Technische Universität Bergakademie Freiberg

Computational Materials Science
Fakultät IV

Personal Programming Project

Nonlinear Finite element implementation of thermoelastic Material Model

Vishal Soni

Matriculation Number: 66158
24.10.2024

Professor Incharge
Dr.-Ing. Arun Prakash
Dr.-Ing. Stefan Prüger

Contents

Abstract

This work presents a nonlinear finite element approach for a fully coupled thermoelastic material model, where heat transmission and mechanical deformation are intrinsically interdependent. The thermoelastic material model is essential in many technical applications, including metal forming, additive manufacturing, and aerospace systems, where temperature changes cause mechanical deformations and vice versa. The suggested approach incorporates mechanical and nonlinear thermal concepts into the finite element model. The transient heat conduction equation, which includes temperature-dependent conductivity and specific heat, governs the thermal problem. Simultaneously, the mechanical response is described using nonlinear elasticity, where properties such as Young's modulus depend on temperature. The Newton-Raphson approach solves the nonlinear governing equations iteratively after deriving the coupled system's weak form. The algorithm also considers nonlinear boundary conditions that affect the mechanical response, such as radiation and convective heat transfer. The algorithm was verified by conducting test cases to verify the temperature field, along with standard numerical testing, and comparing the results with those obtained from Abaqus simulations. The verification demonstrates the accuracy of the proposed approach in predicting temperature distributions, displacements, and stress fields under varying thermal loads.

Keywords: Nonlinear finite element, Thermoelastic material model, Temperature-dependent material properties, Radiation and convective heat transfer.

1 Introduction

Modern engineering applications are becoming more complex, which calls for sophisticated analysis methods that can simulate the interplay between mechanical and thermal phenomena. Temperature changes have a major impact on mechanical behavior in many industrial processes, including metal forming, additive manufacturing, and aerospace engineering, and vice versa. For instance, the heat produced during material deposition in additive manufacturing may result in thermal gradients that distort the finished product and leave residual stresses. Similar to this, temperature variations brought on by external factors can compromise the structural integrity of parts in aerospace systems.

Conventional analysis techniques frequently separate mechanical and thermal effects, producing inaccurate and simplistic results. The need for a fully coupled approach that can capture the complex interdependencies between temperature and mechanical response is highlighted by this limitation. Engineers can more accurately predict how materials will respond to changing thermal loads by incorporating nonlinear thermoelasticity into finite element analysis (FEA). This leads to better designs, increased performance, and fewer failure rates.

1.1 Objective of Fully Coupled Nonlinear Thermoelastic Finite Element Analysis

The main goal of fully coupled nonlinear thermoelastic finite element analysis is to create a reliable computational model that faithfully represents the relationship between mechanical and thermal phenomena in materials exposed to different temperatures. The following particular objectives are the focus of this analysis:

Precise Modeling: To guarantee that both thermal and mechanical behaviors are accurately represented, incorporate temperature-dependent material properties, such as thermal conductivity and Young's modulus, into the finite element model.

Iterative Solution: To solve the coupled governing equations effectively and efficiently while ensuring convergence under various load conditions, apply iterative solution techniques, such as the Newton-Raphson approach.

Verification: To show the suggested method's accuracy and reliability in predicting temperature distributions, displacements, and stress fields, thoroughly test it numerically and compare it to well-known simulation programs like Abaqus.

Applications in Engineering: Give engineers a useful tool to evaluate and improve designs in important applications, resulting in higher-quality, safer, and more effective products. In the end, this analysis will help in the creation of new materials and procedures that are strong to the demands of today's engineering problems.

2 Theory

2.1 Mechanical Equilibrium Equation

In static conditions (without inertia), the mechanical equilibrium equations can be expressed as follows [1] :

$$\nabla \cdot \sigma(T) + \nabla \cdot (\beta(T)\Delta T) = \mathbf{f} \tag{1}$$

Where:

- $\sigma(T)$ is the stress tensor dependent on temperature T.
- $\beta(T)$ is a function of temperature representing the thermal expansion coefficient or related properties.
- ΔT is the temperature change, defined as the difference between the current temperature and a reference temperature.
- $\mathbf{f}$ represents the external body forces per unit volume acting on the material.

2.2 Constitutive Equation for Thermoelasticity

The stress-strain relationship for a linear thermoelastic material with temperature dependence is given by [1]:

$$\sigma = C(T) : (\epsilon - \alpha(T)\Delta T\mathbf{a}) \tag{2}$$

In this equation:

- $C(T)$ is the fourth-order elasticity tensor, which incorporates temperature-dependent elastic modulus $E(T)$ and Poisson's ratio $\nu(T)$.
- ϵ is the total strain tensor, defined as [1]:

$$\epsilon = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right)$$

- $\alpha(T)$ is the thermal expansion coefficient.
- $\mathbf{a}$ is defined as [1]:

$$\mathbf{a} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T$$

This equation demonstrates how mechanical stresses are generated not only by elastic deformation but also by temperature variations.

2.3 Energy Conservation Equation

The heat transfer within the material, influenced by deformation, is governed by the following equation [1]:

$$\rho(T)c(T)\frac{\partial T}{\partial t} + T_0\nabla \cdot \left(\beta(T)\frac{\partial u}{\partial t}\right) - \nabla \cdot (k(T)\nabla T) = H \tag{3}$$

In this equation:

- $\rho(T)$ is the temperature-dependent density.
- $c(T)$ is the specific heat capacity.
- $k(T)$ is the thermal conductivity.
- H is the internal heat source.

2.4 Boundary and Initial Conditions

2.4.1 Mechanical Boundary Conditions

The mechanical boundary conditions are defined as follows [1]:

$$u = \bar{u} \quad \text{on } \Gamma_u, \tag{4}$$

$$\sigma \cdot n = \bar{t} \quad \text{on } \Gamma_s. \tag{5}$$

In these equations:

- $\bar{u}$ is the prescribed displacement.
- $\bar{t}$ is the traction force on the boundary Γ_s.
- n is the normal vector on the surface.

2.4.2 Thermal Boundary Conditions

The thermal boundary conditions are categorized as follows [1]:

Dirichlet Boundary Condition (Fixed Temperature):

$$T = \bar{T} \quad \text{on } \Gamma_T. \tag{6}$$

Neumann Boundary Condition (Heat Flux):

$$-k(T)\nabla T \cdot n = q \quad \text{on } \Gamma_q. \tag{7}$$

Convective Boundary Condition:

$$-k(T)\nabla T \cdot n = h(T - T_\infty) \quad \text{on } \Gamma_c. \tag{8}$$

Radiative Boundary Condition:

$$-k(T)\nabla T \cdot n = \kappa\sigma(T^4 - T_e^4) \quad \text{on } \Gamma_R. \tag{9}$$

In these equations:

- h is the convective heat transfer coefficient.
- T_∞ is the ambient fluid temperature.
- κ is the emissivity of the surface.
- $\sigma = 5.67 \times 10^{-8}\,\mathrm{W/m^2\,K^4}$ is the Stefan-Boltzmann constant.
- T_e is the environment temperature.

2.4.3 Initial Conditions

The initial conditions are given by [1]:

$$T(x,0) = T_0(x), \quad u(x,0) = u_0(x). \tag{10}$$

3 Available Codes on World Wide web

In the Worldwide Web, MATLAB code for 1-D coupled thermo-mechanics linear finite element code from Abdullah Waseem aligns closely with the intended project [4]. Limitation of the source code is linear thermo-mechanics problem and solve only for 1-D case. The intended project will be done in 2-D case and incorporate nonlinearities into the thermoelastic problem i.e., material and thermal properties dependence on temperature.

4 Numerical methods and Implementation Details

4.1 Weak Form of Mechanical Equation

To derive the weak form of equation (1), we multiply by a virtual displacement $\boldsymbol{\delta u}$ and integrate over the domain Ω [1]:

$$\int_\Omega \boldsymbol{\delta u}^T \left(\nabla \cdot \boldsymbol{\sigma}(T) + \nabla \cdot (\boldsymbol{\beta}(T)\Delta T)\right) \, d\Omega = \int_\Omega \boldsymbol{\delta u}^T \boldsymbol{f} \, d\Omega \tag{11}$$

Using the divergence theorem:

$$\int_\Omega \nabla \boldsymbol{\delta u}^T \boldsymbol{\sigma}(T) \, d\Omega + \int_\Omega \nabla \boldsymbol{\delta u}^T \boldsymbol{\beta}(T) \Delta T \, d\Omega = \int_\Omega \boldsymbol{\delta u}^T \boldsymbol{f} \, d\Omega + \int_{\Gamma_t} \boldsymbol{\delta u}^T \bar{\boldsymbol{t}} \, d\Gamma \tag{12}$$

where Γ_t is the boundary with prescribed traction $\bar{\boldsymbol{t}}$.

4.2 Weak Form of Heat Equation

Multiplying by a virtual temperature δT and integrating over the domain on equation (3) Ω [1]:

$$\int_\Omega \delta T \, c(T) \frac{\partial T}{\partial t} \, d\Omega + \int_\Omega \delta T \, T_0 \nabla \cdot \left(\boldsymbol{\beta}(T) \frac{\partial \boldsymbol{u}}{\partial t}\right) \, d\Omega - \int_\Omega \delta T \, \nabla \cdot (k(T) \nabla T) \, d\Omega = \int_\Omega \delta T \, H \, d\Omega \tag{13}$$

Using the divergence theorem on the heat conduction term:

$$\int_\Omega \delta T \, c(T) \frac{\partial T}{\partial t} \, d\Omega + \int_\Omega \delta T \, T_0 \nabla \cdot \left(\boldsymbol{\beta}(T) \frac{\partial \boldsymbol{u}}{\partial t}\right) \, d\Omega + \int_\Omega \nabla \delta T \cdot k(T) \nabla T \, d\Omega = \int_\Omega \delta T \, H \, d\Omega + \int_{\Gamma_q} \delta T \, \bar{q} \, d\Gamma \tag{14}$$

where Γ_q is the boundary with prescribed heat flux $\bar{q}$.

4.3 Discretization Using 2D Shape Functions

We discretize the displacement $\boldsymbol{u}$ and temperature T using 2D shape functions N_i and Θ_i respectively as follows [6]:

$$\boldsymbol{u} = \sum_{i=1}^{n} N_i(\xi, \eta) \, \boldsymbol{u}_i, \quad T = \sum_{i=1}^{n} \Theta_i(\xi, \eta) \, T_i \tag{15}$$

where n is the number of nodes per element, and $\boldsymbol{u}_i$ and T_i are the nodal displacement and temperature values, respectively.
The virtual displacement $\boldsymbol{\delta u}$ and virtual temperature δT are similarly expressed [6]:

$$\boldsymbol{\delta u} = \sum_{i=1}^{n} N_i(\xi, \eta)\, \delta \boldsymbol{u}_i, \quad \delta T = \sum_{i=1}^{n} \Theta_i(\xi, \eta)\, \delta T_i \tag{16}$$

$$N_i = \begin{bmatrix} N_1 & 0 & N_2 & 0 & \dots & N_n & 0 \\ 0 & N_1 & 0 & N_2 & \dots & 0 & N_n \end{bmatrix}$$

$$\Theta_i = \begin{bmatrix} \Theta_1 & \Theta_2 & \dots & \Theta_n \end{bmatrix}$$

The gradients of displacement and temperature fields are expressed as [6]:

$$\nabla \boldsymbol{u} = B_u\, \boldsymbol{U}, \quad \nabla T = B_T\, \boldsymbol{T} \tag{17}$$

$$\delta \nabla \boldsymbol{u} = B_u\, \boldsymbol{\delta U}, \quad \delta \nabla T = B_T\, \boldsymbol{\delta T} \tag{18}$$

where:

- B_u is the displacement, relating nodal displacements to the strain field.
- B_T is the temperature gradient, relating nodal temperatures to the temperature gradient field.
- $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_n]^T$ is the vector of nodal displacements.
- $\boldsymbol{T} = [T_1, T_2, \dots, T_n]^T$ is the vector of nodal temperatures.

The displacement B_u for 2D elements is given by [6]:

$$B_u = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & 0 & \frac{\partial N_2}{\partial \xi} & 0 & \dots & \frac{\partial N_n}{\partial \xi} & 0 \\ 0 & \frac{\partial N_1}{\partial \eta} & 0 & \frac{\partial N_2}{\partial \eta} & \dots & 0 & \frac{\partial N_n}{\partial \eta} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_2}{\partial \xi} & \dots & \frac{\partial N_n}{\partial \eta} & \frac{\partial N_n}{\partial \xi} \end{bmatrix}$$

Similarly, the temperature gradient B_T is [6]:

$$B_T = \begin{bmatrix} \frac{\partial \Theta_1}{\partial \xi} & \frac{\partial \Theta_2}{\partial \xi} & \dots & \frac{\partial \Theta_n}{\partial \xi} \\ \frac{\partial \Theta_1}{\partial \eta} & \frac{\partial \Theta_2}{\partial \eta} & \dots & \frac{\partial \Theta_n}{\partial \eta} \end{bmatrix}$$

Substituting Equations (15), (16), (17) and (18) into Equations (12) and (14), the discretized form of the nonlinear thermomechanical coupling equations in Ω are obtained and transformed into matrix form as follows [1]:

$$\mathbf{K_u}(T)\, \mathbf{u}(t) - \mathbf{L}(T)\, \mathbf{T}(t) = \mathbf{F}(t) \tag{19}$$

$$\mathbf{M_T}(T)\, \dot{\mathbf{T}}(t) + \mathbf{K_T}(T)\, \mathbf{T}(t) + \mathbf{L}(T)^T\, \dot{\mathbf{u}}(t) = \mathbf{G}(t) \tag{20}$$

where the matrices are defined as follows:

$$\mathbf{K_u}(T) = \int_{\Omega} \mathbf{B_u}^T \mathbf{D}(T)\, \mathbf{B_u}\, d\Omega \tag{21}$$

$$\mathbf{L}(T) = \int_{\Omega} \beta(T)\, \mathbf{B_u}^T \boldsymbol{a\Theta}\, d\Omega \tag{22}$$

$$\mathbf{M_T}(T) = \int_\Omega \rho(T)\, c(T)\, \mathbf{\Theta}^T \mathbf{\Theta}\, d\Omega \tag{23}$$

$$\mathbf{K_T}(T) = \int_\Omega \mathbf{B_T}^T k(T)\, \mathbf{B_T}\, d\Omega \tag{24}$$

$$\mathbf{F}(t) = \int_\Omega \mathbf{N}^T \mathbf{f}\, d\Omega + \int_{\Gamma_t} \mathbf{N}^T \bar{\mathbf{p}}\, d\Gamma \tag{25}$$

$$\mathbf{G}(t) = \int_\Omega H\, \mathbf{\Theta}^T\, d\Omega + \int_{\Gamma_q} q\, \mathbf{\Theta}^T\, d\Gamma + \int_{\Gamma_c} h(\mathbf{\Theta}T - T_\infty)\, \mathbf{\Theta}^T\, d\Gamma + \int_{\Gamma_R} \kappa\sigma(\mathbf{\Theta}T^4 - T_e^4)\, \mathbf{\Theta}^T\, d\Gamma \tag{26}$$

The constitutive matrix $\mathbf{D}(T)$ for a thermoelastic material under plane stress conditions is given by [1]:

$$\mathbf{D}(T) = \frac{E(T)}{1-\nu(T)^2} \begin{bmatrix} 1 & \nu(T) & 0 \\ \nu(T) & 1 & 0 \\ 0 & 0 & \frac{1-\nu(T)}{2} \end{bmatrix}$$

The thermal expansion coefficient matrix $\boldsymbol{\beta}(T)$ for a thermoelastic material is given by [1]:

$$\beta(T) = \frac{\alpha(T)E(T)}{1-\nu(T)}$$

where:

- $E(T)$ is the Young's modulus as a function of temperature.
- $\nu(T)$ is the Poisson's ratio as a function of temperature.
- $\alpha(T)$ is the temperature-dependent coefficient of thermal expansion.

The matrices $\mathbf{K_u}(T)$, $\mathbf{L}(T)$, and $\mathbf{M_T}(T)$ represent the temperature-dependent stiffness matrix, thermomechanical coupling matrix, and thermal capacitance matrix, respectively, taking into account material nonlinearity. The matrix $\mathbf{K_T}(T)$ is the heat conductivity matrix, which varies with temperature when considering material and boundary nonlinearities.The vectors $\mathbf{F}(t)$ and $\mathbf{G}(t)$ denote the total mechanical and thermal load vectors, respectively, applied to different boundaries.

4.4 Implicit Time discretization

Using the Euler implicit time integration scheme, the equations (19) and (20) are discretized as follows [1]:
1. The mechanical equation:

$$\mathbf{K_u}(T)u^{n+1} - \mathbf{L}(T)T^{n+1} = \mathbf{F}(t^{n+1}) \tag{27}$$

2. The thermal equation:

$$\mathbf{M_T}(T)\frac{T^{n+1}-T^n}{\Delta t} + \mathbf{K_T}(T)T^{n+1} + \mathbf{L}(T)^T u^{n+1} = \mathbf{G}(t^{n+1}) \tag{28}$$

Define a vector $\mathbf{X}^{n+1}$ that contains the unknowns

$$\mathbf{X}^{n+1} = \begin{bmatrix} u^{n+1} \\ T^{n+1} \end{bmatrix}$$

Rearranging the original equations, we express them in matrix form as follows:

$$\begin{bmatrix} \mathbf{K_u}(T^{n+1}) & -\mathbf{L}(T^{n+1}) \\ \mathbf{L}(T^{n+1})^T & \frac{1}{\Delta t}\mathbf{M_T}(T^{n+1}) + \mathbf{K_T}(T^{n+1}) \end{bmatrix} \begin{bmatrix} u^{n+1} \\ T^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}(t^{n+1}) \\ \mathbf{G}(t^{n+1}) + \frac{1}{\Delta t}\mathbf{M_T}(T)T^n \end{bmatrix} \tag{29}$$

The final combined matrix equation can be summarized as:

$$\mathbf{K(T)X}^{n+1} = \mathbf{F(T)} \tag{30}$$

where:

$$\mathbf{K(T)} = \begin{bmatrix} \mathbf{K_u}(T^{n+1}) & -\mathbf{L}(T^{n+1}) \\ \mathbf{L}(T^{n+1})^T & \frac{1}{\Delta t}\mathbf{M_T}(T^{n+1}) + \mathbf{K_T}(T^{n+1}) \end{bmatrix}$$

and

$$\mathbf{F(T)} = \begin{bmatrix} \mathbf{F}(t^{n+1}) \\ \mathbf{G}(t^{n+1}) + \frac{1}{\Delta t}\mathbf{M_T}(T)T^n \end{bmatrix}$$

This form allows for the simultaneous solution of the displacement and temperature variables at each time step, considering their coupling through the system matrices.

4.5 2D Quadrilateral Elements: Shape Functions and Jacobian Matrix

2D quadrilateral (quad) elements are widely used in finite element analysis to simulate two-dimensional fields. Shape functions based on nodal values are used to interpolate the variation of the field variables, such as temperature or displacement, within each quad element, which has four nodes. Within the range $[-1, 1]$, local coordinates (ξ, η) are used for interpolation of 2D quadrilateral elements.

4.5.1 Shape Functions for 2D Quad Elements

The shape functions $N_i(\xi, \eta)$ for a 2D quadrilateral element are bilinear and can be expressed as [6]:

$$N_1(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 - \eta) \tag{31}$$

$$N_2(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 - \eta) \tag{32}$$

$$N_3(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 + \eta) \tag{33}$$

$$N_4(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 + \eta) \tag{34}$$

These shape functions are used to interpolate the displacement or temperature field within the element.

4.5.2 Jacobian Matrix

The Jacobian matrix is required to transform the derivatives from the local coordinate system (ξ, η) to the global coordinate system (x, y). It is defined as [1]:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{35}$$

The entries of the Jacobian matrix can be computed based on the nodal coordinates of the element:

$$\frac{\partial x}{\partial \xi} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \xi} x_i, \qquad \frac{\partial x}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} x_i \tag{36}$$

$$\frac{\partial y}{\partial \xi} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \xi} y_i, \qquad \frac{\partial y}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} y_i \tag{37}$$

4.5.3 Derivatives of Shape Functions in Global Coordinates

To transform the shape function derivatives from the local coordinates (ξ, η) to the global coordinates (x, y), the inverse of the Jacobian matrix is used. The relationship between the derivatives in the two coordinate systems is given by [6]:

$$\begin{bmatrix} \frac{dN}{dx} \\ \frac{dN}{dy} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{dN}{d\xi} \\ \frac{dN}{d\eta} \end{bmatrix} \tag{38}$$

Finally, the global derivatives $\frac{dN}{dx}$ and $\frac{dN}{dy}$ are obtained by multiplying the inverse of the Jacobian by the local shape function derivatives $\frac{dN}{d\xi}$ and $\frac{dN}{d\eta}$. These are utilized in the construction of the strain-displacement matrix $\mathbf{B_u}$ or gradient matrix for temperature fields $\mathbf{B_T}$ in finite element formulations.

4.6 Numerical Integration Using Gauss Quadrature

Four-point Gauss quadrature is used to numerically integrate the element matrices. For the numerical integration of the heat flux vector, radiation, and convection boundary vectors along the element edge, a two-point quadrature rule is used. For the Four-point Gauss quadrature, there are four Gauss points, located at [6]:

$$(\xi_1, \eta_1) = \left(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right)$$

$$(\xi_2, \eta_2) = \left(\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right)$$

$$(\xi_3, \eta_3) = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$$

$$(\xi_4, \eta_4) = \left(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$$

The corresponding weights for each Gauss point in both ξ and η directions are $w = 1$.
Where $f(\xi, \eta)$ is evaluated at the Gauss points and summed up using the weights to approximate the integral [6].

$$\int_{\Omega_e} f(x, y)\, d\Omega = \int_{-1}^{1} \int_{-1}^{1} f(\xi, \eta)\, |\mathbf{J}|\, d\xi\, d\eta \approx \sum_{i=1}^{4} w_i\, f(\xi_i, \eta_i)\, |\mathbf{J}(\xi_i, \eta_i)| \tag{39}$$

where:

- w_i are the quadrature weights (for 2x2 Gauss quadrature, $w_i = 1$),
- (ξ_i, η_i) are the local coordinates of the Gauss points,
- $|\mathbf{J}|$ is the determinant of the Jacobian matrix, which accounts for the transformation from local to global coordinates.

4.7 Newton-Raphson Method for Nonlinear Coupled System

The finite element formulation leads to nonlinear coupled equations that are solved using the Newton-Raphson method. In structural and thermal analyses involving material nonlinearity, this iterative method is frequently employed because it is especially good at locating the roots of a nonlinear system of equations [1].

4.7.1 Iterative Procedure

1. **Initial Guess:** Begin with an initial guess for the unknowns $\mathbf{X}^{(0)} = \begin{bmatrix} u^{(0)} \\ T^{(0)} \end{bmatrix}$.
2. **Residual Calculation:** At each iteration i, calculate the residuals of equation (30):

$$\mathbf{R}^{(i+1)} = \mathbf{F}(\mathbf{T}^{(i+1)}) - \mathbf{K}(\mathbf{T}^{(i+1)})\mathbf{X}^{(i+1)} \tag{40}$$

3. **Linearization:** The tangent stiffness matrix $\mathbf{K}_t$ is computed, which contains the partial derivatives of the residuals with respect to the unknowns:

$$\mathbf{K}_t = \frac{\partial \mathbf{R}}{\partial \mathbf{X}} \tag{41}$$

4. **Solution Update:** Solve for the correction term:

$$\Delta \mathbf{X} = -\mathbf{K}_t^{-1}\mathbf{R}^{(i)} \tag{42}$$

5. **Update Rule:** The update for the unknowns is given by:

$$\mathbf{X}^{(i+1)} = \mathbf{X}^{(i)} + \Delta \mathbf{X} \tag{43}$$

6. **Convergence Criteria:** The iteration continues until the convergence criteria are met, typically based on the norm of the residuals or the norm of the solution update:

$$\|\mathbf{R}^{(i)}\| < \epsilon \quad \text{or} \quad \|\Delta \mathbf{X}\| < \epsilon \tag{44}$$

where ϵ is approximately $1 \times e^{-6}$ a specified tolerance level.

4.8 Programming Language Selection, Code Adaption and Software selection

- Develop a nonlinear finite element analysis of a thermoelastic material model using the MATLAB programming language.
- The adaptation of the element stiffness routine code for mechanical part from the existing finite element MATLAB framework developed by Siva Srinivas Kolukula will be utilized, which focuses on 2-D plane stress problems [5]. This code solves a plate subjected to uniform tension at its edges using the Linear Finite Element Method. The following functionalities will be integrated to account for nonlinear effects:
  - Incorporation of temperature-dependent stiffness matrix.
  - Implementation of temperature-dependent thermomechanical coupling matrix.
  - Integration of temperature-dependent heat conductivity matrix.
  - Development of a full thermomechanical coupled solution using a monolithic scheme.
- Use Gmsh software to create a mesh for preprocessing for both the square domain and the square domain with a center hole.
- Utilize contour plots in MATLAB for postprocessing and visualization of results.

5 Algorithm FlowChart

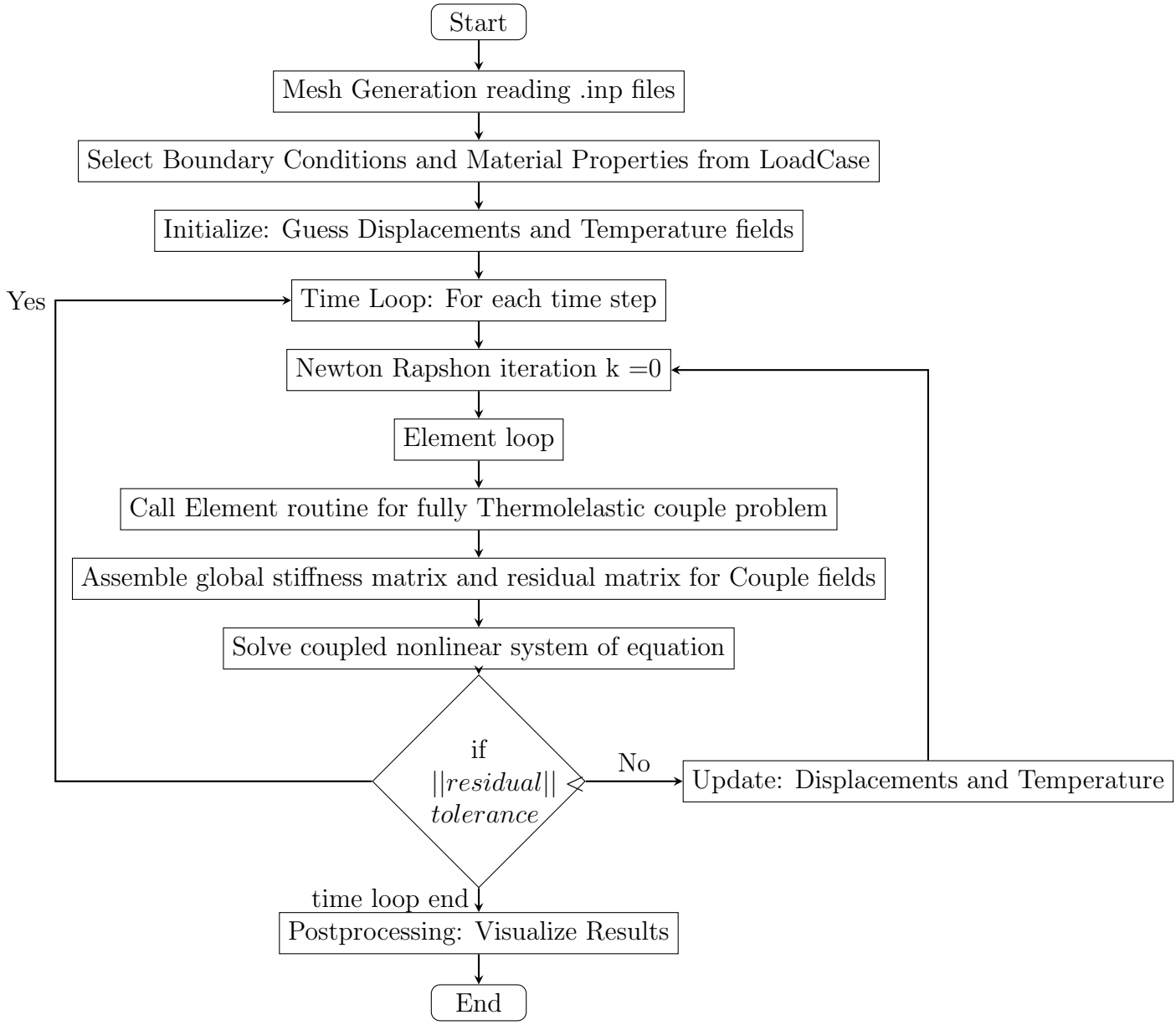


Figure 1: Flowchart for Coupled Nonlinear Finite Element Analysis

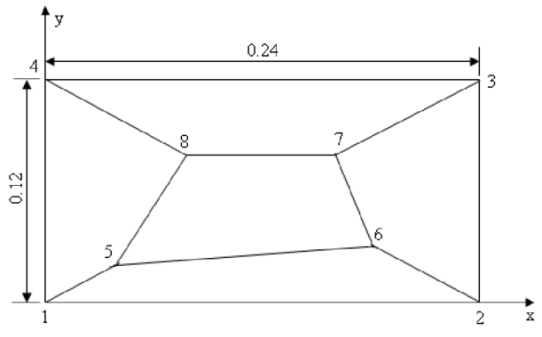
6 Usage of Chatbots

ChatGPT was utilized in the development of code by providing suggestions for optimizations and fixing syntax errors. It also offered assistance in organizing test cases and recognizing possible edge cases for verification.
ChatGPT enhanced the clarity, grammar, and overall readability of the report, while the technical content and conclusions were created manually. Suggestions for formatting and structuring were also provided, but final adjustments were made using conventional tools.

7 Testing

7.1 Heat Transfer Patch Test

The mesh,material and boundary conditions for the Heat Transfer patch test shown figure 2 taken from Abaqus [7].



Node	coordinates	
1	0.0	0.0
2	0.24	0.0
3	0.24	0.12
4	0.0	0.12
5	0.04	0.02
6	0.18	0.03
7	0.16	0.08
8	0.08	0.08

Figure 2: Patch of elements used for the test

Test Case: 1
Aim: The aim of the patch test is to verify the accuracy and consistency of the heat transfer element in reproducing a known linear temperature distribution. The test ensures that the element can handle a simple linear temperature field without introducing numerical errors.
Boundary Condition: The boundary condition applied for the patch test is a linear temperature distribution defined as:

$$T(x, y) = 200 \cdot x + 100 \cdot y$$

Specific temperature values in degrees Celsius are prescribed at the corner nodes:

$$T_1 = 0, T_2 = 48, T_3 = 60, T_4 = 12$$

Material Parameters: Thermal conductivity = 4.85e-4
Expected Result: For the inner nodes, which are located within the patch, the temperature values in degrees Celsius will be:

$$T_5 = 10, T_6 = 39, T_7 = 24, T_8 = 40.$$

Procedure to run this test: Set testCase = 3, eigenValueTest = 'no', numericalTesting = 'no', temperatureDependent = 'no', steadyState = 'yes' and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and results will be displayed in the command window.
Obtained Result: The obtained results are shown in the figure 3 named Heat Transfer patch test results.

```
TGlobal =

         0
   48.0000
   60.0000
   12.0000
   10.0000
   39.0000
   24.0000
   40.0000
```

Figure 3: Heat Transfer patch test results

7.2 Testing against Exact Solution for Steady-State Heat Transfer

The exact solution for the temperature distribution in a square domain is taken from [2],[3] for different element mesh sizes (h). It is expressed as a function of x and y coordinates, providing a benchmark for comparison with the numerical results obtained in this study. The mesh generated for the square plate geometry is shown in Figure 4. User can change element mesh sizes (h) with readFile = fopen('square100Element.inp', 'r') in mainCouplefem.m by using given square400Element.inp for square domain or any different inp files.

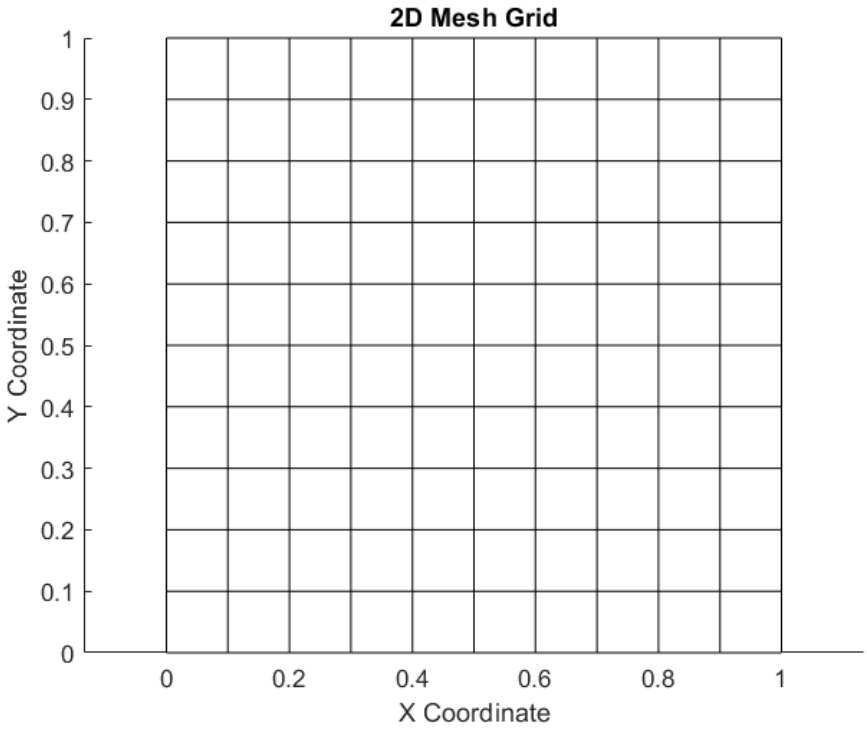


Figure 4: 2d Sqaure Plate Mesh (h = 0.1[m])

Test Case: 2
Aim The aim of the exact solution for steady-state heat transfer in a square domain with structured mesh for different element mesh sizes (h) is to provide a precise reference for validating the results obtained from numerical simulations. This exact solution enables the

assessment of the accuracy and effectiveness of the numerical methods employed in modeling heat transfer.

Boundary Condition For the square domain with a length of 1 meter, the boundary conditions are shown in figure 5:

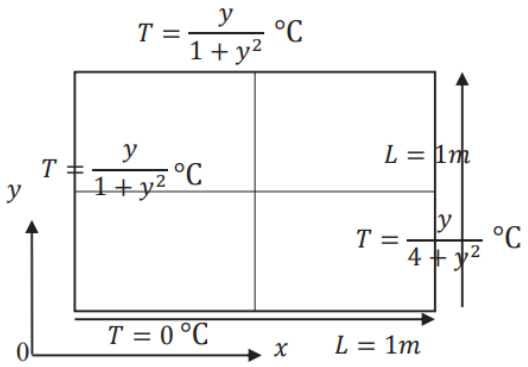


Figure 5: 2D sqaure plate with Dirichlet condition)

Material Parameters: Thermal conductivity = 1

Exact Solution: The exact solution for the temperature distribution T in degree Celsius in the square domain is given by [2]:

$$T_{\text{exact}} = \frac{y}{(1+x)^2 + y^2}$$

Procedure to run this test: Set testCase = 1, eigenValueTest = 'no', numericalTesting = 'no', temperatureDependent = 'no', steadyState = 'yes' and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and results will be displayed in the command window.

Obtained Result: The obtained results are shown in the figure 6 and error norm values shown in Table 1:

Table 1: Comparison of Error Norms of Exact Solution and FEM Solution(Test Case=2)

Mesh Size (h)[m]	**Error Norm** ($T_{\text{exact}} - T_{\text{FEM}}$)
0.1	0.0018
0.05	8.7521e-04
0.025	4.3686e-04
0.011111	1.9407e-04

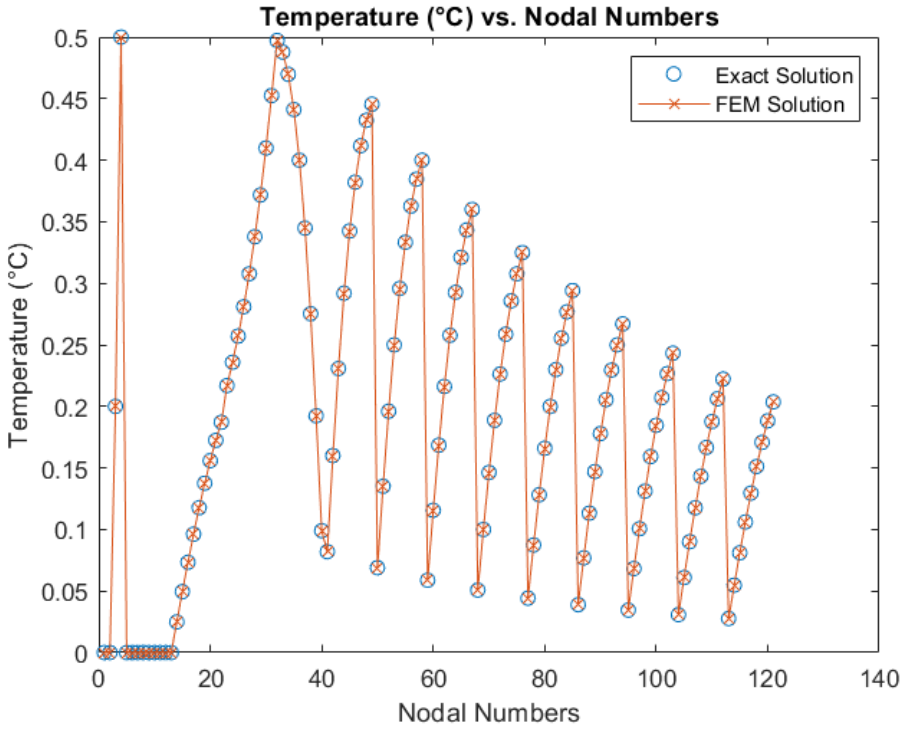


Figure 6: Comparison of Exact Solution and FEM Solution for Steady-State Heat Transfer for (h = 0.1)

Test Case: 3

Aim The aim of the exact solution for steady-state heat transfer in a square domain with structured mesh for different element mesh sizes (h) is to provide a precise reference for validating the results obtained from numerical simulations. This exact solution enables the assessment of the accuracy and effectiveness of the numerical methods employed in modeling heat transfer.

Boundary Condition For the square domain with a length of 1 meter, the boundary conditions are shown in figure 7:

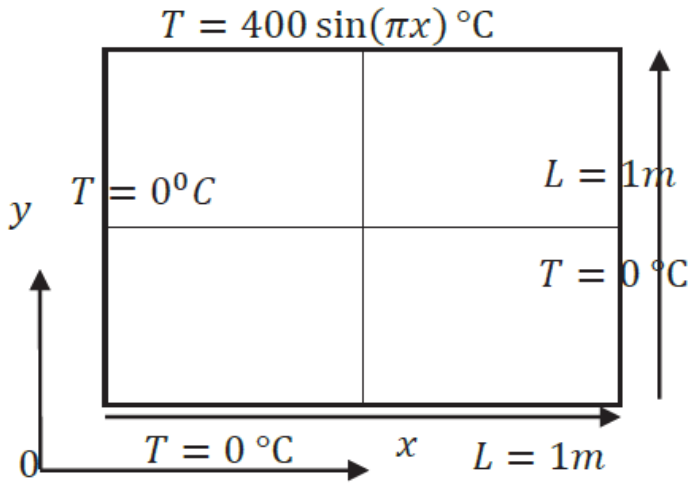


Figure 7: 2D sqaure plate with Dirichlet condition)

Material Parameters: Thermal conductivity = 1
Exact Solution: The exact solution for the temperature distribution T in the square domain is given by [2]:

$$T_{\text{exact}} = \left(-\frac{400e^{\pi y}}{\pi(e^{-\pi} - e^{\pi})} + \frac{400e^{-\pi y}}{\pi(e^{-\pi} - e^{\pi})} \right) \sin(\pi x).$$

Procedure to run this test: Set testCase = 2, eigenValueTest = 'no', numericalTesting = 'no', temperatureDependent = 'no', steadyState = 'yes' and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and results will be displayed in the command window. User can change element mesh sizes (h) with readFile = fopen('square100Element.inp', 'r')
Obtained Result: The obtained results are shown in the figure 8 and norm values shown in Table 2:

Table 2: Comparison of Error Norms of Exact Solution and FEM Solution(Test case=3)

Mesh Size (h)[m]	**Error Norm** $(T_{\text{exact}} - T_{\text{FEM}})$
0.1	1.7798
0.05	0.8843
0.025	0.4415
0.011111	0.1961

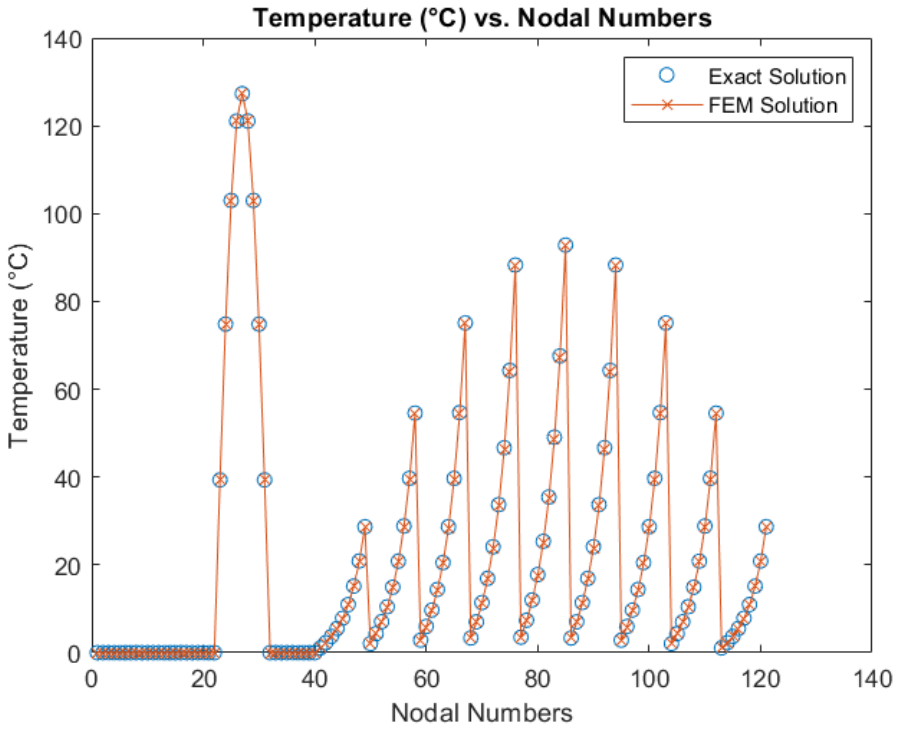


Figure 8: Comparison of Exact Solution and FEM Solution for Steady-State Heat Transfer for (h = 0.1)

Test Case: 4
Aim The aim is to carry out an analysis of heat transfer in a square domain for different element mesh size h while applying various boundary conditions, namely a heat flux boundary condition on the left and a convective boundary condition on the right [3].

Boundary Condition For the square domain with a length of 1 meter, the boundary conditions are shown in figure [8]:

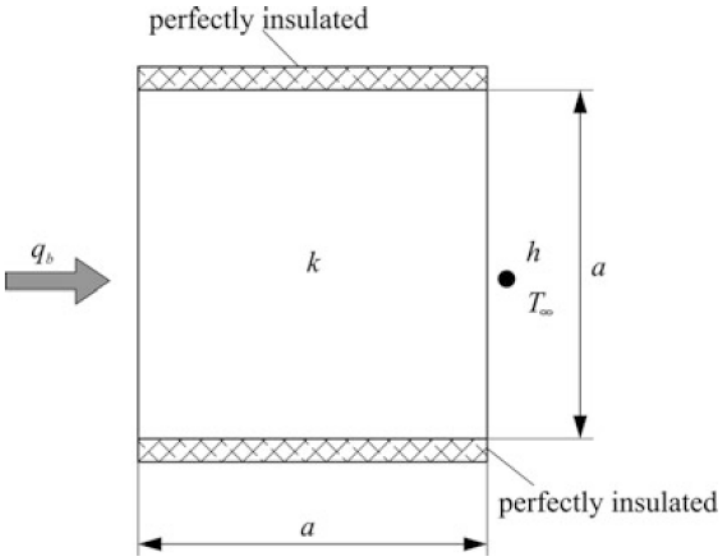


Figure 9: 2D sqaure plate with heat flux and convective condition (a = 1)

Material Parameters: Thermal Conductivity (k) = 50 W/m·K, Heat Flux q_b = 200,000 W/m^2, Heat Transfer Coefficient h = 1,000 W/(m^2·K), Ambient Temperature T_∞ = 20 °C.
Exact Solution: Due to thermal insulation of top and bottom surface, the temperature field is one-dimensional in the cross-section of the domain.The exact solution for the temperature distribution T in the square domain is given by [3]:

$$T(x) = \frac{q_b(a-x)}{k} + \frac{q_b}{h} + T_\infty$$

Procedure to run this test: Set testCase = 4, eigenValueTest = 'no', numericalTesting = 'no', temperatureDependent = 'no', steadyState = 'yes' and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and results will be displayed in the command window. User can change element mesh sizes (h) with readFile = fopen('square100Element.inp', 'r')
Obtained Result: The obtained results are shown in the figure 10 and norm values shown in Table 3:

Table 3: Comparison of Error Norms of Exact Solution and FEM Solution(Test case=4)

Mesh Size (h)[m]	**Error Norm** $(T_{\mathbf{exact}} - T_{\mathbf{FEM}})$
0.1	7.2851e-13
0.05	7.3134e-12
0.025	5.6023e-11
0.011111	5.6023e-11

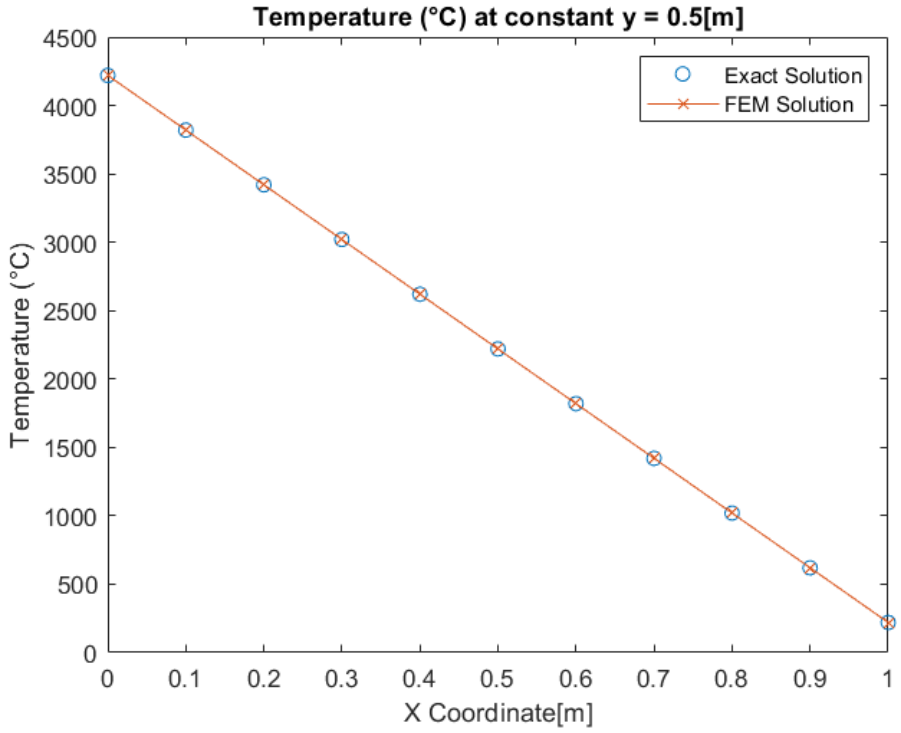


Figure 10: Comparison of Exact Solution and FEM Solution for Steady-State Heat Transfer for (h = 0.1)

7.3 Numerical Testing

Aim: The numerical testing aims to verify and validate the accuracy of the finite element method (FEM) implementation. This is achieved through several key tests:
1. Unity test for shape functions: Ensure that the sum of shape functions equals unity at any point within the element.
2. Sum of derivatives with respect to local variables: Verify that the sum of the derivatives of the shape functions with respect to local variables satisfies consistency conditions.
3. Sum of derivatives with respect to global variables: Check the accuracy of the transformation between local and global coordinates by summing the derivatives with respect to global variables.
4. Comparison of exact and FEM temperature gradients: Validate the FEM solution by comparing the computed temperature gradients to the exact analytical solution.
5. Comparison of analytical and FEM Jacobian matrices: Ensure consistency between the analytical and FEM-calculated Jacobian matrices, crucial for accurate element integration.
6. Measurement of the area using the determinant of the Jacobian: Confirm that the determinant of the Jacobian matrix correctly measures the area of the finite element, ensuring geometric accuracy. These tests collectively validate the reliability of the FEM implementation.
Set up: The analytical Jacobian matrix, which plays a crucial role in the finite element method, is given as:

$$\mathbf{J} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

The temperature field is approximated by the linear equation:

$$T(x, y) = a_0 + a_1 x + a_2 y$$

where a_0, a_1, and a_2 are coefficients to be determined. The known temperature values at specific points are:

$$T = \begin{bmatrix} 0 \\ 48 \\ 60 \\ 12 \end{bmatrix}$$

These values are used to compute the exact temperature gradient.
Procedure to run this test: Set testCase = 1, eigenValueTest = 'no', numericalTesting = 'yes', temperatureDependent = 'no', steadyState = 'yes' and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and results will be displayed in the command window.
Expected Result:The expected outcomes of the numerical testing are:
1. The sum of the shape functions, $\sum_{i=1}^{N} N_i$, will equal 1.
2. The sum of the derivatives of the shape functions with respect to the local variable, $\sum_{i=1}^{N} \frac{dN_i}{d\xi}$, will equal 0.
3. The sum of the derivatives of the shape functions with respect to the global variable, $\sum_{i=1}^{N} \frac{dN_i}{dx}$, will equal 0.
4. The computed temperature gradient will be:

$$\begin{bmatrix} 48 \\ 12 \end{bmatrix}$$

5. The Jacobian matrix will be:

$$\mathbf{J} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

6. The area of the unit square element will be 1 $\mathbf{m^2}$.
Obtained Result: The obtained results are shown in the figure 11:

```
Sum of shape functions at (xi, eta) = (-0.577350, -0.577350): 1.000000
Unity test passed.
Sum of derivatives of the shape functions with respect to local variable test passed.
Sum of derivatives of the shape functions with respect to physical coordinates test passed.
Sum of shape functions at (xi, eta) = (0.577350, -0.577350): 1.000000
Unity test passed.
Sum of derivatives of the shape functions with respect to local variable test passed.
Sum of derivatives of the shape functions with respect to physical coordinates test passed.
Sum of shape functions at (xi, eta) = (0.577350, 0.577350): 1.000000
Unity test passed.
Sum of derivatives of the shape functions with respect to local variable test passed.
Sum of derivatives of the shape functions with respect to physical coordinates test passed.
Sum of shape functions at (xi, eta) = (-0.577350, 0.577350): 1.000000
Unity test passed.
Sum of derivatives of the shape functions with respect to local variable test passed.
Sum of derivatives of the shape functions with respect to physical coordinates test passed.
Area of unit sqaure using determinant of Jacobian: 1
Fem B matrix temperature Gradient:
   48.0000
   12.0000

Fem Jacobian matrix:
    0.5000         0
         0    0.5000
```

Figure 11: Numerical Testing

Aim: The eigenvalue test for heat transfer aims to evaluate the accuracy and stability of the finite element formulation. Through eigenvalue analysis, the test confirms that the numerical model is reliable for predicting temperature distribution.
Procedure to run this test: Set testCase = 1, eigenValueTest = 'yes', numericalTesting = 'no', temperatureDependent = 'no', steadyState = 'yes' and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and results will be displayed in the command window.
Expected Result:The expected output from the Eigenvalue test for the heat transfer element is that there will be one zero eigenvalue.
Obtained Result: The obtained results are shown in the figure 12:

```
Eigen values for thermal element routine:
   -0.0000
    0.5918
    0.5918
    0.6667
    1.0000
    1.4082
    1.4082
    2.0000
    3.0000

Total Zero eigen value for thermal element routine:
     1
```

Figure 12: Zero Eigenvalue Test (4 Elements)

7.4 Testing against Abaqus Simulation Software for Thermoelastic Fully Couple Material Model (Without Temperature dependent Material properties)

The mesh generated for the square plate geometry is shown in Figure 3. User can change element mesh sizes (h) with readFile = fopen('square100Element.inp', 'r') in mainCouplefem.m by using given square400Element.inp for square domain or any different inp files. For all LoadCases Plain Stress condition is assumed.
LoadCase = 5
Boundary Conditions:For the square domain with a length of 1 meter, the Thermal and Mechanical boundary conditions are shown in figure 13:
Material Properties: Thermal conductivity: $k = 43\,\mathrm{W/m^{\circ}C}$, Young's modulus: $E = 2 \times 10^{11}\,\mathrm{Pa}$, Poisson's ratio: $\nu = 0.3$, Coefficient of thermal expansion: $\alpha = 1.2 \times 10^{-5}\,^{\circ}\mathrm{C}^{-1}$ [8].
Procedure to run this test: Set testCase = 0, LoadCase = 5, eigenValueTest = 'no', numericalTesting = 'no', temperatureDependent = 'no', steadyState = 'yes' and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and 2D contour plot results will be displayed.
Obtained results: The Matlab 2D contour plot results for this load case is verified with the Abaqus software 2D contour plot results with the above material parameters which are shown in the Figures 14 to 17. The results of Total strain, Thermal strain, Elastic strain and Thermal Stresses for every gauss points are stored in variable `totalStrain`, `thermalStrain`, `elasticStrain` and `thermalStress` in code.

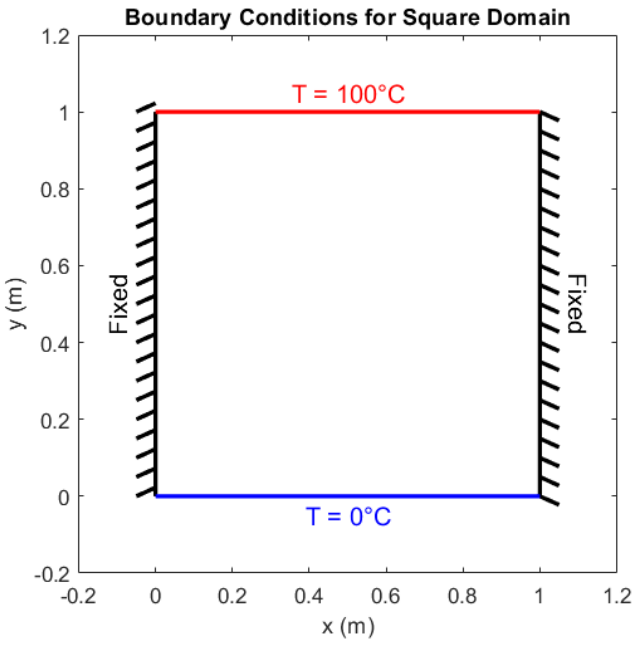
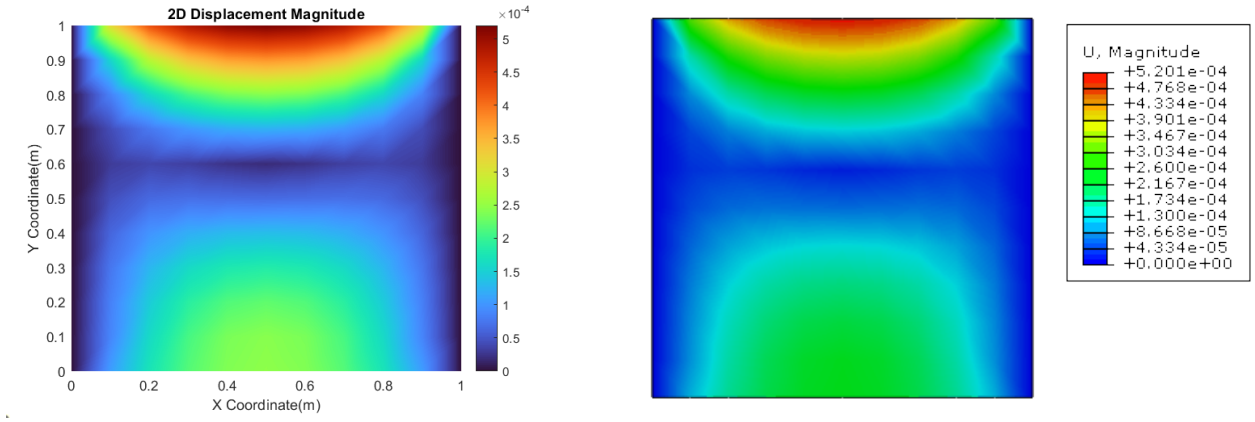
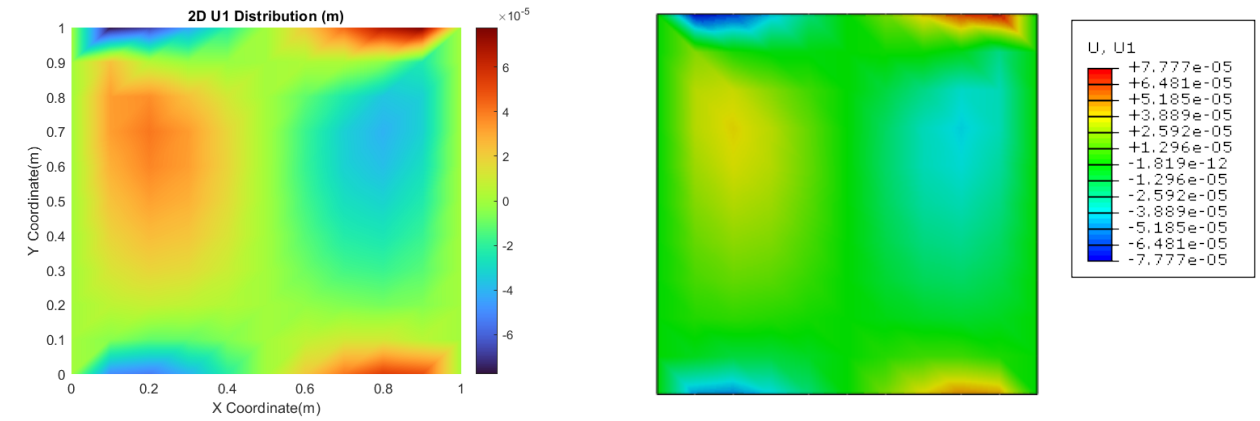


Figure 13: 2D sqaure plate with Boundary conditions



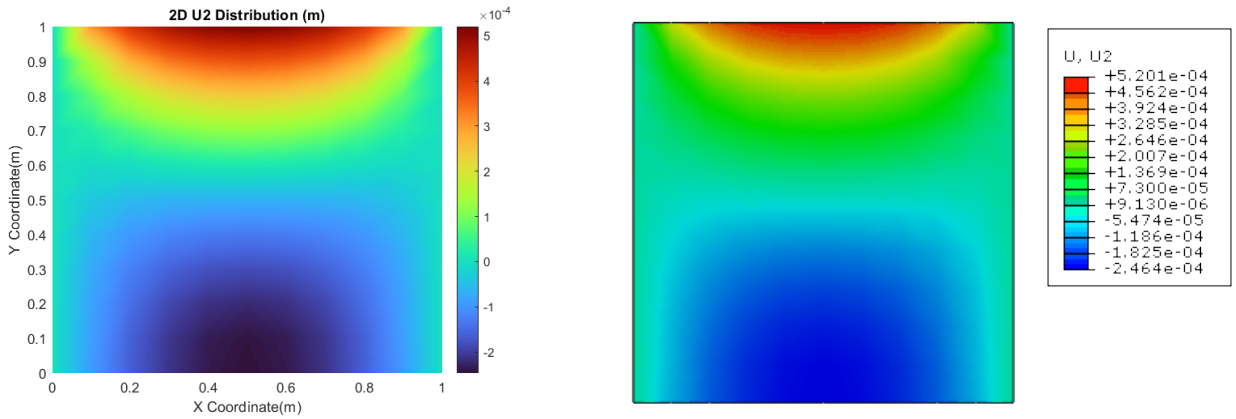
(a) (b)

Figure 14: Displacement Magnitude 'u' for loadcase=5 in (a) Matlab (b) Abaqus.



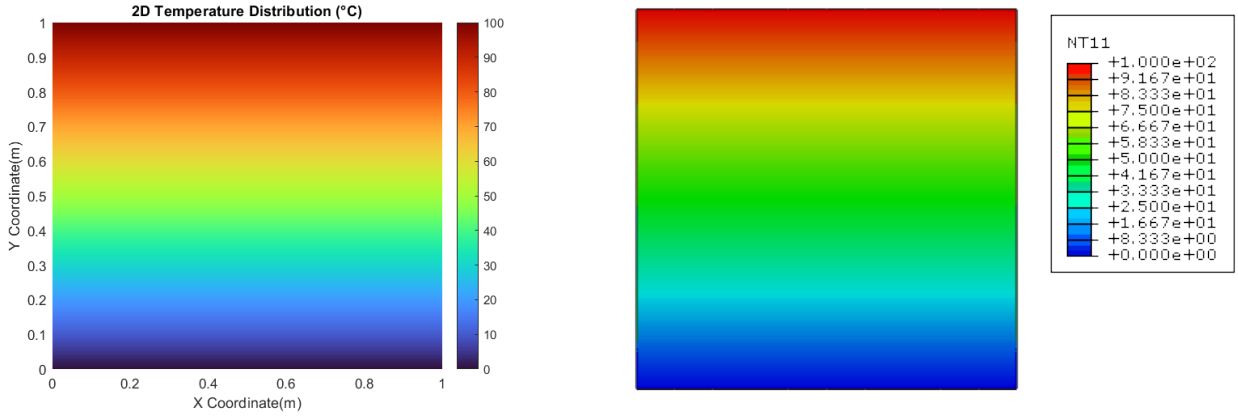
(a) (b)

Figure 15: Displacement 'u1' for loadcase=5 in (a) Matlab (b) Abaqus.



(a) (b)

Figure 16: Displacement 'u2' for loadcase=5 in (a) Matlab (b) Abaqus.



(a) (b)

Figure 17: Temperature distribution for loadcase=5 in (a) Matlab (b) Abaqus.

LoadCase = 6

Boundary Conditions:For the square domain with a length of 1 meter, the Thermal and Mechanical boundary conditions are shown in figure 18:

Material Properties: Thermal conductivity: $k = 50\,\mathrm{W/m^{\circ}C}$, Young's modulus: $E = 2 \times 10^{11}\,\mathrm{Pa}$, Poisson's ratio: $\nu = 0.3$, Coefficient of thermal expansion: $\alpha = 1.2 \times 10^{-5}\,^{\circ}\mathrm{C}^{-1}$, Heat Flux q_b = 200,000 $\mathrm{W/m^2}$, Heat Transfer Coefficient h = 1,000 $\mathrm{W/(m^2{\cdot}K)}$, Ambient Temperature $T_\infty = 20\ ^{\circ}\mathrm{C}$ [3], [8].

Procedure to run this test: Set testCase = 0, LoadCase = 6, eigenValueTest = 'no', numericalTesting = 'no', temperatureDependent = 'no', steadyState = 'yes' and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and 2D contour plot results will be displayed. window.

Obtained results:The Matlab 2D contour plot results for this load case is verified with the Abaqus software 2D contour plot results with the above material parameters which are shown in the Figures 19 to 22. The results of Total strain, Thermal strain, Elastic strain and Thermal Stresses for every gauss points are stored in variable `totalStrain`, `thermalStrain`, `elasticStrain` and `thermalStress` in code.

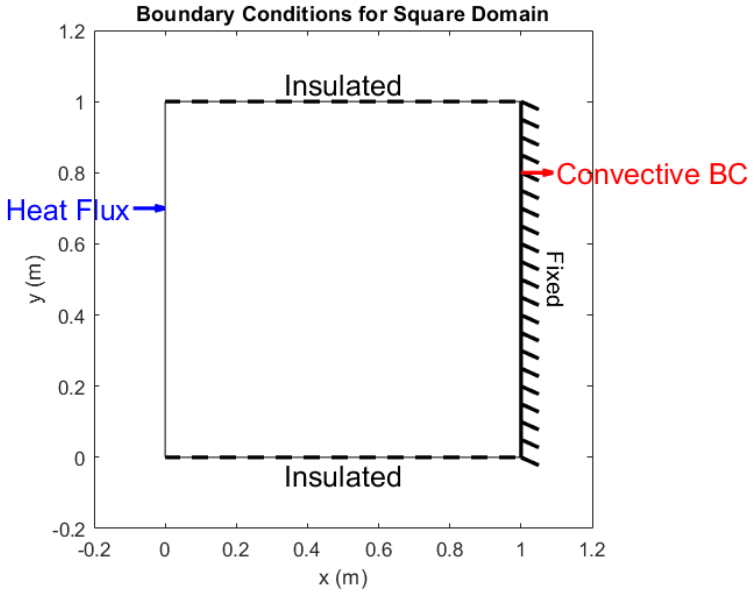
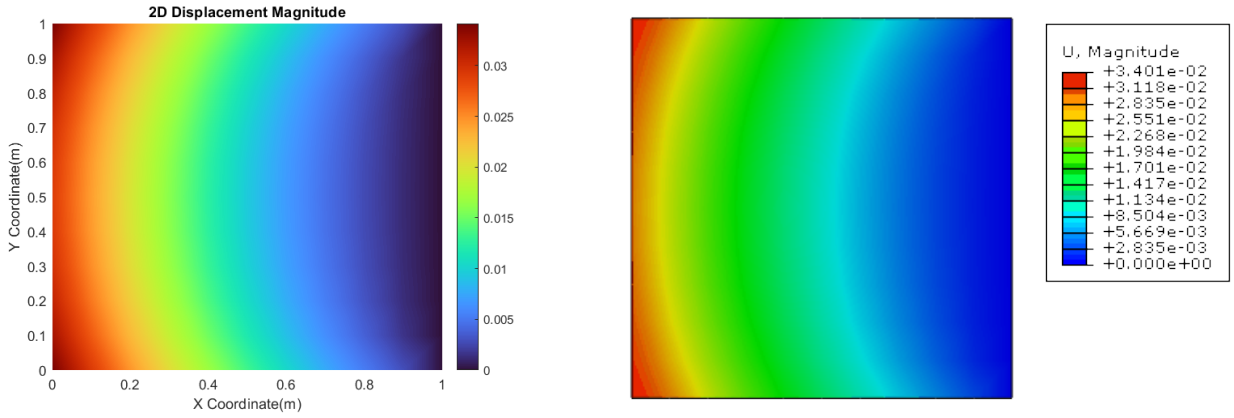
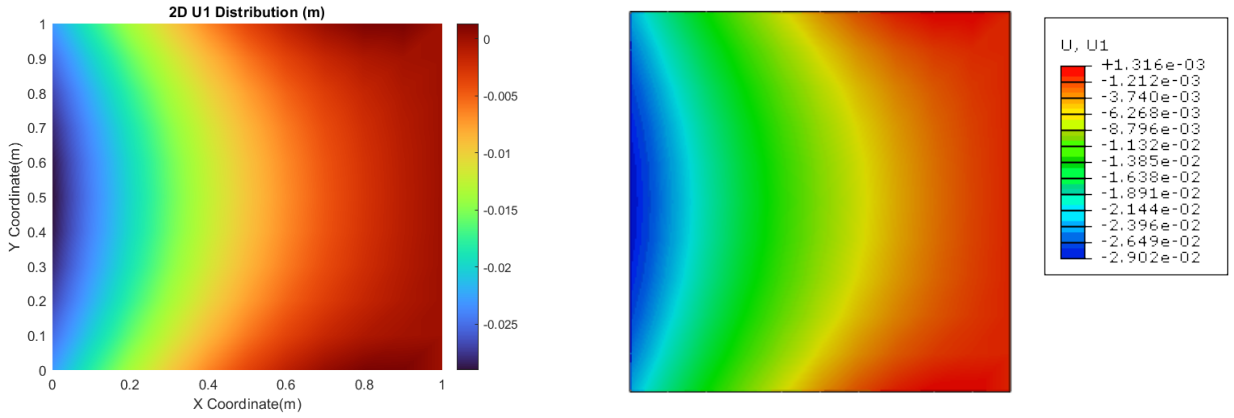


Figure 18: 2D sqaure plate with Boundary conditions



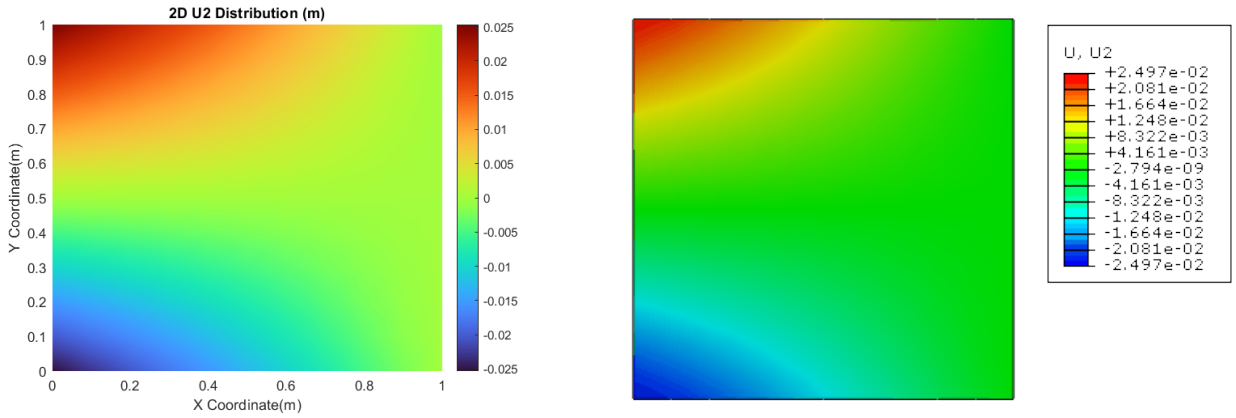
(a) (b)

Figure 19: Displacement Magnitude 'u' for loadcase=6 in (a) Matlab (b) Abaqus.



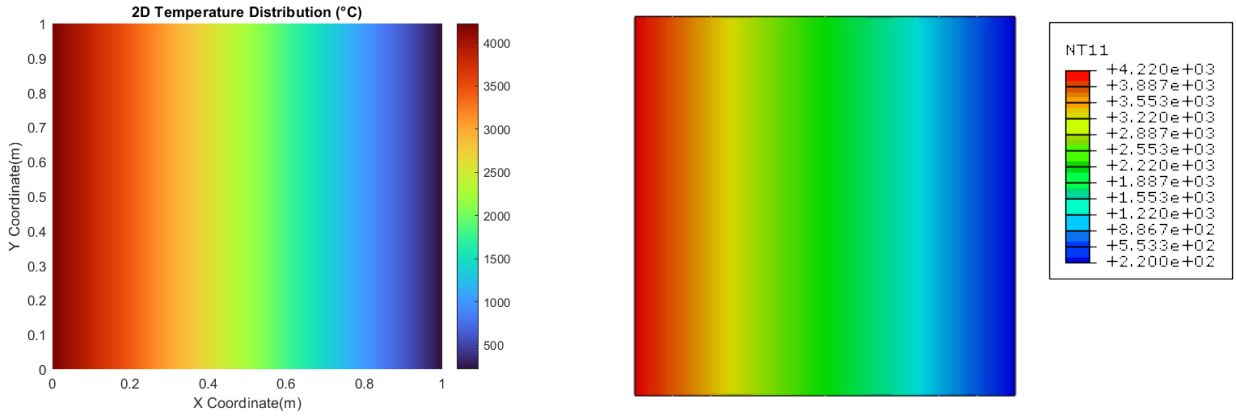
(a) (b)

Figure 20: Displacement 'u1' for loadcase=6 in (a) Matlab (b) Abaqus.



(a) (b)

Figure 21: Displacement 'u2' for loadcase=6 in (a) Matlab (b) Abaqus.



(a) (b)

Figure 22: Temperature distribution for loadcase=6 in (a) Matlab (b) Abaqus.

LoadCase = 7

Boundary Conditions:For the square domain with a length of 1 meter, the Thermal and Mechanical boundary conditions are shown in figure 23 [1]:

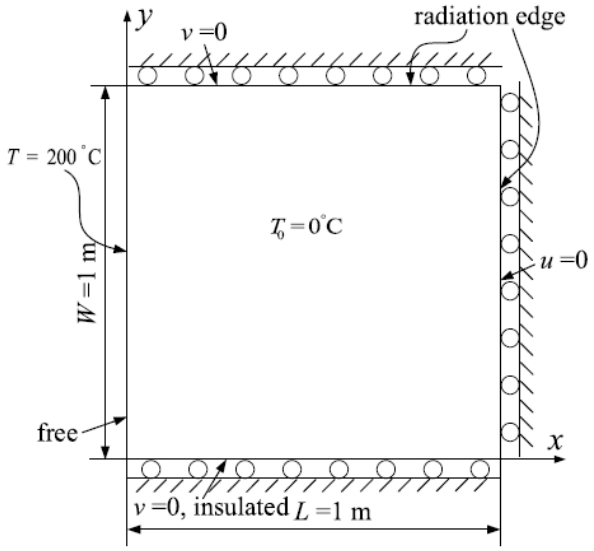
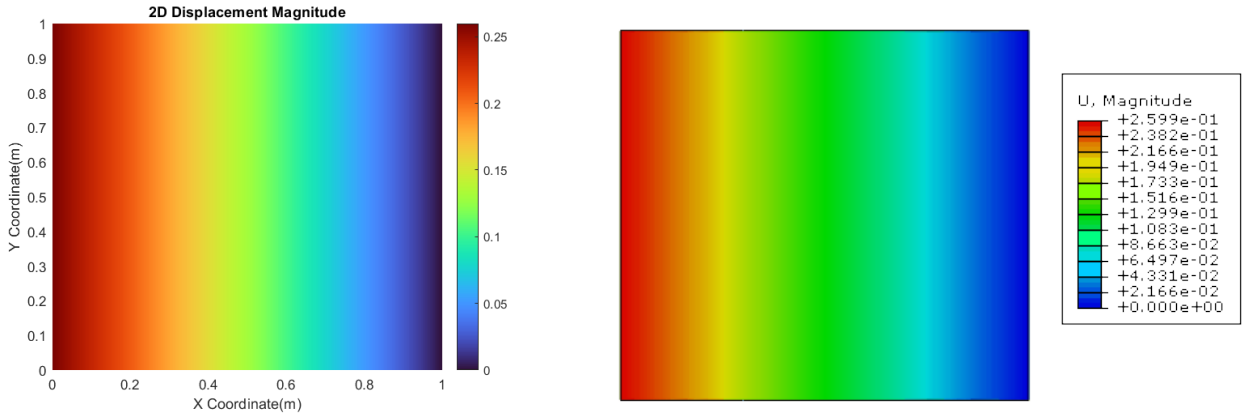


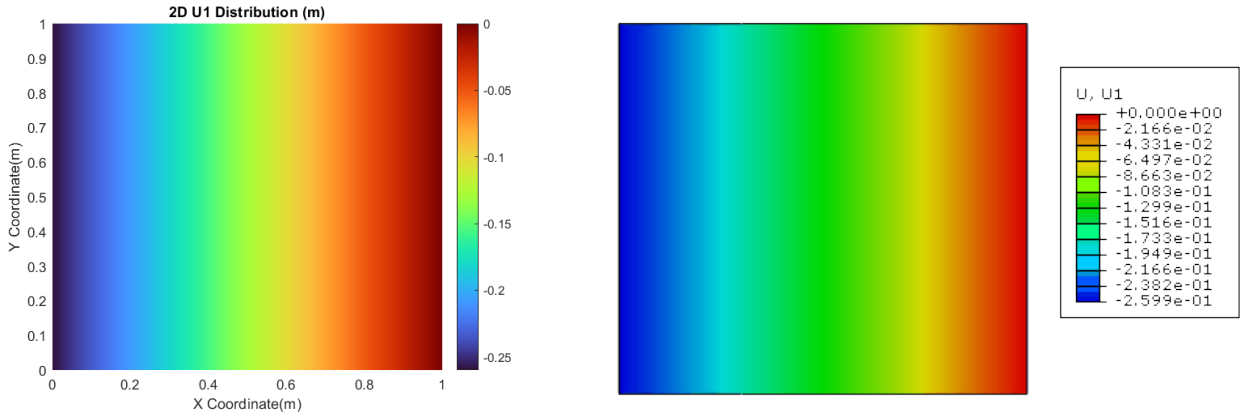
Figure 23: 2D sqaure plate with Boundary conditions

Material Properties: Thermal conductivity: $k = 400\,\mathrm{W/m^{\circ}C}$, Young's modulus: $E = 2\times10^{7}\,\mathrm{Pa}$, Poisson's ratio: $\nu = 0.3$, Thermal expansion: $\alpha = 0.001\,^{\circ}\mathrm{C}^{-1}$, Emissivity: $\epsilon = 0.5$, Stefan-Boltzmann :$\sigma = 5.67\times10^{-8}\,\mathrm{W/m^2\,K^4}$, Environmental temperature: $T_e = 20\,^{\circ}\mathrm{C}$ [1].
Procedure to run this test: Set testCase = 0, LoadCase = 7, eigenValueTest = 'no', numericalTesting = 'no', temperatureDependent = 'no', steadyState = 'yes' and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and and 2D contour plot results will be displayed.
Obtained results:The Matlab 2D contour plot results for this load case is verified with the Abaqus software 2D contour plot results with the above material parameters which are shown in the Figures 24 to 27.



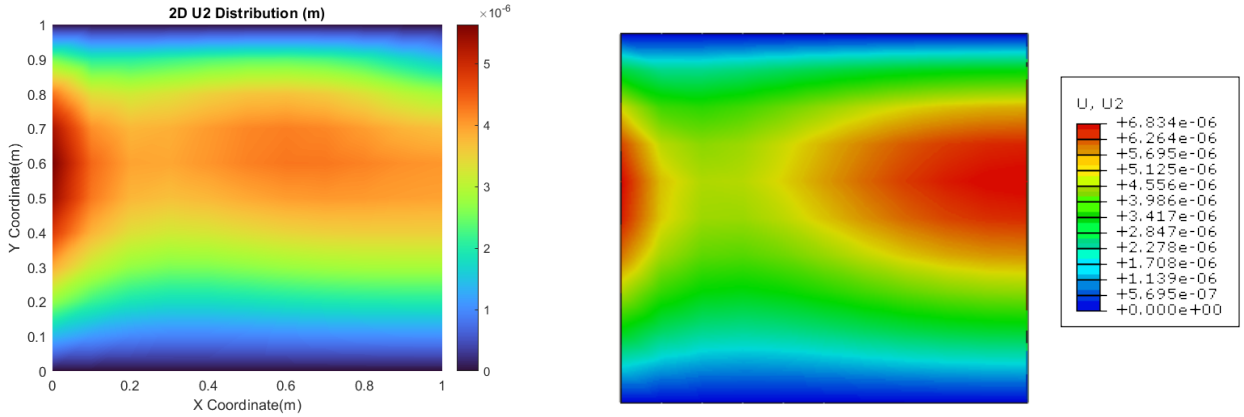
(a) (b)

Figure 24: Displacement Magnitude 'u' for loadcase=7 in (a) Matlab (b) Abaqus.



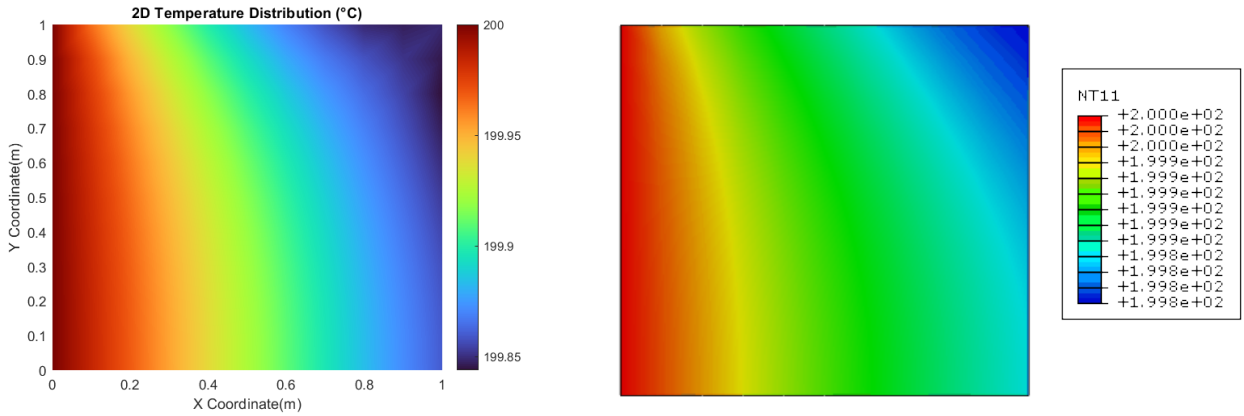
(a) (b)

Figure 25: Displacement 'u1' for loadcase=7 in (a) Matlab (b) Abaqus.



(a) (b)

Figure 26: Displacement 'u2' for loadcase=7 in (a) Matlab (b) Abaqus.



(a) (b)

Figure 27: Temperature distribution for loadcase=7 in (a) Matlab (b) Abaqus.

8 Results for Thermoelastic Fully Couple Material Model (With Temperature dependent Material properties)

The mesh generated for the square plate geometry is shown in Figure 3. Users can change element mesh sizes (h) with readFile = fopen('square100Element.inp', 'r') in mainCouplefem.m by using given square400Element.inp for square domain or any different inp files.
Result No = 1: Fully Coupled Steady State Analysis. Plain stress condition is assumed.
Boundary Conditions:Thermal and Mechanical boundary conditions are shown in figure 23:
Material Properties: Temperature dependent Material Properties are taken from reference [1]:
Thermal conductivity: $k(T) = (400 + 0.04T)\,\mathrm{W/m^{\circ}C}$,
Young's modulus: $E(T) = (2 \times 10^7 - 2 \times 10^4 T)\,\mathrm{Pa}$,
Poisson's ratio: $\nu = 0.3$,
Thermal expansion: $\alpha(T) = (0.001 - 0.000001T)\,^{\circ}\mathrm{C}^{-1}$,
Emissivity: $\epsilon = 0.5$,
Stefan-Boltzmann :$\sigma = 5.67 \times 10^{-8}\,\mathrm{W/m^2\,K^4}$,
Environmental temperature: $T_e = 20\,^{\circ}\mathrm{C}$.
Procedure to run this test: Set testCase = 0, LoadCase = 7, eigenValueTest = 'no', numericalTesting = 'no', temperatureDependent = 'yes', steadyState = 'yes', readFile = fopen('square100Element.inp', 'r') and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and 2D contour plot results will be displayed. Users can change the LoadCase numbers to see different results for different Thermal loads.
Obtained results:The Matlab 2D contour plot results for this load case which are shown in the Figures 28 to 31.

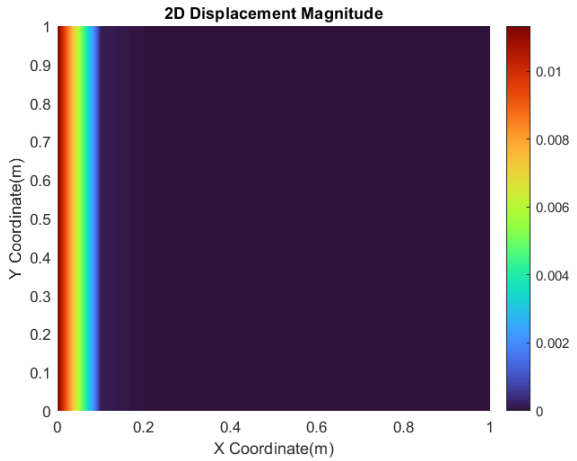


Figure 28: Displacement Magnitude 'u' for loadcase=7 in Matlab

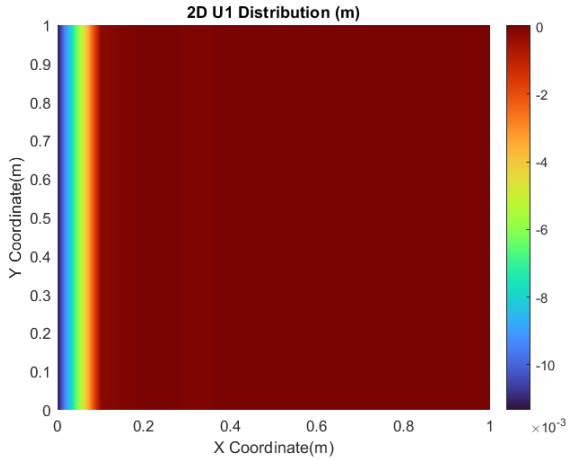


Figure 29: Displacement 'u1' for loadcase=7 in Matlab

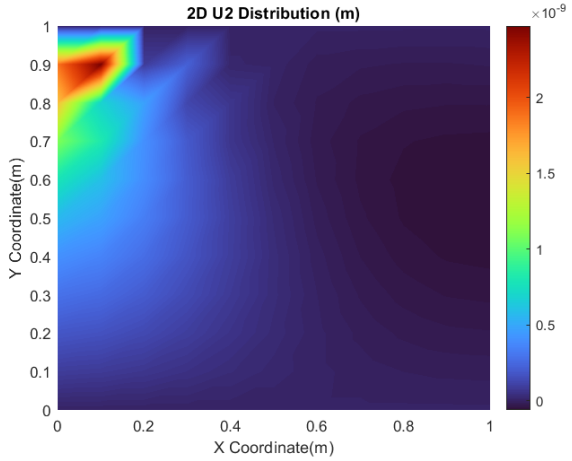


Figure 30: Displacement 'u2' for loadcase=7 in Matlab

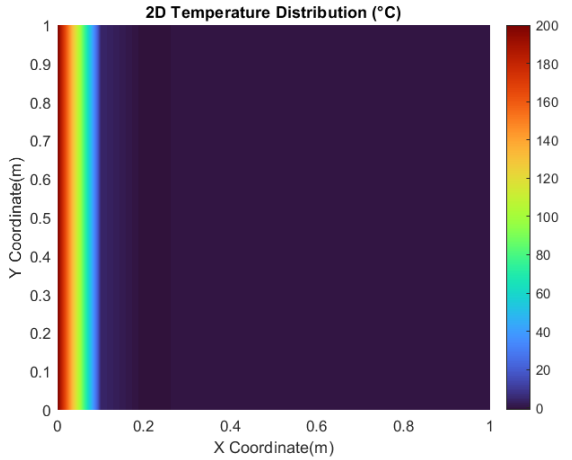


Figure 31: Temperature distribution for loadcase=7 in Matlab

Result No = 2: Fully Coupled Transient Analysis. Plain stress condition is assumed.
Boundary Conditions:Thermal and Mechanical boundary conditions are shown in figure 23:
Material Properties: Temperature dependent Material Properties are taken from reference [1]:
Thermal conductivity: $k(T) = (400+0.04T)\,\mathrm{W/m^{\circ}C}$, Density: $\rho(T) = (900+0.009T)\,\mathrm{kg/m^3}$, Specific Heat : $c(T) = (300+0.03T)\,\mathrm{J/kg^{\circ}C}$, Young's modulus: $E(T) = (2\times10^7-2\times10^4T)\,\mathrm{Pa}$, Poisson's ratio: $\nu = 0.3$, Thermal expansion: $\alpha(T) = (0.001 - 0.000001T)\,^{\circ}\mathrm{C}^{-1}$,
Emissivity: $\epsilon = 0.5$, Stefan-Boltzmann :$\sigma = 5.67 \times 10^{-8}\,\mathrm{W/m^2\,K^4}$, Environmental temperature: $T_e = 20\,^{\circ}\mathrm{C}$.
Procedure to run this test: Set testCase = 0, LoadCase = 7, eigenValueTest = 'no', numericalTesting = 'no', temperatureDependent = 'yes', steadyState = 'no', readFile = fopen('square100Element.inp', 'r') and then Run mainCouplefem.m in the command window then code will automatically consider the required parameters and 2D contour plot results will be displayed. Users can change the LoadCase numbers to see different results for different Thermal loads.
Obtained results:The Matlab 2D contour plot results for this load case which are shown in the Figures 32 to 34. Displacement variation with time on the left surface is show in Figure 35(a). And Temperature distribution for different time step along X direction is shown in Figure 35(b). The total computational solution time is set as 10000 s, and the time step of the solution is $\Delta t = 100\,\mathrm{s}$.

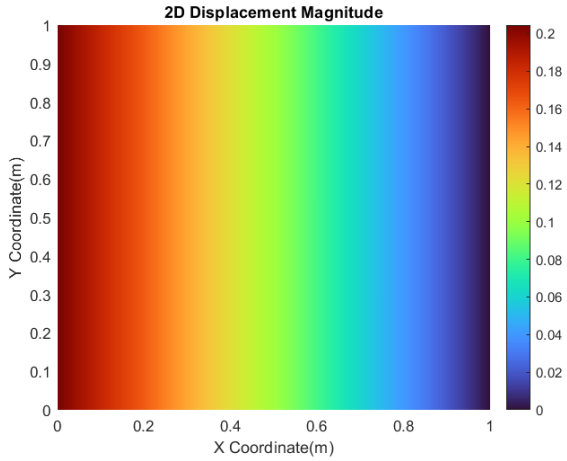
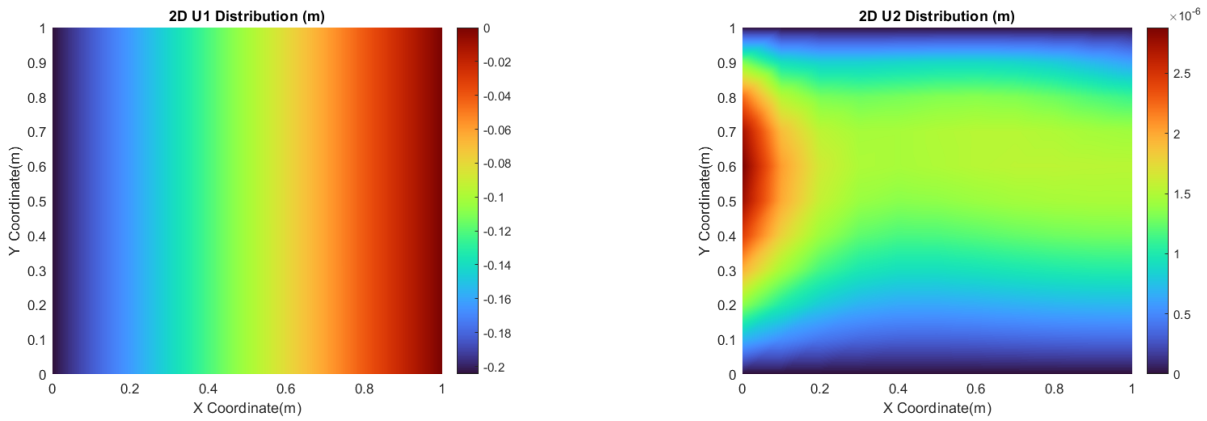


Figure 32: Displacement Magnitude 'u' for loadcase=7 (t = 1000s)



(a) (b)

Figure 33: Displacement for loadcase=7 in (a) u1 (b) u2 (t = 1000s)

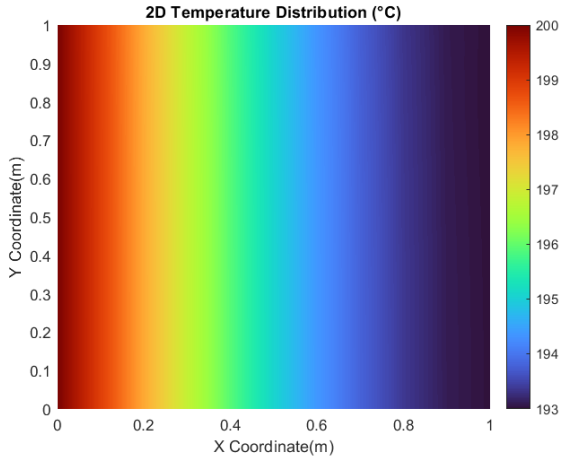
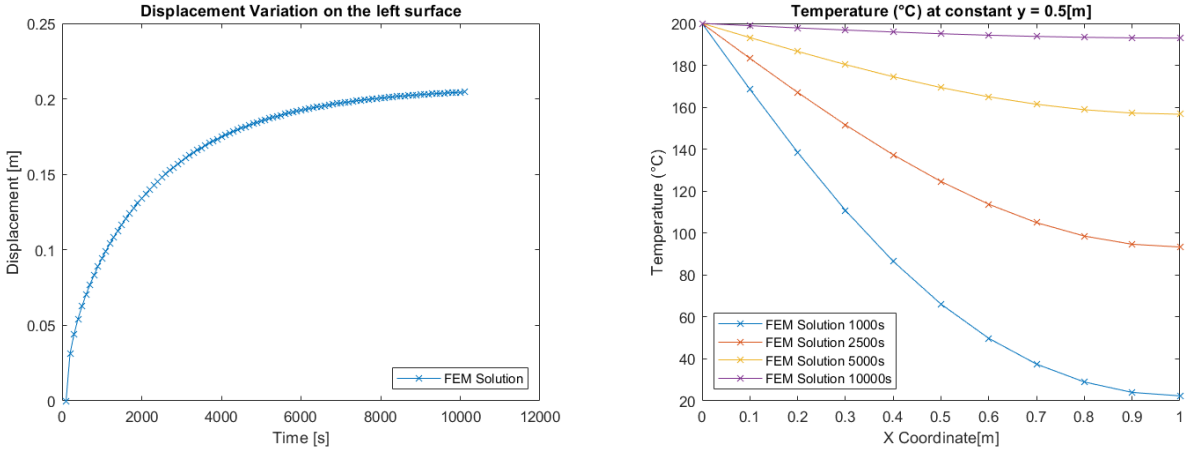


Figure 34: Temperature distribution for loadcase=7 (t = 1000s)



(a) Displacement variation with time (b) Temperature along X direction

Figure 35

8.1 Results for square plate with hole

The mesh generated for the square plate with hole geometry is shown in Figure 36. Consider

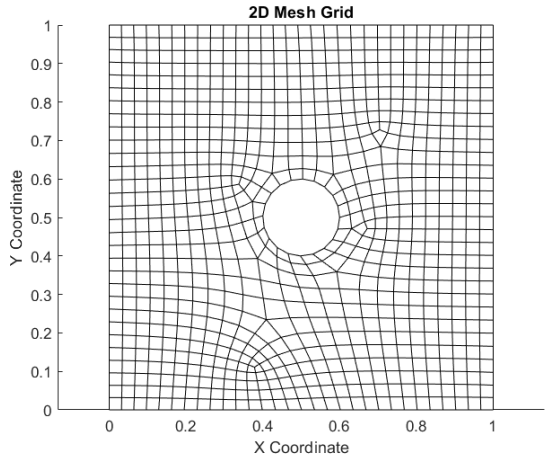
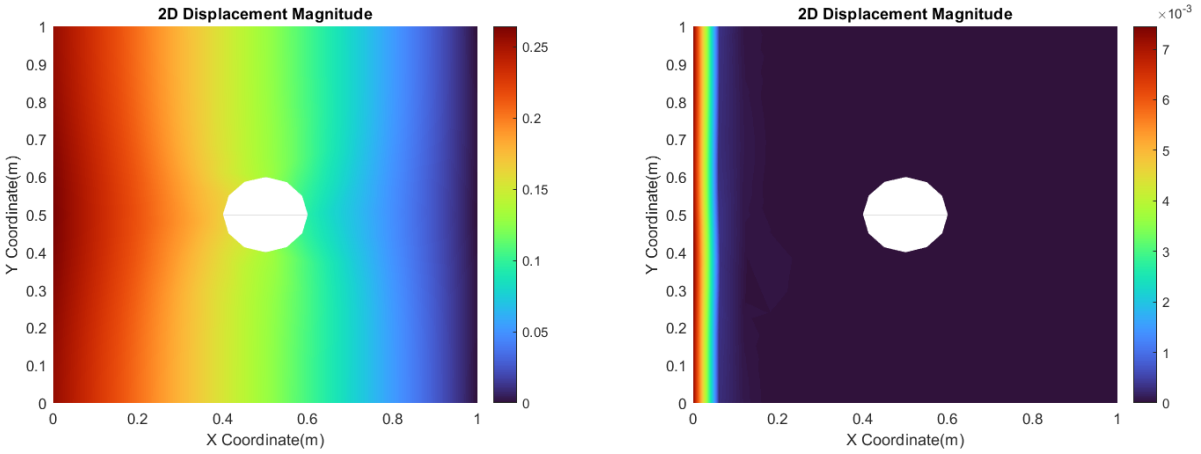


Figure 36: 2d Sqaure Plate with hole Mesh

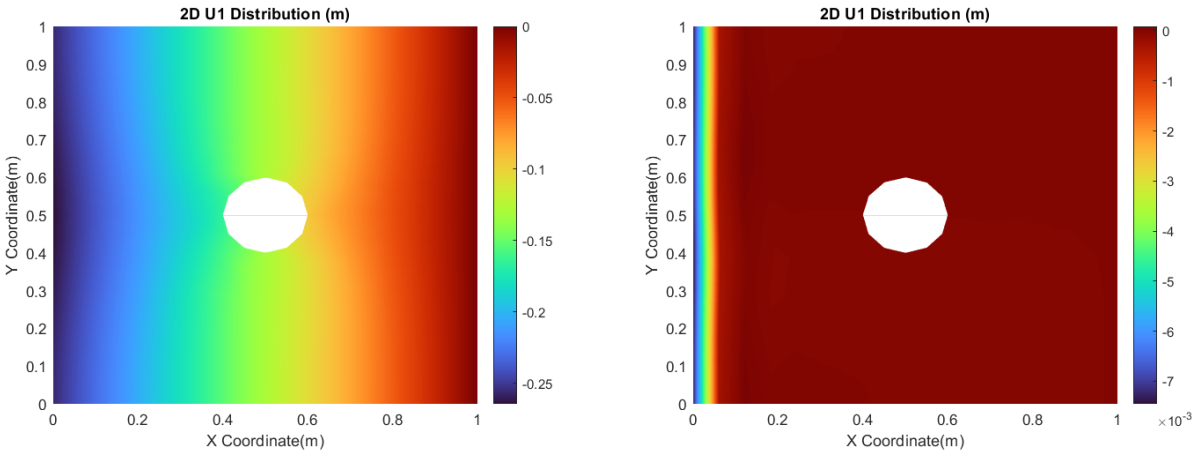
from Result no = 1 the same boundary conditions, Material property, and same procedure to run the code. Change readFile = fopen('squareCenterHoleFineMesh.inp', 'r') to read a mesh data for geometry. For material properties with not Temperature dependent set emperatureDependent = 'no'.

Obtained results: The Matlab 2D contour plot results for this load case which are shown in the Figures 37 to 40 with Material properties (a) not Temperature dependent (b) Temperature dependent



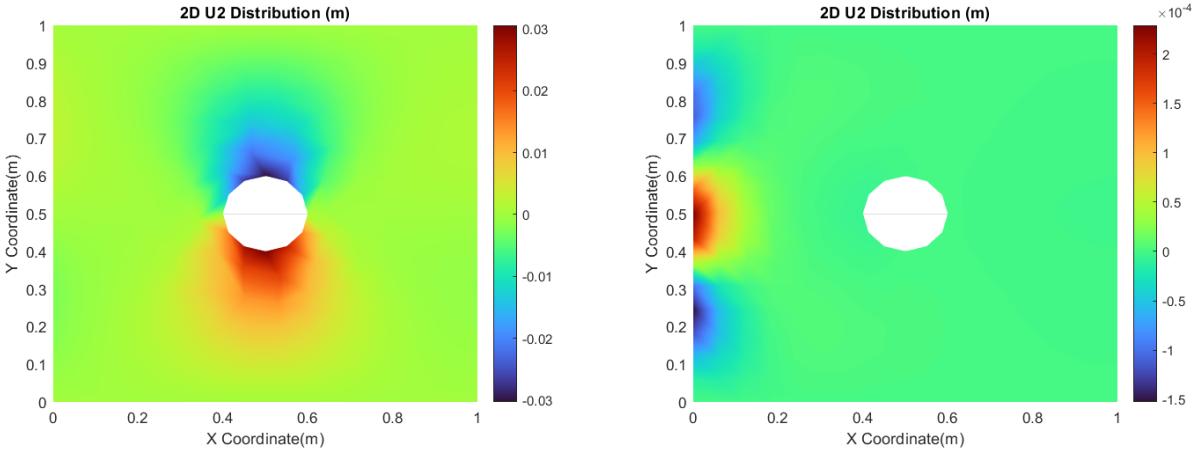
(a) (b)

Figure 37: Displacement Magnitude 'u' with Material properties (a) not Temperature dependent (b) Temperature dependent.



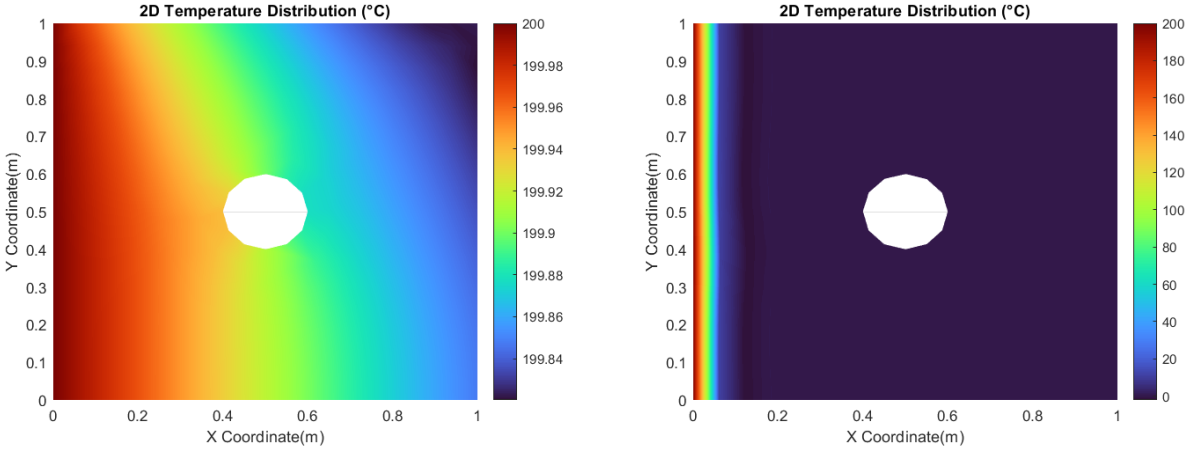
(a) (b)

Figure 38: Displacement 'u1' with Material properties (a) not Temperature dependent (b) Temperature dependent.



(a) (b)

Figure 39: Displacement 'u2' with Material properties (a) not Temperature dependent (b) Temperature dependent.



(a) (b)

Figure 40: Displacement 'u2' with Material properties (a) not Temperature dependent (b) Temperature dependent.

9 Manual

We need to keep all Matlab .m files and also .inp files in a single folder then type "mainCouplefem" in the Matlab command window for running the program. We must keep all of the .inp and.m files in the same location in order to perform various testing cases. The command for commenting a line in Matlab is 'ctrl+R', and for uncommenting a line is 'ctrl+T'.

9.1 Procedure for running the main program:

Depending on the issue that needs to be resolved, "mainCouplefem.m" runs through three different load cases with steady state analysis or not and with material properties dependent on temperature or not. The user must choose which load case to perform by changing the variable `LoadCase` value and mentioning "yes" or "no" for `steadyState` or `temperatureDependent` variables. We go over the three types of load cases in the "mainCouplefem.m" function below.
1) LoadCase: Different thermal load cases require different material properties and boundary conditions requirements, so the load case must be chosen based on the necessary material specifications and boundary conditions. For the variable `LoadCase`, users can set values 5 or 6 or 7.
Note: The variable `testCase` must be equal to zero. And variables `eigenValueTest`, `numericalTesting` must be equal to 'no'.
2) Temperature dependent: For the variable `temperatureDependent`, users can set values 'yes' or 'no'. If the user sets the value 'yes' then the code will consider material properties dependent on temperature and vice versa.
3) Steady state or Transient analysis: For the variable `steadyState`, users can set values 'yes' or 'no'. The code will take into account all necessary material attributes for transient analysis if the user sets the value to "yes," and vice versa.
4) Domain: The user has two types of domains available: a square domain and a square with a center hole domain. The user can select either of these domains by changing the `readFile` variable accordingly.
Note: The file `square100Element.inp` corresponds to the square domain, and the file `squareCenterHoleFineMesh.inp` corresponds to the square with a center hole domain.

5) Mesh: The user can increase the density of the mesh by utilizing the `readFile` variable to access different .inp files available.
Note: The file name (e.g., `square100Element.inp`) indicates the number of elements in the mesh. For instance, `square100Element.inp` corresponds to a mesh with 100 elements or `squareCenterHoleCoarseMesh.inp` corresponds to a Coarse mesh.
Procedure for viewing the results: The 2D contour plots will appear on the desktop after entering "mainCouplefem" in the Matlab command window.
For every load case, the following contour plots are visible:
1) Displacements in X and Y direction i.e. U1 and U2 respectively.
2) Magnitude of Displacements.
3) Temperature distribution,
Example: (Sqaure domain for load case 7 and steady state analysis without temperature dependent material properties.)
`readFile` = fopen('square100Element.inp', 'r')
`testCase` = 0
`LoadCase` = 7
`eigenValueTest` = 'no'
`numericalTesting` = 'no'

`temperatureDependent` = 'no'
`steadyState` = 'yes'

9.1.1 Procedure for running the test cases:

1) Patch test:
`testCase` = 3 will perform a patch test. The code will then automatically consider nodal coordinates, element connectivity, and required material parameters.
2) Exact Solution test:
`testCase` = 1 or 2 or 4 will perform the exact solution test. Users can increase mesh density by considering different .inp files for square domain.
Note: The variable `Loadcase` must be equal to zero. And variables `eigenValueTest`, `numericalTesting`, `temperatureDependent` must be equal to 'no'. Variable `steadyState` must be equal to 'yes' for patch test and Exact Solution test.
Should only select square domain which is `readFile` = fopen('square100Element.inp', 'r').
3) Numerical testing: If variable `numericalTesting` = 'yes' will perform all numerical test cases for the given code. The code will then automatically consider nodal coordinates, and element connectivity for a unit square domain.
Note: The variable `Loadcase` must be equal to zero, `testCase` must be equal to 1. And variables `eigenValueTest`, `temperatureDependent` must be equal to 'no'. Variable `steadyState` must be equal to 'yes'.
4) Eigen value Test: If variable `eigenValueTest` = 'yes' will perform zero eigen value test for the given code.
Note: The variable `Loadcase` must be equal to zero, `testCase` must be equal to 1. And variables `numericalTesting`, `temperatureDependent` must be equal to 'no'. Variable `steadyState` must be equal to 'yes'.
Should only select square domain with 4 Element which is `readFile` = fopen('square4Element.inp', 'r') since finding eigen values for more numbers of elements will be computational costly process.
Procedure for viewing the results: All the results will visible on Matlab command window.

10 Conclusion

It concludes that this approach can solve fully coupled nonlinear thermoelastic finite element analysis problems for various thermal loads. The temperature and displacement results are accurate enough for engineering applications.
An extension of this material model to thermoplastic or thermo-elastic-plastic coupled material models is also practical. Additionally, the code can be expanded to include bimetallic materials.

11 Literature

[**1**] ZhihuiLiu, ZhihuiLi , Qiang Ma. "Nonlinear finite element algorithm for solving fully coupled thermomechanical problems under strong aerothermodynamic environment", Acta Astronautica, 2023.

[**2**] Collins O. Akeremale,Olusegun A Olaij,Yeak Su Hoe, "Finite Element Method for Heat Transfer Phenomenon on a Closed Rectangular Plate",EUREKA,2020.

[**3**] Jan Taler, Pawel Oclon."Finite Element Method in Steady-State and Transient Heat Conduction". Institute of Thermal Power Engineering, Faculty of Mechanical Engineering, Cracow University of Technology, Cracow, Poland.2013

[**4**] 1-d Coupled Thermo-mechanics Finite Element Code from Abdullah Waseem. (WebLink)

[**5**] The Plane Stress Problem from Siva Srinivas Kolukula. (WebLink)

[**6**] Dr.-Ing. Stefan Prüger. "Non-Linear Finite element Methods Lectures",Technische universität Bergakademie Freiberg. 2022

[**7**] Abaqus Heat transfer Patch Test Element type "DC2D4". (WebLink)

[**8**] Validation manual from "SIMSolid" for material properties. (WebLink)

12 Git Log

commit a27e5a900712cce5c39c041e0a2f790586629531 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Oct 23 19:34:25 2024 +0200

Numerical Testing.

commit 4dfe689e46f59015ff0ba62be5ef3b6bbc19b1b7 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Oct 23 19:32:33 2024 +0200

Eigen Value test for thermal element routine.

commit 8b20fb4abd0e813807858eb28f767282d0f68d67 Author: visho ¡visho@Vishal-Pc¿ Date: Sat Oct 5 00:21:51 2024 +0530

Radiation boundary condition testCase 7.

commit a0bce443f8128dc753e78336bbee1cc461006ea2 Author: visho ¡visho@Vishal-Pc¿ Date: Sat Oct 5 00:19:36 2024 +0530

Temperature dependent density and specific heat.

commit b609d5f96dac4d26d85469148330965ff3a5bd41 Author: visho ¡visho@Vishal-Pc¿ Date: Sat Oct 5 00:18:22 2024 +0530

Radition boundary condition testcase 7.

commit 14c1e517ab482fbab9115227a89323ccadf4a9e4 Author: visho ¡visho@Vishal-Pc¿ Date: Fri Oct 4 17:15:38 2024 +0530

Multiply shape with alphaDeri.

commit fed41129854b48444f95ec9e7a56b7593bf1b156 Author: visho ¡visho@Vishal-Pc¿ Date: Fri Oct 4 13:20:29 2024 +0530

File name change.

commit e9aba492a6af1b7ae60148f68c0a75b5442881d2 Author: visho ¡visho@Vishal-Pc¿ Date: Fri Oct 4 13:18:59 2024 +0530

Change of "if condition" for left and right boundary node.

commit d530e30a98ab73f7516239932eec63afb6b6165b Author: visho ¡visho@Vishal-Pc¿ Date: Fri Oct 4 12:23:04 2024 +0530

Change of file name.

commit 67e886891e338f57096a3e74ccd228fcc95553e3 Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 30 19:36:53 2024 +0530

Testing of code for shape function and B matrix.

commit 637c7fdde5a0e13ea01d43764b7dbf590928d41d Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 30 19:31:02 2024 +0530

Delete file

commit 6d707fbbece454b0198d1a951909d1b574b084a0 Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 30 19:29:25 2024 +0530

Change of file name.

commit ebdbaeb0a08a33696d2f3e87217dfbddfc4ddab3 Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 30 15:47:20 2024 +0530

Function to calculate B matrix.

commit 4d1ba5a24defba82b812471d5d4c9751eb0b7dd6 Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 30 15:46:00 2024 +0530

Calling B matrix function.

commit 7e9f431aeec1dd32169c8194c6feb26f97acee99 Author: visho ¡visho@Vishal-Pc¿ Date: Sun Sep 29 21:08:49 2024 +0530

Final test for Temperature field. Test case 1,2,3,4.

commit 40168aa1c3dd7e8832f792a07bb3aeeebd6b4233 Author: visho ¡visho@Vishal-Pc¿ Date: Sun Sep 29 21:03:22 2024 +0530

Change of test case number.

commit 2c083d88ed55cd83d9e6807f9338f94bcbfe2c05 Author: visho ¡visho@Vishal-Pc¿ Date: Sun Sep 29 21:02:42 2024 +0530

Change of title.

commit 0337894a0ffeee4951dbe5451ec4d7fc7ddc7413 Author: visho ¡visho@Vishal-Pc¿ Date: Sun Sep 29 17:03:23 2024 +0530

This function is not needed.

commit 7635a82674a1c9733b81dc8e5008519a1958b04a Author: visho ¡visho@Vishal-Pc¿ Date: Thu Sep 26 20:21:16 2024 +0530

Solved Implicit problem for test case 4 and 5.

commit 0e7bc4715e5de87a592db3dae4fb4f51db3ea8b5 Author: visho ¡visho@Vishal-Pc¿ Date: Thu Sep 26 20:12:30 2024 +0530

Change variable name of Temptime.

commit 976f0908cf6182caefc834f424c30d559fb9a2d5 Author: visho ¡visho@Vishal-Pc¿ Date: Thu Sep 26 20:11:31 2024 +0530

Temperature dependent material property.

commit 830b02a6b24ff5a37524c04c8df54c0ef801d48c Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 25 22:35:35 2024 +0530

Removal of Temperature distribution plot.

commit 57cc83abdcecd332c961f4594d217b0ca7815a2f Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 25 22:33:02 2024 +0530

2d Time step nodal Temperature visualization.

commit b5e66f180cb2cef26cb131b4e1b43516c2fbefe9 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 25 22:31:57 2024 +0530

2d Nodal Temperature visualization.

commit 9d4bd321f568963375d574f90b20c1208b9ce9e4 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 25 21:49:14 2024 +0530

Implicit Time integration for Test case 1,2,3.

commit 432a3b154c20bc3875a878354ec0e5591e0fc02a Author: visho ¡visho@Vishal-Pc¿ Date: Sat Sep 21 21:09:32 2024 +0530

Separation of test Case 4 and 5 from other test cases.

commit 02cbd2effbf3da98530139351f4f8a052f565027 Author: visho ¡visho@Vishal-Pc¿ Date: Sat Sep 21 21:06:29 2024 +0530

Test case 4 and 5 is separated from other test case.

commit 7f4ae2001efbb7e32c3025e76a570884db697bf4 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 18 20:10:12 2024 +0530

Test 4 and 5 using temperature dependent material properties.

commit 5f85c062e38b03a5ec444d2ca5075ccf59a825b4 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 18 19:55:28 2024 +0530

Calling temperature dependent mechanical material routine.

commit bccc6e5df422ad1cc43ecdb89e445f7a93da7ed8 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 18 19:54:24 2024 +0530

Calling temperature dependent mechanical material routine.

commit 2050e6cff82b62f1236a9c0e2047771786d001d8 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 18 19:53:02 2024 +0530

Calling temperature dependent mechanical material routine .

commit ef4ee3ab9f52ba88739682f3c91513c32746f3fc Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 18 19:51:14 2024 +0530

Temperature dependent Young's modulus and thermal expansion.

commit 04cd1389a424abc025b4a139bf05c2939f32af40 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 18 19:49:33 2024 +0530

Material routine for temperature dependent conductivity.

commit 558d84cd0eafc7ef5e75cac8c222541fff64af08 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 18 19:48:19 2024 +0530

Thermal conductivity dependent on temperature.

commit b0c8ec93c5d03ff19e78e5030b77e3667816ebb2 Author: visho ¡visho@Vishal-Pc¿ Date: Tue Sep 17 21:43:14 2024 +0530

Element routine for thermal Mechanical couple.

commit bad8696878e2d0961268c7d6c19936a54d4cb8e8 Author: visho ¡visho@Vishal-Pc¿ Date: Thu Sep 12 15:49:48 2024 +0530

Test case 5 couple problem with heat flux and convection boundary condition.

commit 5797792e5d913ad71c07793f99fcf7c94570ac0b Author: visho ¡visho@Vishal-Pc¿ Date: Thu Sep 12 15:40:56 2024 +0530

Added Test case 5 in if condtion.

commit 7f5117b8ef52235d34f8178127781d546df8b0fd Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 11 19:01:11 2024 +0530

Removing test case for temperature dependence thermal conductivity.

commit 70460ea3c3436a534e945841d0f69242a56d426b Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 11 18:59:40 2024 +0530

Comment test cases for temperature dependence Thermal conductivity.

commit 45d989583a64e533d4556b2c38c6642c61470f37 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 11 18:43:23 2024 +0530

Calling gaussPointsWeights function.

commit c0be01d126c666ac4d2dd9b864921a592e090127 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 11 18:42:25 2024 +0530

Function for Gauss points and weights.

commit 0f0a22beae5baa01f3916d30b41f5b2c8f9fef5b Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 11 15:09:29 2024 +0530

Final full Thermomech coupled problem solved.

commit a6cbcd65e9fa82df22c709f4829887472720802f Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 11 15:06:53 2024 +0530

Multiplying D matrix into stiffness matrix.

commit 1ab791bc6c4f7c5b819394b2bfd8450f8e28b414 Author: visho ¡visho@Vishal-Pc¿ Date: Tue Sep 10 14:52:59 2024 +0530

Removal of D matrix.

commit 7097695276e751dae09e6d69ca9b8a85258b8a17 Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 9 21:01:41 2024 +0530

Addition of ThermoMech couple code.

commit c6f89da227d4eb2b397d3760ebb575149bec804f Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 9 21:00:10 2024 +0530

Test Case 6

commit 83161a0df336518670fd4aa0e0a631de65b6a1d8 Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 9 20:59:27 2024 +0530

Element routine for Mechanical Thermal couple.

commit fee7c44ad6918751d0b60f09ffa4b635d549934c Author: visho ¡visho@Vishal-Pc¿ Date: Sat Sep 7 15:22:39 2024 +0530

Material Routine for displacement field.

commit a83e7a8785b0cfe0c9ad615fdedbb70872d355a8 Author: visho ¡visho@Vishal-Pc¿ Date: Sat Sep 7 15:21:47 2024 +0530

Element routine for displacement fieid.

commit d2c6c8621ab87825e8146e39101c27f6416cd14f Author: visho ¡visho@Vishal-Pc¿ Date: Thu Sep 5 17:15:38 2024 +0530

Sparse for Stiffness Matrix.

commit d822d539bd98baa363ff41ce49c21e4e9ecf78e9 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 4 16:07:11 2024 +0530

Covection boundary condition testCase =5.

commit 627ed5abbfd2d15a9947485c99137da63b103331 Author: visho ¡visho@Vishal-Pc¿ Date: Wed Sep 4 16:06:03 2024 +0530

Convection Stiffness matrix.

commit 0a9d8f64c148f5b7ff5186356ac0e5b9ab09108c Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 2 22:35:33 2024 +0530

Material routine for Thermal FEM.

commit 96849ead0aee70a9c19073c13634449b0a23bace Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 2 22:34:40 2024 +0530

2 Cases for Thermal conductivity temperature dependence.

commit eba323cd299ea75a860ff1a7dd101d59920effbe Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 2 22:33:33 2024 +0530

Temperature depend Thermal conductivity.

commit 69122172f02fd2b8e48b38c6e11e764bb8e08996 Author: visho ¡visho@Vishal-Pc¿ Date: Mon Sep 2 12:09:14 2024 +0530

Addition of second test case.

commit 7c6404de0a8d4166177a407025aa2d6cee049bcb Author: visho ¡visho@Vishal-Pc¿ Date: Sun Sep 1 22:07:30 2024 +0530

Post processing of Temperature distribution.

commit 79cd593e8a4458376a6be3f1067ff5a0218a33e4 Author: visho ¡visho@Vishal-Pc¿ Date: Sun Sep 1 22:06:42 2024 +0530

Plot Mesh

commit 62c330dd852ed7494031e3375e07ca0b3b1ef380 Author: visho ¡visho@Vishal-Pc¿ Date: Sun Sep 1 17:39:16 2024 +0530

Newton Raphson loop added.

commit c2eeddb4b81f3ab27480516f907a65ac222f24e5 Author: visho ¡visho@Vishal-Pc¿ Date: Sun Sep 1 13:49:54 2024 +0530

Addition of global Residual.

commit d0144ce6b71caa04076f5a0db3e5844b8d1fb463 Author: visho ¡visho@Vishal-Pc¿ Date: Sun Sep 1 13:48:53 2024 +0530

Calculating Residual for element.

commit 9ce87717e1ef734c9addd78a6e0e38528028542d Author: visho ¡visho@Vishal-Pc¿ Date: Sat Aug 31 22:53:19 2024 +0530

Square domain with 4 elements INP file.

commit 30ad3c9ecdf1150944a1063336e60e13700f6081 Author: visho ¡visho@Vishal-Pc¿ Date: Sat Aug 31 22:51:58 2024 +0530

Element routine for Thermal FEM.

commit 6e2d77f9c7a2b3286622676b3b96b1bcdb6f5941 Author: visho ¡visho@Vishal-Pc¿ Date: Sat Aug 31 22:48:24 2024 +0530

Shape functions and It's derivatives.

commit 0d1c5c66f22be5a9d11e3f0c1239f8f6f66716ba Author: visho ¡visho@Vishal-Pc¿ Date: Sat Aug 31 22:47:15 2024 +0530

Extract data from .inp file.

commit 2a1471a5f5bdecdd1d3fa48f688d80b09aa337f0 Author: visho ¡visho@Vishal-Pc¿ Date: Sat Aug 31 22:45:34 2024 +0530

Thermal FEM main file

commit ab6186fae9661854550aa20bdeb1420575ea8846 Author: vishal soni ¡sonivishal199@gmail.com¿ Date: Sat Aug 31 17:03:04 2024 +0000

Initial commit