video tutorial** (prepared by the faculty) explaining the logic of generating Python code for stress analysis. This blended-learning component enabled asynchronous, self-paced learning and helped students visualize the problem computationally.
- **Collaborative Learning:** The task was assigned in **groups of three**, promoting peer-to-peer discussion in generation and debugging of code and interpretation of results.

3. Pedagogical Objectives and Learning Outcomes

This activity was not simply a coding exercise but a **concept-reinforcement activity** with multiple cognitive and skill-based outcomes:

Learning Domain	Expected Outcome
Cognitive (Understanding)	Students internalize how a 3D stress tensor is represented
Analytical (Application)	They apply eigenvalue concepts to determine principal stresses and directions.
Skill (Modern Tool Usage)	Students learn to use Python as a computational tool for numerical analysis in solid mechanics. Students also gets an exposure to use generative AI tools to create and debug code.
Collaborative (Soft Skills)	Students engage in team problem-solving, developing communication skills.

4. Reflection and Future Scope

The activity successfully promotes **computational thinking** within a traditional mechanics course and demonstrates how use of modern tools can enrich conceptual subjects.

Future iterations could include:

- Visualization of principal planes using 3D Python plotting (e.g., matplotlib).

Overall, this assignment exemplifies **innovative pedagogy through integration of computation, collaboration, and creativity**, aligning with the vision of NEP-2020 to promote *experiential and skill-based learning* in engineering education.

Mechanics of Deformable Solids

Assignment 1

CO1: Determine the principal stresses using tensorial and graphical approach.

Mapped POs: PO1(3), PO2(3), PO5(2) and PO9(3)

Question:

The state of stress at a point is given by $\sigma x=70$ MPa, $\sigma y=10$ MPa, $\sigma z=-20$ MPa, $\tau xy=-40$ MPa, $\tau yz=\tau xz=20$ MPa. Determine the principal stresses, direction of maximum principal stress and the value of maximum shear stress.

Use a python program to compute the principal stresses, direction of maximum principal stress and the value of maximum shear stress.

Hint: Principal stresses are the eigen values of stress tensor and direction of principal stress is the eigen vector. Max Shear stress is $\tau_{max}= (\sigma_{max}-\sigma_{min})/2$

This is a **group assignment**. Maximum number of students in a group **should not exceed 3**.

How to submit:

Assignment should be uploaded in ETLAB in a pdf format. It should contain the **names and roll numbers of all group members**. The **program and the results** in proper format. The document should be in **A4 size**.

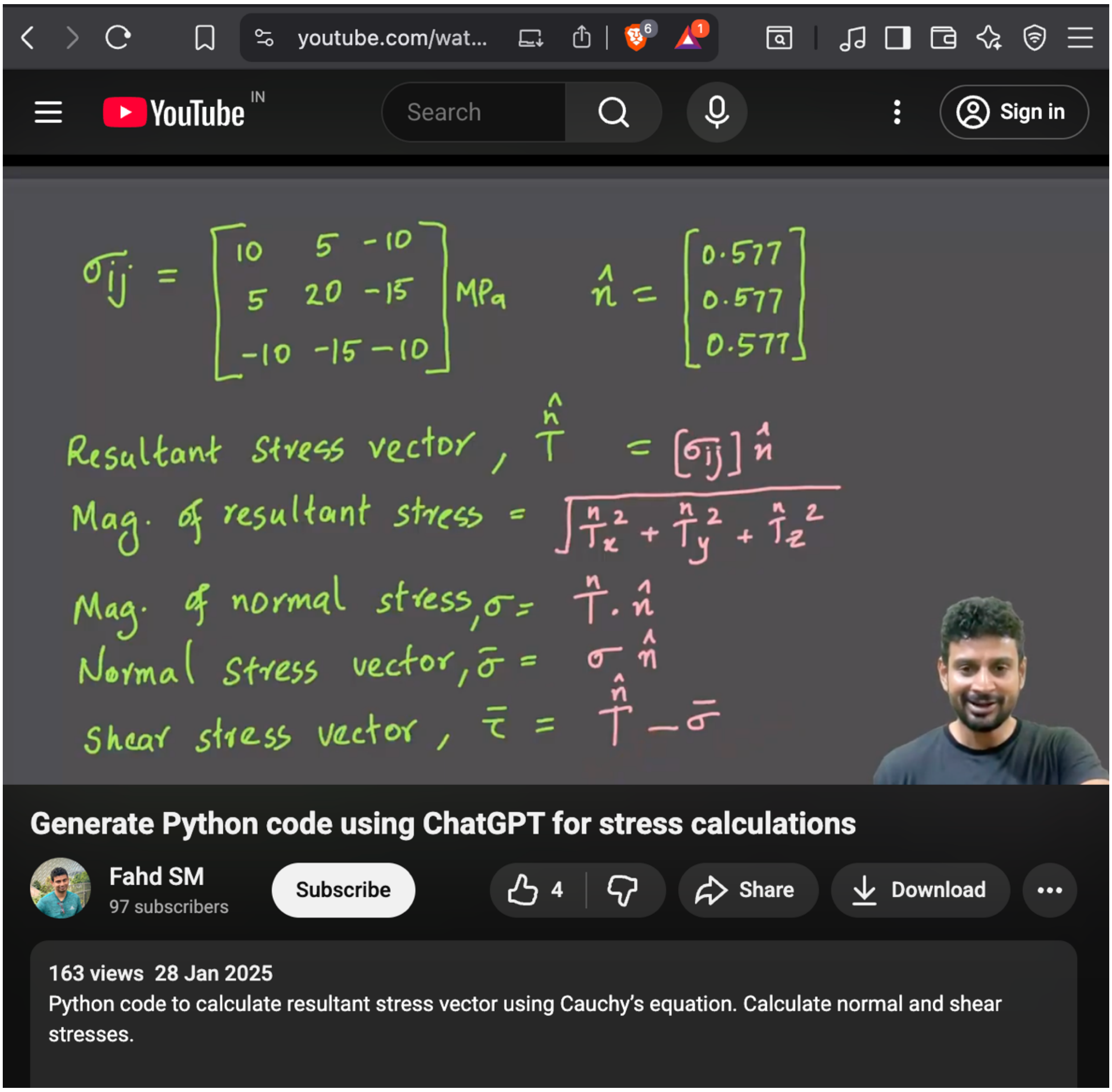
All students must upload the assignment document.

You may watch the following video on how to use python for stress calculations:

https://m.youtube.com/watch?v=9aiGXVP6gug

Video Link

https://www.youtube.com/watch?v=9aiGXVP6gug



EVALUATION GUIDELINES/RUBRICS

Criteria	Description	Weight (Marks)	Performance Indicators / Levels
1. Problem Formulation and Understanding	Understanding of the 3D state of stress, correct construction of the stress tensor, and clarity in defining the problem.	**3 Marks**	**3** – Clear understanding and correct formulation of stress tensor; identifies all components correctly. **2** – Minor conceptual or notation errors. **1** – Incomplete or partially correct formulation. **0** – Incorrect or missing formulation.
2. Code Accuracy and Logic	Correctness and efficiency of Python code, including matrix representation, eigenvalue–eigenvector computation, and sorting logic.	**4 Marks**	**4** – Code fully correct, logically structured, and produces correct outputs without errors. **3** – Minor syntax or formatting errors not affecting results. **2** – Partial implementation or logical flaws. **1** – Major conceptual/syntax errors. **0** – Non-functional or missing code.
3. Results and Presentation	Accuracy of computed principal stresses, directions, and maximum shear stress; clarity of output formatting and documentation.	**3 Marks**	**3** – All results accurate, clearly presented with units and explanations. **2** – Minor computational or presentation errors. **1** – Results incomplete or with significant errors. **0** – No or incorrect results.

Sample Answers

Mechanics of Deformable Solids
Assignment 1

Group Members : Rana Rishan, Akhil Alex, Karthik K
Roll Numbers : B23MEC58, B23MEC12,B23MEC35.

Question : The state of stress at a point is given by σx=70 MPa, σy=10 MPa, σz=−20 MPa, τxy=−40 MPa, τyz=τxz=20 MPa. Determine the principal stresses, direction of maximum principal stress and the value of maximum shear stress. Use a python program to compute the principal stresses, direction of maximum principal stress and the value of maximum shear
stress.

Code :

```
import numpy as np

# Given stress components
sigma_x = 70  # MPa
sigma_y = 10  # MPa
sigma_z = -20 # MPa
tau_xy = -40  # MPa
tau_yz = 20   # MPa
tau_xz = 20   # MPa

# Stress tensor
stress_tensor = np.array([[sigma_x, tau_xy, tau_xz],
                          [tau_xy, sigma_y, tau_yz],
                          [tau_xz, tau_yz, sigma_z]])

# Compute principal stresses (eigenvalues) and eigenvectors
principal_stresses, principal_directions = np.linalg.eig(stress_tensor)

# Sort eigenvalues and corresponding eigenvectors
sorted_indices = np.argsort(principal_stresses)[::-1]  # Sort in descending order
principal_stresses = principal_stresses[sorted_indices]
principal_directions = principal_directions[:, sorted_indices]

# Maximum shear stress
max_shear_stress = (principal_stresses[0] - principal_stresses[2]) / 2

# Display results
print("Principal Stresses (MPa):")
print(f"σ1 = {principal_stresses[0]:.2f}, σ2 = {principal_stresses[1]:.2f}, σ3 =
{principal_stresses[2]:.2f}")
```

```
print("\nDirection Cosines of Maximum Principal Stress (σ1):")
print(f"l = {principal_directions[0, 0]:.3f}, m = {principal_directions[1, 0]:.3f}, n =
{principal_directions[2, 0]:.3f}")

print(f"\nMaximum Shear Stress: {max_shear_stress:.2f} MPa")
```

<u>Output</u> :

Principal Stresses (MPa):
$\sigma 1$ = 90.77, $\sigma 2$ = 11.88, $\sigma 3$ = -42.65

Direction Cosines of Maximum Principal Stress ($\sigma 1$):
l = -0.901, m = 0.425, n = -0.086

Maximum Shear Stress: 66.71 MPa

Mechanics Of Deformable Solids
23MET401
Assignment no: 1

Group Members:

Avinash T S - B23MEC20

Aswaghosh A S - B23MEC19

Anjith K A - B23MEC16

Question:

The state of stress at a point is given by $\sigma_x=70$ MPa, $\sigma_y=10$ MPa, $\sigma_z=-20$ MPa, $\tau_{xy}=-40$ MPa, $\tau_{yz}=\tau_{xz}=20$ MPa. Determine the principal stresses, direction of maximum principal stress and the value of maximum shear stress. Use a python program to compute the principal stresses, direction of maximum principal stress and the value of maximum shear stress.

Hint: Principal stresses are the eigen values of stress tensor and direction of principal stress is the eigen vector. Max Shear stress is $\tau_{max}=(\sigma_{max}-\sigma_{min})/2$

```
import numpy as np

# Given stress components (in MPa)
σx  =  70
σy  =  10
σz  = -20
τxy = -40
τyz =  20
τxz =  20

# Stress tensor
stress_tensor = np.array([[σx, τxy, τxz],
                          [τxy, σy, τyz],
                          [τxz, τyz, σz]])

# Compute principal stresses (eigenvalues) and directions (eigenvectors)
principal_stresses, principal_directions = np.linalg.eig(stress_tensor)

# Sort the principal stresses and corresponding eigenvectors
sorted_indices = np.argsort(principal_stresses)[::-1]  # Descending order
principal_stresses = principal_stresses[sorted_indices]
principal_directions = principal_directions[:, sorted_indices]

# Maximum shear stress
max_shear_stress = (max(principal_stresses) - min(principal_stresses)) / 2

#Round of the results to 3 decimal places
principal_stresses = np.round(principal_stresses, 3)
principal_directions = np.round(principal_directions, 3)
max_shear_stress = np.round(max_shear_stress, 3)

# Print results
print("Principal Stresses:")
print(f"σ1 = {principal_stresses[0]:.3f} MPa")
print(f"σ2 = {principal_stresses[1]:.3f} MPa")
print(f"σ3 = {principal_stresses[2]:.3f} MPa")

print("\nDirection of Maximum Principal Stress (σ1):")
print(f"nx = {principal_directions[0, 0]:.3f}")
print(f"ny = {principal_directions[1, 0]:.3f}")
print(f"nz = {principal_directions[2, 0]:.3f}")

print(f"\nMaximum Shear Stress,τmax = {max_shear_stress:.2f} MPa")
```

```
Principal Stresses:
σ1 = 90.772 MPa
σ2 = 11.881 MPa
σ3 = -42.653 MPa

Direction of Maximum Principal Stress (σ1):
nx = -0.901
ny = 0.425
nz = -0.086

Maximum Shear Stress,τmax = 66.71 MPa
```