Introduction to Mechanics: Stress, Strain, and Thermal Effects

Yazhuo Liu

October 15, 2024

Contents

1 Stress and Strain: Definitions and Importance

1.1 Stress (σ)

- **Definition**: Stress is the internal force experienced by a material per unit area due to an external load.
- **Formula**:

$$\sigma = \frac{F}{A}$$

  - F = Force applied (in Newtons, N)
  - A = Cross-sectional area (in square meters, m^2)

Why It's Defined: Stress helps us understand how materials react internally to external forces. It tells us whether a material will withstand the load or fail.

1.2 Strain (ε)

- **Definition**: Strain is the measure of deformation representing the displacement between particles in the material body.
- **Formula**:

$$\varepsilon = \frac{\Delta L}{L_0}$$

  - ΔL = Change in length (in meters, m)
  - L_0 = Original length (in meters, m)

Why It's Defined: Strain quantifies how much a material deforms under stress, essential for predicting material behavior.

1.3 Thermal Strain (ε_t)

Thermal strain is the deformation per unit length resulting from a temperature change.

$$\varepsilon_t = \alpha \Delta T$$

- α = Coefficient of thermal expansion (per °C)
- ΔT = Change in temperature (°C)

1.3.1 Unconstrained Expansion

- If a material is free to expand, thermal strain does not produce stress.
- The material uniformly expands or contracts without internal resistance.

1.3.2 Constrained Expansion

- If the material is restrained (e.g., fixed at both ends), thermal expansion generates internal stresses.
- The restraint causes a mechanical strain opposite to the thermal strain to satisfy compatibility.

Real-World Example: Expansion joints in bridges accommodate thermal expansion to prevent structural damage.

2 Derivation of Equilibrium Equations

2.1 Fundamental Principle

- **Newton's Second Law**: The rate of change of momentum of a body is equal to the net external force acting on it. For statics (no acceleration), the sum of forces is zero.
- **Static Equilibrium**: Sum of all forces and moments acting on a body equals zero.

2.2 Derivation in One Dimension

Consider a small element of material with length dx, subjected to stress σ that varies along x.

2.2.1 Step 1: Consider a Differential Element

- Length of the element: dx
- Stress on the left face: $\sigma(x)$
- Stress on the right face: $\sigma(x + dx)$

2.2.2 Step 2: Express Stress on the Right Face Using Taylor Series

Expand $\sigma(x + dx)$ around x:

$$\sigma(x + dx) \approx \sigma(x) + \frac{d\sigma(x)}{dx}dx$$

2.2.3 Step 3: Calculate the Net Force on the Element

- Force on the left face: $\sigma(x)A$
- Force on the right face: $\left(\sigma(x) + \frac{d\sigma}{dx}dx\right)A$
- Net force (dF):

$$dF = \left(\sigma + \frac{d\sigma}{dx}dx\right)A - \sigma A = \frac{d\sigma}{dx}Adx$$

2.2.4 Step 4: Apply Equilibrium Condition

For static equilibrium, net force is zero:

$$dF = 0 \implies \frac{d\sigma}{dx} A dx = 0$$

Since $Adx \neq 0$, we have:

$$\frac{d\sigma}{dx} = 0$$

Conclusion: Stress is constant along the length if there are no body forces.

2.3 Derivation in Three Dimensions

2.3.1 Step 1: Consider an Infinitesimal Cube Element

- Dimensions: dx, dy, dz
- Stress components:
  - Normal stresses: $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}$
  - Shear stresses: $\tau_{xy}, \tau_{xz}, \tau_{yz}$, etc.

2.3.2 Step 2: Write Equilibrium Equations for Each Direction

Along x-Direction Sum of forces in x-direction equals zero:

$$\sum F_x = 0$$

Forces due to stresses on faces perpendicular to x:

- At x: $\sigma_{xx}(x) \cdot dy \cdot dz$
- At $x + dx$: $\left[\sigma_{xx} + \frac{\partial \sigma_{xx}}{\partial x} dx\right] \cdot dy \cdot dz$

Forces due to shear stresses on faces perpendicular to y and z:

- Shear stress τ_{yx} on face at y: $\tau_{yx}(y) \cdot dx \cdot dz$
- Shear stress τ_{zx} on face at z: $\tau_{zx}(z) \cdot dx \cdot dy$

Net force in x-direction:

$$dF_x = \left(\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \right) dxdydz$$

Ignore the body force and apply equilibrium condition For static equilibrium:

$$dF_x = 0 \implies \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} = 0$$

Similarly for y and z directions:

$$\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + Y = 0$$

$$\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + Z = 0$$

Conclusion: These are the equilibrium equations in three dimensions.

$$\nabla \cdot \boldsymbol{\sigma} = \mathbf{0}$$

3 Derivation of Hooke's Law

3.1 Hooke's Law for Linear Elastic Materials

Hooke's Law describes the linear relationship between stress and strain in an elastic material. In three dimensions, this relationship accounts for stresses and strains in all three normal directions and the three shear directions, incorporating Poisson's effect.

3.2 Understanding Poisson's Effect

- **Poisson's Ratio (ν)**: It quantifies the negative ratio of transverse to axial strain. When a material is stretched in one direction, it tends to contract in the perpendicular directions.

$$\nu = -\frac{\varepsilon_{\text{transverse}}}{\varepsilon_{\text{axial}}}$$

3.3 Stress and Strain Tensors

In three dimensions, stress and strain are represented as second-order tensors.

3.3.1 Stress Tensor (σ_{ij})

$$\sigma_{ij} = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix}$$

- $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}$: Normal stresses.
- $\tau_{xy}, \tau_{xz}, \tau_{yz}$: Shear stresses.

3.3.2 Strain Tensor (ε_{ij})

$$\varepsilon_{ij} = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz} \end{bmatrix}$$

- $\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{zz}$: Normal strains.
- $\varepsilon_{xy}, \varepsilon_{xz}, \varepsilon_{yz}$: Shear strains.

3.4 Derivation of Hooke's Law for Normal Stresses and Strains

Consider an isotropic, linear elastic material subjected to normal stresses in all three directions.

3.4.1 Step 1: Axial Strain Due to Axial Stress

In the absence of other stresses, the axial strain in the x-direction due to σ_{xx} is:

$$\varepsilon_{xx} = \frac{\sigma_{xx}}{E}$$

3.4.2 Step 2: Transverse Strains Due to Poisson's Effect

However, due to Poisson's effect, applying σ_{xx} also causes transverse strains:

$$\varepsilon_{yy} = \varepsilon_{zz} = -\nu \frac{\sigma_{xx}}{E}$$

Similarly, stresses σ_{yy} and σ_{zz} cause transverse strains in other directions.

3.4.3 Step 3: Superposition of Effects

The total strain in each direction is the sum of strains due to all three normal stresses:

- **Strain in x-Direction:**
$$\varepsilon_{xx} = \frac{1}{E}\left[\sigma_{xx} - \nu(\sigma_{yy} + \sigma_{zz})\right]$$
- **Strain in y-Direction:**
$$\varepsilon_{yy} = \frac{1}{E}\left[\sigma_{yy} - \nu(\sigma_{xx} + \sigma_{zz})\right]$$
- **Strain in z-Direction:**
$$\varepsilon_{zz} = \frac{1}{E}\left[\sigma_{zz} - \nu(\sigma_{xx} + \sigma_{yy})\right]$$

Explanation:

- The strain in one direction is caused by the direct stress in that direction and the Poisson's effect from stresses in the other two directions.
- Negative signs in Poisson's effect indicate contraction in the transverse directions when the axial stress is tensile.

3.4.4 Step 4: Rearranging Equations

Multiply both sides by E to simplify:

$$E\varepsilon_{xx} = \sigma_{xx} - \nu(\sigma_{yy} + \sigma_{zz})$$

Similarly for ε_{yy} and ε_{zz}.

3.4.5 Step 5: Expressing Stresses in Terms of Strains

Group the terms to express σ_{xx}, σ_{yy}, and σ_{zz} in terms of strains:

- **Express σ_{xx}:**

  $$\sigma_{xx} = E\varepsilon_{xx} + \nu(\sigma_{yy} + \sigma_{zz})$$

  But σ_{yy} and σ_{zz} themselves depend on strains in a similar way. To solve for σ_{xx}, we can consider the system of equations for all three normal stresses.

3.4.6 Step 6: Solving the System of Equations

Set up the equations:

$$\begin{cases} E\varepsilon_{xx} = \sigma_{xx} - \nu(\sigma_{yy} + \sigma_{zz}) \\ E\varepsilon_{yy} = \sigma_{yy} - \nu(\sigma_{xx} + \sigma_{zz}) \\ E\varepsilon_{zz} = \sigma_{zz} - \nu(\sigma_{xx} + \sigma_{yy}) \end{cases}$$

Sum all three equations:

$$E(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}) = (\sigma_{xx} + \sigma_{yy} + \sigma_{zz}) - 2\nu(\sigma_{xx} + \sigma_{yy} + \sigma_{zz})$$

Simplify:

$$E(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}) = (1 - 2\nu)(\sigma_{xx} + \sigma_{yy} + \sigma_{zz})$$

Solve for $\sigma_{xx} + \sigma_{yy} + \sigma_{zz}$:

$$\sigma_{xx} + \sigma_{yy} + \sigma_{zz} = \frac{E}{1 - 2\nu}(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz})$$

3.4.7 Step 7: Substitute Back to Find Individual Stresses

Now, substitute $\sigma_{yy} + \sigma_{zz}$ into the equation for σ_{xx}:

$$E\varepsilon_{xx} = \sigma_{xx} - \nu\left(\frac{E}{1 - 2\nu}(\varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz})\right)$$

Rearrange and solve for σ_{xx}:

$$\sigma_{xx} = \frac{E}{(1 + \nu)(1 - 2\nu)}\left[(1 - \nu)\varepsilon_{xx} + \nu(\varepsilon_{yy} + \varepsilon_{zz})\right]$$

Similarly, for σ_{yy} and σ_{zz}:

$$\sigma_{yy} = \frac{E}{(1 + \nu)(1 - 2\nu)}\left[(1 - \nu)\varepsilon_{yy} + \nu(\varepsilon_{xx} + \varepsilon_{zz})\right]$$

$$\sigma_{zz} = \frac{E}{(1 + \nu)(1 - 2\nu)}\left[(1 - \nu)\varepsilon_{zz} + \nu(\varepsilon_{xx} + \varepsilon_{yy})\right]$$

3.5 Derivation of Hooke's Law for Shear Stresses and Strains

For shear components, Hooke's Law states that shear stress is directly proportional to shear strain:

- **Shear Modulus (G)**:
$$G = \frac{E}{2(1+\nu)}$$

- **Shear Stress-Strain Relationships**:

$$\begin{cases} \tau_{xy} = G\gamma_{xy} \\ \tau_{yz} = G\gamma_{yz} \\ \tau_{xz} = G\gamma_{xz} \end{cases}$$

Note: γ_{xy}, γ_{yz}, and γ_{xz} are engineering shear strains (twice the tensorial shear strain components).

3.6 Combining Normal and Shear Components

We can now combine both normal and shear components into a single constitutive relation.

3.6.1 General Constitutive Equations

- **For Normal Stresses**:
$$\sigma_{ij} = \lambda\varepsilon_{kk} + 2G\varepsilon_{ij} \quad \text{for} \quad i = j$$

- **For Shear Stresses**:
$$\sigma_{ij} = 2G\varepsilon_{ij} \quad \text{for} \quad i \neq j$$

Lamé's First Parameter (λ):

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$$

3.6.2 Simplifying the Constitutive Equations

For both normal and shear components, the general form can be written as:

$$\sigma_{ij} = 2G\varepsilon_{ij} + \lambda\delta_{ij}\varepsilon_{kk}$$

- $\varepsilon_{kk} = \varepsilon_{xx} + \varepsilon_{yy} + \varepsilon_{zz}$ is the trace of the strain tensor (volumetric strain).
- δ_{ij} is the Kronecker delta ($\delta_{ij} = 1$ if $i = j$, $\delta_{ij} = 0$ if $i \neq j$).

3.7 Incorporating Thermal Strain

When thermal strains are present due to temperature changes, they must be included in the total strain.

3.7.1 Total Strain

$$\varepsilon_{ij} = \varepsilon_{ij}^{\text{mechanical}} + \varepsilon_{ij}^{\text{thermal}}$$

3.7.2 Thermal Strain for Isotropic Materials

$$\varepsilon_{ij}^{\text{thermal}} = \alpha \Delta T \delta_{ij}$$

- α is the coefficient of thermal expansion.
- ΔT is the temperature change.

3.7.3 Modified Hooke's Law Including Thermal Strain

Only mechanical strain can lead to stresses, therefore,

$$\sigma_{ij} = 2G\left(\varepsilon_{ij} - \varepsilon_{ij}^{\text{thermal}}\right) + \lambda \delta_{ij}\left(\varepsilon_{kk} - \varepsilon_{kk}^{\text{thermal}}\right)$$

4 Stresses from Non-uniform Temperature Distribution (1D)

4.1 Non-uniform Temperature Fields

- **Definition**: Temperature varies spatially within the material.
- **Effect**: Causes differential expansion/contraction leading to internal stresses even if the material is unconstrained externally.

4.2 Derivation of Thermal Stresses

4.2.1 Step 1: Total Strain

Total strain (ε) is the sum of mechanical strain (ε_m) and thermal strain (ε_t):

$$\varepsilon = \varepsilon_m + \varepsilon_t$$

4.2.2 Step 2: Constitutive Equation Including Thermal Effects

From Hooke's Law with thermal strain:

$$\sigma = E\varepsilon_m = E(\varepsilon - \varepsilon_t)$$

4.2.3 Step 3: Equilibrium and Compatibility Conditions

- **Equilibrium**: The internal stresses must satisfy the equilibrium equations derived earlier.
- **Compatibility**: The strains must be compatible; the material must deform continuously without gaps or overlaps.

4.2.4 Step 4: Apply to Non-uniform Temperature

Let the temperature T be a function of position x: $T = T(x)$

Thermal strain becomes:

$$\varepsilon_t(x) = \alpha[T(x) - T_0]$$

- T_0 = Reference temperature (usually the initial temperature)

4.2.5 Step 5: Derive the Stress Distribution

The mechanical strain is related to displacement gradient:

$$\varepsilon_m(x) = \frac{du}{dx}$$

Substituting into the constitutive equation:

$$\sigma(x) = 2G\left(\frac{du}{dx} - \alpha[T(x) - T_0]\right) + \lambda\delta_{ij}\left(\frac{du}{dx} - \alpha[T(x) - T_0]\right)$$

Simplifying for isotropic materials and assuming plane stress conditions:

$$\sigma(x) = E\left(\frac{du}{dx} - \alpha[T(x) - T_0]\right)$$

4.2.6 Step 6: Apply Equilibrium Equation

From equilibrium:

$$\frac{d\sigma}{dx} = 0$$

Substitute $\sigma(x)$:

$$\frac{d}{dx}\left[E\left(\frac{du}{dx} - \alpha[T(x) - T_0]\right)\right] = 0$$

Simplify (assuming E and α are constants):

$$E\left(\frac{d^2u}{dx^2} - \alpha\frac{dT}{dx}\right) = 0$$

Rearranged:

$$\frac{d^2u}{dx^2} = \alpha\frac{dT}{dx}$$

4.2.7 Step 7: Integrate to Find Displacement

Integrate once:

$$\frac{du}{dx} = \alpha[T(x) - T_0] + C_1$$

Integrate again:

$$u(x) = \alpha\int_{x_0}^{x}[T(x') - T_0]dx' + C_1x + C_2$$

- C_1 and C_2 are constants determined by boundary conditions.

4.2.8 Step 8: Determine Stress

Substitute back into $\sigma(x)$:

$$\sigma(x) = E\left(\frac{du}{dx} - \alpha[T(x) - T_0]\right) = EC_1$$

Conclusion: The stress $\sigma(x)$ is constant if no external forces or constraints are applied, but the displacement varies due to the non-uniform temperature field.

Implications: In real structures, constraints often exist, leading to non-zero stress distributions due to temperature gradients.

5 FEniCS Example: Solving Stresses in a Heated 3D Block

5.1 Problem Statement

- **Objective**: Compute the stress distribution in a 3D rectangular block with a non-uniform temperature distribution.
- **Assumptions**:
  - Linear elastic, isotropic material.
  - Thickness is small compared to other dimensions.
  - 2 fixed boundaries (no displacement).
  - Temperature varies along one axis (e.g., temperature increases along the x-axis).

5.2 Step-by-Step Code with Line-by-Line Explanation

5.2.1 Import Necessary Libraries

```
from fenics import *             # Import FEniCS library
import numpy as np               # Import NumPy for numerical operations
import matplotlib.pyplot as plt # Import Matplotlib for plotting results
from mpl_toolkits.mplot3d import Axes3D
```

Listing 1: Import Necessary Libraries

Explanation: We import the required libraries for finite element analysis, numerical computations, and visualization.

5.2.2 Define Material Properties

```
E = 210e9           # Young's Modulus in Pascals (e.g., for steel)
nu = 0.3            # Poisson's Ratio (typical for steel)
alpha = 12.0e-6      # Coefficient of thermal expansion per degree Celsius
```

Listing 2: Define Material Properties

Explanation: We set the material properties. These constants define how the material responds to stress and temperature changes.

5.2.3 Compute Lamé Parameters

```python
# Compute Lame's first parameter (lambda) and shear modulus (mu)
mu = E / (2 * (1 + nu))         # Shear modulus
lmbda = E * nu / ((1 + nu) * (1 - 2 * nu))  # Lame's first parameter
```

Listing 3: Compute Lamé Parameters

Explanation: These parameters are used in the constitutive equations relating stress and strain in three dimensions.

5.2.4 Create Mesh and Define Function Space

```python
# Define geometry dimensions
length = 1.0       # Length of the block in meters
height = 0.1       # Height of the block in meters
depth = 0.1        # Depth of the block in meters

# Create a 3D rectangular mesh
mesh = BoxMesh(Point(0, 0, 0), Point(length, height, depth), 20, 5, 5)

# Define function space for vector-valued functions (displacement field)
V = VectorFunctionSpace(mesh, 'P', 1, dim=3)  # Linear Lagrange elements
```

Listing 4: Create Mesh and Define Function Space

Explanation:

- We define the geometry of the block and create a 3D mesh that discretizes the domain.
- The function space V is where the displacement solution will live.

5.2.5 Define Boundary Conditions

```python
# Define boundary condition: Fix the left face (x = 0)
def left_boundary(x, on_boundary):
    return near(x[0], 0.0) and on_boundary

# Define boundary condition: Fix the bottom face (y = 0) to prevent rigid body motion
def bottom_boundary(x, on_boundary):
    return near(x[1], 0.0) and on_boundary

# Apply zero displacement on the left boundary and fix z-displacement on the bottom
bc_left = DirichletBC(V, Constant((0.0, 0.0, 0.0)), left_boundary)
bc_bottom = DirichletBC(V.sub(2), Constant(0.0), bottom_boundary)  # Fix z-displacement on bottom

# Combine boundary conditions
bcs = [bc_left, bc_bottom]
```

Listing 5: Define Boundary Conditions

Explanation:

- We define functions to identify points on the left and bottom boundaries.

- We apply Dirichlet boundary conditions (fixed displacement) on these boundaries to prevent rigid body motions.

5.2.6 Define Non-uniform Temperature Distribution

```
# Reference and maximum temperatures
T0 = 20.0      # Reference temperature in Celsius
T1 = 100.0     # Maximum temperature in Celsius

# Define temperature distribution as an expression varying along the x-axis
temperature = Expression('T0 + (T1 - T0) * x[0] / L',
                         T0=T0, T1=T1, L=length, degree=1)
```

Listing 6: Define Non-uniform Temperature Distribution

Explanation:

- We define a linear temperature gradient from T_0 at $x = 0$ to T_1 at $x = L$.
- The ‘Expression‘ allows the temperature to vary spatially.

5.2.7 Define Strain and Stress Expressions

```
# Define the strain tensor (symmetric gradient of displacement)
def epsilon(u):
    return sym(grad(u))  # Returns symmetric part of the gradient

# Define the thermal strain tensor
def epsilon_t():
    return alpha * (temperature - T0) * Identity(3)  # 3D identity matrix

# Define the stress tensor using Hooke's Law with thermal strain
def sigma(u):
    return lmbda * tr(epsilon(u) - epsilon_t()) * Identity(3) + 2 * mu * (
   epsilon(u) - epsilon_t())
```

Listing 7: Define Strain and Stress Expressions

Explanation:

- ‘epsilon(u)‘ computes the strain tensor from the displacement field ‘u‘.
- ‘epsilon_t()‘ computes the thermal strain due to temperature change.
- ‘sigma(u)‘ computes the stress tensor, considering both mechanical and thermal strains.

5.2.8 Define Variational Problem

```
# Define trial and test functions
u = TrialFunction(V)  # Displacement field to solve for
v = TestFunction(V)   # Test function for variational formulation

# Define the variational form (weak form of the equilibrium equations)
F = inner(sigma(u), epsilon(v)) * dx
```

Listing 8: Define Variational Problem

5.2.9 Compute the Solution

```
# Solve the variational problem
u_sol = Function(V)

problem = LinearVariationalProblem(lhs(F), rhs(F), u_sol, bcs)
solver = LinearVariationalSolver(problem)
solver.solve()
```

Listing 9: Compute the Solution

Explanation:

- We initialize ‘u_sol‘ as a ‘Function‘ in the space ‘V‘ to hold the solution.
- Then solves the variational equation ‘F == 0‘ subject to boundary conditions ‘bcs‘.

5.2.10 Post-processing and Visualization

```
# Compute stress tensor at each point
S = sigma(u_sol)          # Stress tensor as a function

# Compute von Mises stress for visualization
V_von = FunctionSpace(mesh, 'P', 1)  # Scalar function space
von_Mises = Function(V_von)          # Function to store von Mises stress

# Expression for von Mises stress in 3D
von_Mises_expr = sqrt(0.5 * ((S[0,0] - S[1,1])**2 +
                             (S[1,1] - S[2,2])**2 +
                             (S[2,2] - S[0,0])**2 +
                             6*(S[0,1]**2 + S[1,2]**2 + S[0,2]**2)))

# Project von Mises stress to function space for plotting
von_Mises = project(von_Mises_expr, V_von)

# Plot results
stress_array = von_Mises.compute_vertex_values(mesh)
coordinates = mesh.coordinates()
x = coordinates[:,0]
y = coordinates[:,1]
z = coordinates[:,2]

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
p = ax.scatter(x, y, z, c=stress_array, cmap='viridis', marker='o')
fig.colorbar(p, ax=ax, label='Von Mises Stress (Pa)')
ax.set_title('Von Mises Stress Distribution in 3D Block')
ax.set_xlabel('X (m)')
ax.set_ylabel('Y (m)')
ax.set_zlabel('Z (m)')
plt.show()
```

Listing 10: Post-processing and Visualization

Explanation:

- We calculate the von Mises stress, which is a scalar value representing the yield criterion in ductile materials.

- The ‘project‘ function interpolates the expression into the finite element space for visualization.
- We use Matplotlib to create a 3D scatter plot of the von Mises stress distribution.

5.3 Full Code Listing

Below is the complete code combining all steps for the 3D FEniCS example:

```
from fenics import *
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Material properties
E = 210e9           # Young's Modulus (Pa)
nu = 0.3            # Poisson's Ratio
alpha = 1.2e-5      # Thermal expansion coefficient (/K)

# Lame parameters
mu = E / (2 * (1 + nu))
lmbda = E * nu / ((1 + nu)*(1 - 2*nu))

# Geometry parameters
length = 1.0        # Length of the block (m)
height = 0.1        # Height of the block (m)
depth = 0.1         # Depth of the block (m)

# Mesh
mesh = BoxMesh(Point(0, 0, 0), Point(length, height, depth), 20, 5, 5)

# Function space
V = VectorFunctionSpace(mesh, 'P', 1, dim=3)

# Boundary conditions
def left_boundary(x, on_boundary):
    return near(x[0], 0.0) and on_boundary

def bottom_boundary(x, on_boundary):
    return near(x[1], 0.0) and on_boundary

# Apply zero displacement on the left boundary and fix z-displacement on the bottom
bc_left = DirichletBC(V, Constant((0.0, 0.0, 0.0)), left_boundary)
bc_bottom = DirichletBC(V.sub(2), Constant(0.0), bottom_boundary)  # Fix z-displacement on bottom

bcs = [bc_left, bc_bottom]

# Temperature distribution
T0 = 20.0           # Reference temperature (C)
T1 = 100.0          # Maximum temperature (C)

temperature = Expression('T0 + (T1 - T0) * x[0] / L',
                         T0=T0, T1=T1, L=length, degree=1)

```

```python
# Define the strain tensor (symmetric gradient of displacement)
def epsilon(u):
    return sym(grad(u))

# Define the thermal strain tensor using as_tensor for correct shape
def epsilon_t():
    return alpha * (temperature - T0) * as_tensor(((1, 0, 0),
                                                   (0, 1, 0),
                                                   (0, 0, 1)))

# Define the stress tensor using Hooke's Law with thermal strain
def sigma(u):
    return lmbda * tr(epsilon(u) - epsilon_t()) * Identity(3) + 2 * mu * (
    epsilon(u) - epsilon_t())

# Variational problem
u = TrialFunction(V)
v = TestFunction(V)

# weak form
F = inner(sigma(u), epsilon(v)) * dx

# Assemble and solve the system
u_sol = Function(V)

problem = LinearVariationalProblem(lhs(F), rhs(F), u_sol, bcs)
solver = LinearVariationalSolver(problem)
solver.solve()


# Compute stress tensor at each point
S = sigma(u_sol)

# Compute von Mises stress for visualization
V_von = FunctionSpace(mesh, 'P', 1)  # Scalar function space
von_Mises = Function(V_von)          # Function to store von Mises stress

# Expression for von Mises stress in 3D
von_Mises_expr = sqrt(0.5 * ((S[0,0] - S[1,1])**2 +
                             (S[1,1] - S[2,2])**2 +
                             (S[2,2] - S[0,0])**2 +
                             6*(S[0,1]**2 + S[1,2]**2 + S[0,2]**2)))

# Project von Mises stress to function space for plotting
von_Mises = project(von_Mises_expr, V_von)

# Plot results
stress_array = von_Mises.compute_vertex_values(mesh)
coordinates = mesh.coordinates()
x = coordinates[:,0]
y = coordinates[:,1]
z = coordinates[:,2]

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
p = ax.scatter(x, y, z, c=stress_array, cmap='viridis', marker='o')
```

```python
fig.colorbar(p, ax=ax, label='Von Mises Stress (Pa)')
ax.set_title('Von Mises Stress Distribution in 3D Block')
ax.set_xlabel('X (m)')
ax.set_ylabel('Y (m)')
ax.set_zlabel('Z (m)')
plt.show()
```

Listing 11: Full 3D FEniCS Code

5.4 Explanation of the Results

- **Displacement Field** (u): Shows how the block deforms due to thermal expansion. Since the left face is fixed, displacement primarily occurs on the opposite face where temperature is higher.
- **Stress Distribution** (σ): Reveals areas with higher stress, particularly near the fixed boundaries where thermal expansion is most constrained.
- **Von Mises Stress**: Provides a scalar measure of the stress state, useful for assessing yield according to the von Mises criterion.

Observation: The maximum stress occurs near the fixed boundaries (left face and bottom face), where thermal expansion is most constrained. The stress gradually decreases towards the free faces where temperature effects are less pronounced.

Visualization: The 3D scatter plot colored by von Mises stress allows for easy identification of high-stress regions within the block.

6 Conclusion

Understanding stress, strain, and thermal effects is crucial in mechanics to predict material behavior under various conditions. By integrating detailed theoretical derivations with computational tools like FEniCS, we can simulate complex problems such as stress distributions due to non-uniform temperature fields in three-dimensional structures. This approach aids in better design and analysis of engineering components, ensuring safety and reliability in real-world applications.

References

- Gere, J. M., & Goodno, B. J. (2012). *Mechanics of Materials*. Cengage Learning.
- Logg, A., Mardal, K.-A., & Wells, G. N. (Eds.). (2012). *Automated Solution of Differential Equations by the Finite Element Method*. Springer.
- FEniCS Project Documentation: https://fenicsproject.org/documentation/