Pandit Deendayal Energy University
Academic Year 2023-2024
Computational Engineering Laboratory

PRESENTATION ON MATLAB CASE STUDY

Presented by:

1. Vora Khyaat Darshan 21BME118D
2. Narinder Singh Bhadwal 21BME006

Presented To:

Dr. Annirudh Kulkarni
Assistant Professor
Department of Mechanical Engineering
Pandit Deendayal Energy University

Outline/Content

- **Case Study 1**
- **Problem Statement**
- **Derivations**
- **Matlab Code**
- **Results**

- **Case Study 2**
- **Problem Statement**
- **Derivations**
- **Matlab Code**
- **Results**

Case Study 1

Problem Statement

To Solve a problem on Heat Transfer of Natural Convection in H – Shaped Cavity.

Derivations

Case Study -1 Equations

* The derivation ~~the~~ the equations we need :-

1) Governing equations :-

$\nabla \cdot V = 0$

Its represents conservation of mass.

2) Navier Stokes equation :-

$$\rho \left(\frac{\partial v}{\partial t} + V \cdot \nabla v \right)$$

$$= -\nabla p + \mu \nabla^2 v + \rho g.$$

It's represents the Conservation Momentum

3) Energy equations :-

$$\rho c_p \left(\frac{\partial T}{\partial t} + V \cdot \nabla T \right) = k \nabla^2 T$$

It's represents the Conservation of energy.

Assumptions on Equation

1. **Steady State Conduction : d/dt=0**
2. **Incompressible Flow : $\nabla . v = 0$**
3. **Boussinesq approximation (Density Variation with Compression): $\delta = \delta 0\left(1 - \beta(T - T0)\right)$**
4. **Negligible viscous Dissipation : Neglecting the Various terms $\mu \nabla^2 v$**
5. **Constant Fluid Properties : δ, μ, k and Cp are Constant within the Domain**

Matlab Code

```
title('Temperature Distribution in H-Shaped Cavity');
% Parameters
L = 1; % Length of cavity
W = 1; % Width of cavity
H = 0.1; % Height of cavity
T_hot = 100; % Temperature of hot wall
T_cold = 0; % Temperature of cold walls
nx = 51; % Number of grid points in x-direction
ny = 51; % Number of grid points in y-direction
dx = L/(nx-1); % Grid spacing in x-direction
dy = W/(ny-1); % Grid spacing in y-direction
max_iter = 1000; % Maximum number of iterations
tolerance = 1e-5; % Convergence tolerance

% Initialization
T = ones(nx, ny) * T_cold; % Initialize temperature array

% Main loop (iterate until convergence)
for iter = 1:max_iter
    % Boundary conditions
    T(:,1) = T_cold; % Left cold wall
    T(:,end) = T_cold; % Right cold wall
    T(1,:) = T_hot; % Hot wall

    % Compute temperature field using simple averaging
    T_new = T;
    for i = 2:nx-1
        for j = 2:ny-1
            % Average of neighboring points
            T_new(i,j) = 0.25*(T(i+1,j) + T(i-1,j) + T(i,j+1) + T(i,j-1));
        end
    end

    % Check for convergence
```

```
    if max(abs(T_new(:) - T(:))) < tolerance
        disp(['Converged at iteration ', num2str(iter)]);
        break;
    end

    % Update temperature array
    T = T_new;
end

% Plotting the results
[X, Y] = meshgrid(linspace(0,L,nx), linspace(0,W,ny));
contourf(X, Y, T', 50, 'LineColor', 'none');
colorbar;
xlabel('x');
ylabel('y');
```

Results

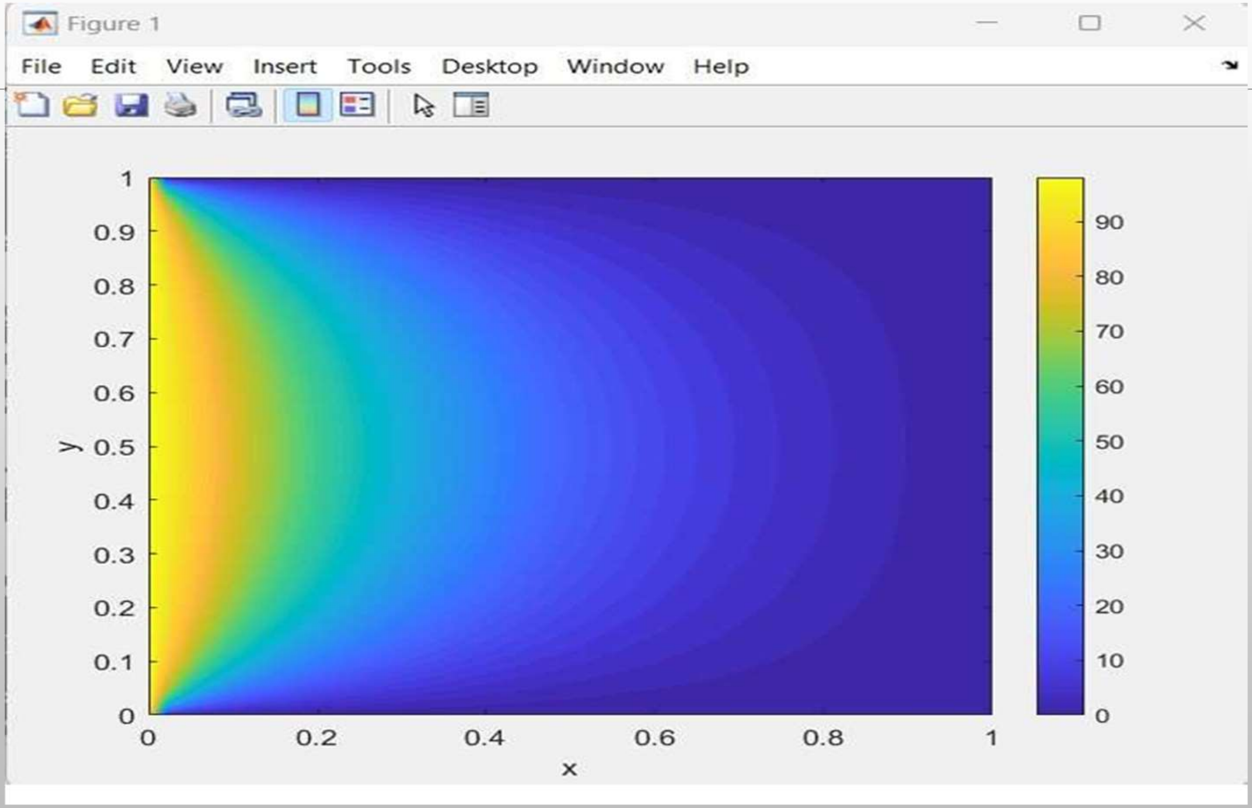


Figure 1
File Edit View Insert Tools Desktop Window Help
y
x
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
0 0.2 0.4 0.6 0.8 1
0 10 20 30 40 50 60 70 80 90

Case Study 2

Problem Statement

To Update the code on the problem on Heat Transfer of Natural Convection in H – Shaped Cavity.

Derivations

Case Study 2 Equations

* To derive the equations :-

⇒ Governing equations :-

Start with the continuity equations :-

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$$

By using Navier-Stokes equations (momentum equation):

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + g_x$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + g_y$$

Then,

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \alpha\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

Matlab Code

```
% Parameters
L = 1;          % Length of cavity
W = 1;          % Width of cavity
H = 0.1;        % Height of cavity
T_hot = 100;    % Temperature of hot wall
T_cold = 0;     % Temperature of cold walls
nx = 51;        % Number of grid points in x-direction
ny = 51;        % Number of grid points in y-direction
dx = L/(nx-1);  % Grid spacing in x-direction
dy = W/(ny-1);  % Grid spacing in y-direction
max_iter = 100; % Maximum number of iterations
tolerance = 1e-5;% Convergence tolerance
alpha = 0.1;    % Thermal diffusivity

% Initialize temperature array
T = ones(nx, ny) * T_cold;

% Set up VideoWriter object

writerObj = VideoWriter('temperature_evolution.avi'); % Specify the file name
writerObj.FrameRate = 10; % Set the frame rate

% Open the video writer object
open(writerObj);

% Create a figure for animation
figure;

% Main loop (iterate until convergence)
for iter = 1:max_iter
    % Boundary conditions
    T(:,1) = T_cold;            % Left cold wall
    T(:,end) = T_cold;          % Right cold wall
    T(1,:) = T_hot;             % Hot wall
    T(ceil(end/2-round(H/(2*dy))):floor(end/2+round(H/(2*dy))), end) = T_cold; % Middle cold wall

    % Compute temperature field using finite difference method
    T_new = T;
    for i = 2:nx-1
        for j = 2:ny-1
            % Finite difference equation
```

Matlab Code

```
            T_new(i,j) = T(i,j) + alpha * ( (T(i+1,j) - 2*T(i,j) + T(i-1,j))/dx^2 + (T(i,j+1) - 2*T(i,j) + T(i,j-1))/dy^2 );
        end
    end

    % Check for convergence
    if max(abs(T_new(:) - T(:))) < tolerance
        disp(['Converged at iteration ', num2str(iter)]);
        break;
    end

    % Update temperature array
    T = T_new;

    % Plot the current temperature distribution in 3D
    [X, Y] = meshgrid(linspace(0,L,nx), linspace(0,W,ny));
    surf(X, Y, T', 'EdgeColor', 'none');
    colorbar;
    xlabel('x');
    title(['Iteration: ', num2str(iter)]);
    axis([0 L 0 W 0 100]); % Set axis limits for consistency

    % Capture the frame
    frame = getframe(gcf);

    % Write the frame to the video file
    writeVideo(writerObj, frame);
end

% Close the video writer object
close(writerObj);
```

Results

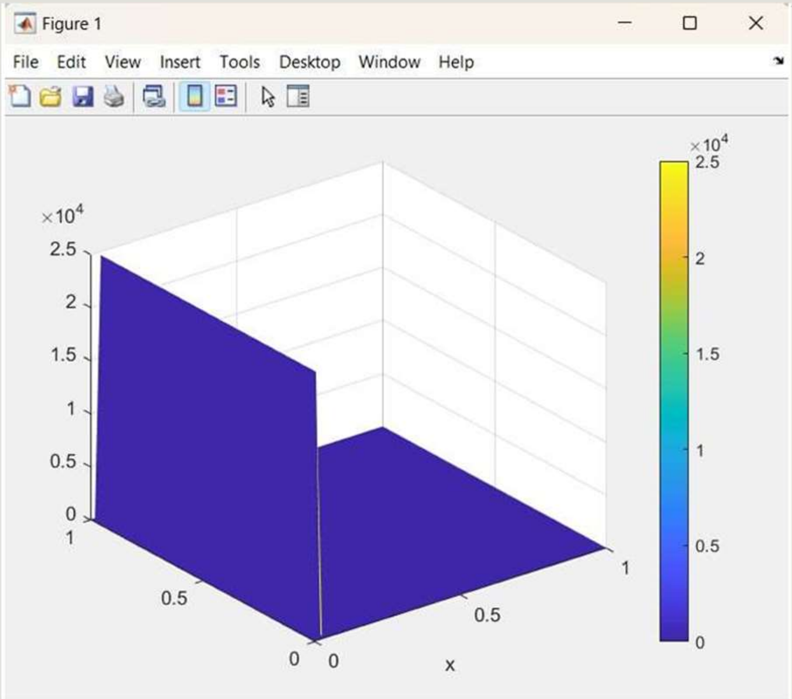
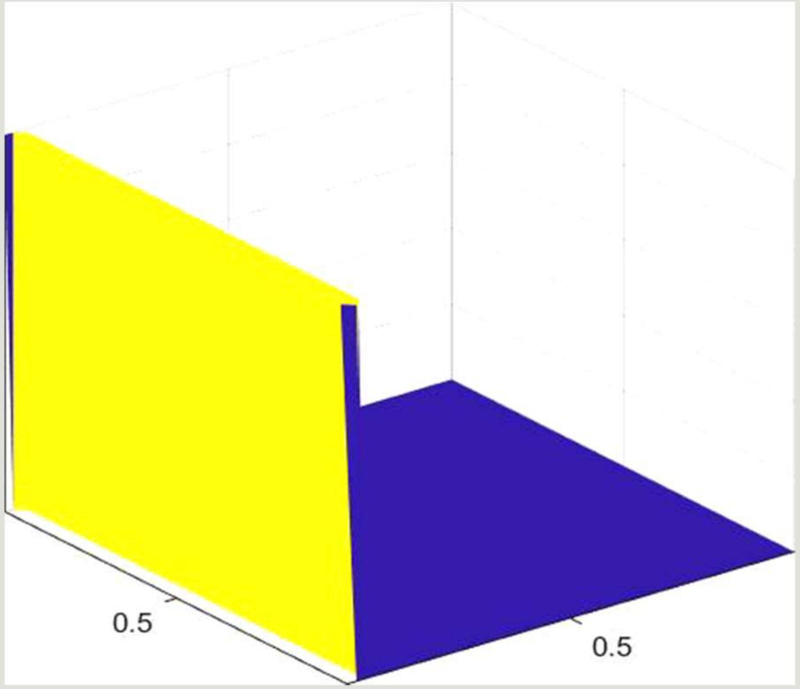


Figure 1
File Edit View Insert Tools Desktop Window Help
×10^4
2.5
2
1.5
1
0.5
0
1
0.5
0 0
0.5
1
x
×10^4
2.5
2
1.5
1
0.5
0



0.5
0.5

Thank You