Global Journal of Sciences
Volume 2, Issue 2, 2025, Pages 61-85
DOI: https://doi.org/10.48165/gjs.2025.2206; ISSN: 3049-0456
https://acspublisher.com/journals/index.php/gjs

Research Article

FEM Approach for Efficient Solution of Two-Dimensional Navier-Stokes Equations

Haider Ali[1*], Muhammad Amjad[1], Tehreem[2], Shah Jahan[3], Muhammad Zaman[3], Aamir Hussain Khan[4], Muhammad Ramzan[1]

1. *Department of Mathematics, COMSATS University Islamabad, Vehari Campus, Vehari 61100, Pakistan.*
2. *Department of Mathematics, Faculty of Basic and Applied Sciences, Air University, Islamabad 44000, Pakistan.*
3. *Institute of Mathematics, Khwaja Fareed University of Engineering & Information Technology, Rahim Yar Khan, Pakistan.*
4. *Department of Mathematics, Thal University, Bhakkar, Pakistan.*

ARTICLE INFO

Article History
Received 13 Jul 2025
Revised 26 Aug 2025
Accepted 4 Sep 2025
Published 03 Oct 2025

Keywords
Financial Development;
MHD nanofluid;
FEM;
Entropy Generation;
Navier-Stokes Equations;
Eight-node rectangular element.

ABSTRACT

In this paper, the Galerkin finite element method was used to solve the Navier-Stokes equations for two-dimensional steady flow of Newtonian and incompressible fluid with no body forces using MATLAB. The method was applied to the lid-driven cavity problem. The eight-noded rectangular element was used for the formulation of element equations. The velocity and pressure components were located at all 8 nodes of the element. From the location of velocity components and pressure, it is obvious that this element consists of 20 unknowns for velocities and pressure. The quadratic interpolation functions represent velocity components, while the bilinear interpolation function represents pressure. Finite element codes were developed for implementation.

Introduction

The Navier–Stokes (N-S) equations form the backbone of fluid mechanics. They represent the conservation of linear momentum for fluid motion and are typically solved alongside the continuity equation. As highlighted by Çengel et al. [1], these equations are highly nonlinear and cannot be solved exactly in most practical cases. Therefore, engineers and scientists often rely on simplifying assumptions, approximations, or numerical methods to obtain approximate solutions. With the advancement of high-speed computing, it has become feasible to replace these differential equations with algebraic forms and solve them numerically using methods such as the finite difference method (FDM), the finite volume method (FVM), and the finite element method (FEM).

*Corresponding author. Email: haiderali@cuivehari.edu.pk

Among these approaches, the finite element method stands out as one of the most powerful and versatile techniques in computational fluid dynamics. FEM is especially suitable for handling complex geometrical domains and offers flexibility in choosing approximation functions. The method transforms a system of partial differential equations (PDEs) into a system of algebraic equations that can be solved using linear algebra techniques. The computational domain is subdivided into smaller regions called finite elements, and within each element, the unknowns are expressed as a linear combination of shape functions. These shape functions are associated with the nodes of the element and determine the behavior of the variable inside the element. Once local approximations are established for all elements, they are assembled into a global system that describes the behavior of the entire domain. One of the key advantages of FEM, as noted by Kwon et al. [2], is that general-purpose codes can be developed and applied to a wide range of problems, including those involving irregular shapes and complex boundary conditions.

Over the years, several researchers have explored FEM for solving the N-S equations. Jiajan [3] applied the Galerkin finite element method to the two-dimensional, unsteady, incompressible N-S equations using six-noded triangular elements, with velocity components defined at all six nodes and pressure defined at the corner nodes. Ghia et al. [4] investigated high Reynolds number incompressible flows using the multigrid method in combination with the N-S equations. Similarly, Persson [5] implemented an FEM-based solver on unstructured triangular meshes to analyze the well-known lid-driven cavity problem for Reynolds numbers ranging from 100 to 2000.

Other notable contributions include Glaisner et al. [6], who studied finite-element formulations of the N-S equations in both primitive variable and vorticity–stream function forms, achieving steady-state solutions for the lid-driven cavity using nine-noded rectangular elements. Taylor et al. [7] employed eight-noded rectangular meshes with FORTRAN programming to solve two-dimensional incompressible flows, while Rhaman et al. [8] developed a FreeFem++ code to simulate fluid particle motion governed by unsteady N-S equations. Further, Simpson [9] applied nine-noded rectangular elements with two degrees of freedom per node in MATLAB for a coupled reaction-diffusion problem, and Khennane [10] designed MATLAB codes for static analysis of two-dimensional elasticity problems using four- and eight-noded quadrilateral elements.

The finite element method has proven to be a robust computational tool, not only for solving fluid flow governed by N-S equations but also for tackling a wide range of differential and integral equations in engineering. Its versatility makes it invaluable in industries such as petroleum, chemical, pharmaceutical, automotive, telecommunications, as well as in emerging fields like nanotechnology and biotechnology. FEM also plays a vital role in heat transfer applications within fluid mechanics. Tsega et al. [11] demonstrated the application of Galerkin FEM to steady, two-dimensional, incompressible Newtonian flows in the absence of body forces. The widely studied lid-driven cavity problem has also been analyzed by Vajjal et al. [12] and Bedrossian et al. [13].

In these studies, element equations were often developed using eight-noded rectangular elements, with pressure variables located at the four corner nodes and velocity components defined at all eight nodes. This setup leads to a total of 20 unknowns per element—four for pressure and sixteen for velocity. Pressure is typically interpolated using bilinear functions, while velocity components are approximated with quadratic functions. For practical implementation, researchers have developed finite element codes that incorporate these formulations to effectively simulate fluid flow.

Problem statement

Governing equations for x and y components.

$$\frac{\partial u^*}{\partial x}+\frac{\partial v^*}{\partial y}=0 \tag{1}$$

$$\rho.[u*.\frac{\partial u*}{\partial x}+\vartheta.\frac{\partial u*}{\partial y}]=-\frac{\partial p}{\partial x}+\mu.[u*.\frac{\partial^2 u*}{\partial x^2}+\vartheta.\frac{\partial^2 u*}{\partial y^2}] \quad (2)$$

$$\rho.[u*.\frac{\partial v*}{\partial x}+\vartheta.\frac{\partial v*}{\partial y}]=-\frac{\partial p}{\partial x}+\mu.[u*.\frac{\partial^2 v*}{\partial x^2}+\vartheta.\frac{\partial^2 v*}{\partial y^2}] \quad (3)$$

Whereas

$static\ \ pressure = p$

$density = \rho$

$dynamic\ \ vis\cos ity = \mu$

L and V*, which epitomize the distinguishing length and velocity that are used to

Let's elucidate the parameters.

$$x' = \frac{x}{L} \rightarrow x = x'.\ L$$

$$y' = \frac{y}{L} \rightarrow y = y'.L$$

$$u* = \frac{u}{\vartheta} \rightarrow u = u*.\vartheta$$

$$v* = \frac{v}{\vartheta} \rightarrow v = v*.\vartheta$$

$$p' = \frac{p}{\rho\vartheta^2} \rightarrow p = p'.\rho\vartheta^2$$

Consuming these similarities, we have to find the equations (1), (2), and (3)

$$\frac{\partial u*}{\partial x}+\frac{\partial v*}{\partial y}=0 \quad (4)$$

For the x component

$$u*.\frac{\partial u*}{\partial x}+v*.\frac{\partial u*}{\partial y}=-\frac{\partial p}{\partial x}+\frac{1}{R_e}.[u*.\frac{\partial^2 u*}{\partial x^2}+\vartheta.\frac{\partial^2 u*}{\partial y^2}] \quad (5)$$

Similarly, for the y-component

$$u*.\frac{\partial v*}{\partial x}+v*.\frac{\partial v*}{\partial y}=-\frac{\partial p}{\partial x}+\frac{1}{R_e}.[u*.\frac{\partial^2 v*}{\partial x^2}+\vartheta.\frac{\partial^2 v*}{\partial y^2}] \quad (6)$$

Methodology

The flow is assumed in a rectangular cavity with a uniform horizontal velocity. This is done to create a more compatible study state solution for the N-S Equations. The permanent three sides don't move. Rectangular elements with 08 nodes are applied for the flow model. The four corner nodes are used to illustrate pressure, whereas all 08 nodes of every element are recycled to make an exemplary idea for the

velocity components u* and v* separately, the x-direction is used to estimate meshes. The number of degrees of freedom is u*-p-v*. Let's designate an element. Local coordinates ς and τ are used to precisely define the shape functions for the rectangular elements.

$$\varsigma = \frac{2.(x - x.c)}{l.x}$$

$$\tau = \frac{2.(y - y.c)}{l.y}$$

Let's say the nodes 1, 2, 3, 4, 5, 6, 7, and in the local coordinate system, 8 have coordinates of (-1, -1), (0, -1), (1, -1), (0, 1), (-1, 1), and (-1, 0).

Using local coordinates, the shape functions for a 4-noded bilinear rectangular element account for corner nodes, which can be expressed generally as

$$S = a^* + b^*\varsigma + c^*\tau + d^*\varsigma\tau \tag{7}$$

When utilizing local coordinates, the shape functions for an 8-noded quadratic rectangular element (taking into account all nodes) take on the general form

$$T = a^* + b^*\varsigma + c^*\tau + d^*\varsigma^2 + e^*\tau\varsigma + f^*\tau^2 + g^*\varsigma^2\tau + h^*\varsigma\tau^2 \tag{8}$$

Shape functions for rectangular elements with (04) four and (08) eight nodes can result from Equations (7) and (8) and be used consequently.

$$S_1 = \frac{(\varsigma - 1).(\tau - 1)}{(-2).(-2)}$$

$$S_1 = \frac{1 - \varsigma + \tau + \varsigma.\tau}{4}$$

Similarly

$$S_2 = \frac{(\varsigma + 1).(\tau - 1)}{(2).(-2)}$$

$$S_2 = \frac{1 - \varsigma - \tau - \varsigma.\tau}{4}$$

$$S_3 = \frac{(\varsigma + 1).(\tau + 1)}{(1 + 1).(1 + 1)}$$

$$S_3 = \frac{1 + \varsigma + \tau + \varsigma.\tau}{4}$$

$$S_4 = \frac{(\varsigma - 1).(\tau + 1)}{(-1 - 1).(1 + 1)}$$

$$S_4 = \frac{1 - \varsigma - \tau - \varsigma.\tau}{4} \tag{9}$$

$$T_1 = \frac{(\varsigma - 1).(\tau - 1).(1 + \varsigma + \tau)}{4}$$

$$T_2 = \frac{(1 - \varsigma^2).(1 - \tau)}{2}$$

$$T_3 = -\frac{(\varsigma+1).(1-\tau).(1-\varsigma+\tau)}{4}$$
$$T_4 = \frac{(1+\varsigma).(1-\tau^2)}{2}$$
$$T_5 = -\frac{(\varsigma+1).(1+\tau).(1-\varsigma+\tau)}{4}$$
$$T_6 = \frac{(1-\varsigma^2).(1+\tau)}{2}$$
$$T_7 = -\frac{(\varsigma-1).(1+\tau).(1-\varsigma+\tau)}{4}$$
$$T_8 = \frac{(1-\varsigma).(1-\tau^2)}{2} \quad (10)$$

Consequently, interpolated bilinear functions are utilized for pressure, and quadratic interpolation functions are engaged for velocity components. Therefore, there are twenty unknown pressure and velocity-related factors for every constituent.

Hence, the three dependent variables u*, v*, and p are conveyed as

$$u^* = \sum_{i=1}^{8} T_i u_i^*$$
$$v^* = \sum_{i=1}^{8} T_i v^*_i \quad (11)$$
$$p = \sum_{i=1}^{8} S_i p_i$$

The velocity and pressure values at nodes are represented by the variables u_i v_i and p_i

We get by using the shape function on equations (4),(5), and (6).

$$\sum_{j=1}^{8} \frac{\partial T_j}{\partial x} u^*_j + \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} v^*_j = 0 \quad (12)$$

$$\sum_{k=1}^{8} T_k u^*_k \sum_{j=1}^{8} \frac{\partial T_j}{\partial x} u^*_j + \sum_{k=1}^{8} T_k v^*_k \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} u^*_j = -\sum_{l=1}^{4} \frac{\partial S_l}{\partial x} p_l + \frac{1}{R_e}\left[\sum_{j=1}^{8} \frac{\partial^2 T_j}{\partial x^2} u^*_j + \sum_{j=1}^{8} \frac{\partial^2 T_j}{\partial y^2} u^*_j\right] \quad (13)$$

$$\sum_{k=1}^{8} T_k u^*_k \sum_{j=1}^{8} \frac{\partial T_j}{\partial x} v^*_j + \sum_{k=1}^{8} T_k v^*_k \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} v^*_j = -\sum_{l=1}^{4} \frac{\partial S_l}{\partial y} p_l + \frac{1}{R_e}\left[\sum_{j=1}^{8} \frac{\partial^2 T_j}{\partial x^2} v^*_j + \sum_{j=1}^{8} \frac{\partial^2 T_j}{\partial y^2} v^*_j\right] \quad (14)$$

Apply the Galerkin weighted residual approach, multiply equation (13) by a weighted function, and then integrate across the domain Ω.

Using the Gauss Divergence Theorem

$$\iint \frac{1}{R_e}[W_i \sum_{j=1}^{8} \frac{\partial^2 T_j}{\partial x^2} u^*{}_j + W_i \sum_{j=1}^{8} \frac{\partial^2 T_j}{\partial y^2} u^*{}_j] dA$$

$$= -\frac{1}{R_e}[\iint W_i \sum_{j=1}^{8} \frac{\partial T_j}{\partial x} u^*{}_j \, dA + W_i \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} v^*{}_j \, dA] + \frac{1}{R_e} \int T_i \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} u^*{}_j \, ds$$

While we have

Γ = the edge of the Ω element

$n = (n_x + n_y)$, unit and direction normal are to the element and boundary, respectively.

Γ is $\frac{\partial T_j}{\partial n} = \frac{\partial T_j}{\partial x}.n_x + \frac{\partial T_j}{\partial y}.n_y$

Therefore,

$$\iint W_i [\sum_{k=1}^{8} T_k u^*{}_k \sum_{j=1}^{8} \frac{\partial T_j}{\partial x} u^*{}_j + W_i \sum_{k=1}^{8} T_k v^*{}_k \frac{\partial T_j}{\partial y} u^*{}_j + W_i \sum_{l=1}^{4} \frac{\partial S_l}{\partial x} p_l + \frac{1}{R_e}[\frac{\partial T_j}{\partial x} \sum_{j=1}^{8} \frac{\partial T_j}{\partial x} u^*{}_j$$

$$+ \frac{\partial T_j}{\partial y} \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} v^*{}_j \, dA] - \frac{1}{R_e} \int T_i \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} u^*{}_j \, ds = 0 \qquad (15)$$

for $i = j = 1,2,3,...8$

Regarding the Dirichlet boundary condition, we possess

$$\iint W_i \sum_{k=1}^{8} T_k u^*{}_k \sum_{j=1}^{8} \frac{\partial T_j}{\partial x} u^*{}_j + W_i \sum_{k=1}^{8} T_k v^*{}_k \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} u^*{}_j +$$

$$W_i \sum_{l=1}^{4} \frac{\partial S_l}{\partial x} p_l + \frac{1}{R_e} \left[\frac{\partial T_j}{\partial x} \sum_{j=1}^{8} \frac{\partial T_j}{\partial x} u^*{}_j + \frac{\partial T_j}{\partial y} \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} u^*{}_j \, dA \right] dA = 0 \qquad (16)$$

Using the same process for Equation (14), we arrive at

$$\iint W_i \sum_{k=1}^{8} T_k u^*{}_k \sum_{j=1}^{8} \frac{\partial T_j}{\partial x} v^*{}_j + W_i \sum_{k=1}^{8} T_k v^*{}_k \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} v^*{}_j$$

$$+W_i \sum_{l=1}^{4} \frac{\partial S_l}{\partial x} p_l + \frac{1}{R_e}\left[\frac{\partial T_j}{\partial x}\sum_{j=1}^{8}\frac{\partial T_j}{\partial x} v^*{}_j + \frac{\partial T_j}{\partial y}\sum_{j=1}^{8}\frac{\partial T_j}{\partial y} v^*{}_j \, dA\right].dA = 0 \quad (17)$$

for $i = j = 1,2,3,...8$

Using the weight functions S_1 across the domain Ω and a Galerkin weighted residual technique on Equation (12), we obtain

$$\iint S_l \left[\sum_{j=1}^{8} \frac{\partial T_j}{\partial x} u^*{}_j + \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} v^*{}_j\right] dA = 0$$

$$\iint \left[S_l \sum_{j=1}^{8} \frac{\partial T_j}{\partial x} u^*{}_j + S_l \sum_{j=1}^{8} \frac{\partial T_j}{\partial y} v^*{}_j\right] dA = 0$$

$whereasl = 1,2,3$ (18)

We are on a system of nonlinear algebraic equations; a single solution isn't feasible; instead, an iterative approach is needed. There are numerous ways to linearize nonlinear terms in an iterative solution like this. The most straightforward choice that will be employed is called Picard linearization, where the nonlinear elements are swapped out for

$$u^*.\frac{\partial u^*}{\partial x} + v^*.\frac{\partial u^*}{\partial y} + \bar{u}.\frac{\partial u^*}{\partial x} + \bar{v}.\frac{\partial u}{\partial y}$$

$$u^*.\frac{\partial v^*}{\partial x} + v^*.\frac{\partial v^*}{\partial y} + \bar{u}.\frac{\partial v^*}{\partial x} + \bar{v}.\frac{\partial v^*}{\partial y}$$

Where the velocity components' approximate values are denoted by $\bar{u}$ and $\bar{v}$. Starting values for the problem $u^*_{01,}u^*_{02,}u^*_{03,}u^*_{04,}...u^*_{08,}$ *and* $v^*_{01,}v^*_{02,}v^*_{03,}v^*_{04,}...v^*_{08,}$

$$\bar{u} = \sum_{i=1}^{8} T_k u^*_{0\,k}$$

$$\bar{v} = \sum_{i=1}^{8} T_k v^*_{0\,k}$$

Until forbearance is satisfied, the iteration process is repeated by substituting $u^*_{0\,k}$ *and* $v^*_{0\ k}$ *where* $k=1,2,3,...8$ with the average of the velocity component values from the previous two iterations. Equations (16), (17), and (18) yield the system of equations in matrix form.

$$Ah = b \quad (19)$$

Where

$$A=\begin{bmatrix}
a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & a_{14}^{(1)} & a_{15}^{(1)} & a_{16}^{(1)} & a_{17}^{(1)} & a_{18}^{(1)} & a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} & a_{14}^{(2)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} & a_{25}^{(1)} & a_{26}^{(1)} & a_{27}^{(1)} & a_{28}^{(1)} & a_{21}^{(2)} & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} & a_{35}^{(1)} & a_{36}^{(1)} & a_{37}^{(1)} & a_{38}^{(1)} & a_{31}^{(2)} & a_{32}^{(2)} & a_{33}^{(2)} & a_{34}^{(2)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
a_{41}^{(1)} & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} & a_{45}^{(1)} & a_{46}^{(1)} & a_{47}^{(1)} & a_{48}^{(1)} & a_{41}^{(2)} & a_{42}^{(2)} & a_{43}^{(2)} & a_{44}^{(2)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
a_{51}^{(1)} & a_{52}^{(1)} & a_{53}^{(1)} & a_{54}^{(1)} & a_{55}^{(1)} & a_{56}^{(1)} & a_{57}^{(1)} & a_{58}^{(1)} & a_{51}^{(2)} & a_{52}^{(2)} & a_{53}^{(2)} & a_{54}^{(2)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
a_{61}^{(1)} & a_{62}^{(1)} & a_{63}^{(1)} & a_{64}^{(1)} & a_{65}^{(1)} & a_{66}^{(1)} & a_{67}^{(1)} & a_{68}^{(1)} & a_{61}^{(2)} & a_{62}^{(2)} & a_{63}^{(2)} & a_{64}^{(2)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
a_{71}^{(1)} & a_{72}^{(1)} & a_{73}^{(1)} & a_{74}^{(1)} & a_{75}^{(1)} & a_{76}^{(1)} & a_{77}^{(1)} & a_{78}^{(1)} & a_{71}^{(2)} & a_{72}^{(2)} & a_{73}^{(2)} & a_{74}^{(2)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
a_{81}^{(1)} & a_{82}^{(1)} & a_{83}^{(1)} & a_{84}^{(1)} & a_{85}^{(1)} & a_{86}^{(1)} & a_{87}^{(1)} & a_{88}^{(1)} & a_{81}^{(2)} & a_{82}^{(2)} & a_{83}^{(2)} & a_{84}^{(2)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
a_{11}^{(4)} & a_{12}^{(4)} & a_{13}^{(4)} & a_{14}^{(4)} & a_{15}^{(4)} & a_{16}^{(4)} & a_{17}^{(4)} & a_{18}^{(4)} & 0 & 0 & 0 & 0 & a_{11}^{(6)} & a_{12}^{(6)} & a_{13}^{(6)} & a_{14}^{(6)} & a_{15}^{(6)} & a_{16}^{(6)} & a_{17}^{(6)} & a_{18}^{(6)}\\
a_{21}^{(4)} & a_{22}^{(4)} & a_{23}^{(4)} & a_{24}^{(4)} & a_{25}^{(4)} & a_{26}^{(4)} & a_{27}^{(4)} & a_{28}^{(4)} & 0 & 0 & 0 & 0 & a_{21}^{(6)} & a_{22}^{(6)} & a_{23}^{(6)} & a_{24}^{(6)} & a_{25}^{(6)} & a_{26}^{(6)} & a_{27}^{(6)} & a_{28}^{(6)}\\
a_{31}^{(4)} & a_{32}^{(4)} & a_{33}^{(4)} & a_{34}^{(4)} & a_{35}^{(4)} & a_{36}^{(4)} & a_{37}^{(4)} & a_{38}^{(4)} & 0 & 0 & 0 & 0 & a_{31}^{(6)} & a_{32}^{(6)} & a_{33}^{(6)} & a_{34}^{(6)} & a_{35}^{(6)} & a_{36}^{(6)} & a_{37}^{(6)} & a_{38}^{(6)}\\
a_{41}^{(4)} & a_{42}^{(4)} & a_{43}^{(4)} & a_{44}^{(4)} & a_{45}^{(4)} & a_{46}^{(4)} & a_{47}^{(4)} & a_{48}^{(4)} & 0 & 0 & 0 & 0 & a_{41}^{(6)} & a_{42}^{(6)} & a_{43}^{(6)} & a_{44}^{(6)} & a_{45}^{(6)} & a_{46}^{(6)} & a_{47}^{(6)} & a_{48}^{(6)}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11}^{(8)} & a_{12}^{(8)} & a_{13}^{(8)} & a_{14}^{(8)} & a_{11}^{(9)} & a_{12}^{(9)} & a_{13}^{(9)} & a_{14}^{(9)} & a_{15}^{(9)} & a_{16}^{(9)} & a_{17}^{(9)} & a_{18}^{(9)}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{21}^{(8)} & a_{22}^{(8)} & a_{23}^{(8)} & a_{24}^{(8)} & a_{21}^{(9)} & a_{22}^{(9)} & a_{23}^{(9)} & a_{24}^{(9)} & a_{25}^{(9)} & a_{26}^{(9)} & a_{27}^{(9)} & a_{28}^{(9)}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{31}^{(8)} & a_{32}^{(8)} & a_{33}^{(8)} & a_{34}^{(8)} & a_{31}^{(9)} & a_{32}^{(9)} & a_{33}^{(9)} & a_{34}^{(9)} & a_{35}^{(9)} & a_{36}^{(9)} & a_{37}^{(9)} & a_{38}^{(9)}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{41}^{(8)} & a_{42}^{(8)} & a_{43}^{(8)} & a_{44}^{(8)} & a_{41}^{(9)} & a_{42}^{(9)} & a_{43}^{(9)} & a_{44}^{(9)} & a_{45}^{(9)} & a_{46}^{(9)} & a_{47}^{(9)} & a_{48}^{(9)}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{51}^{(8)} & a_{52}^{(8)} & a_{53}^{(8)} & a_{54}^{(8)} & a_{51}^{(9)} & a_{52}^{(9)} & a_{53}^{(9)} & a_{54}^{(9)} & a_{55}^{(9)} & a_{56}^{(9)} & a_{57}^{(9)} & a_{58}^{(9)}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{61}^{(8)} & a_{62}^{(8)} & a_{63}^{(8)} & a_{64}^{(8)} & a_{61}^{(9)} & a_{62}^{(9)} & a_{63}^{(9)} & a_{64}^{(9)} & a_{65}^{(9)} & a_{66}^{(9)} & a_{67}^{(9)} & a_{68}^{(9)}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{71}^{(8)} & a_{72}^{(8)} & a_{73}^{(8)} & a_{74}^{(8)} & a_{71}^{(9)} & a_{72}^{(9)} & a_{73}^{(9)} & a_{74}^{(9)} & a_{75}^{(9)} & a_{76}^{(9)} & a_{77}^{(9)} & a_{78}^{(9)}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{81}^{(8)} & a_{82}^{(8)} & a_{83}^{(8)} & a_{84}^{(8)} & a_{81}^{(9)} & a_{82}^{(9)} & a_{83}^{(9)} & a_{84}^{(9)} & a_{85}^{(9)} & a_{86}^{(9)} & a_{87}^{(9)} & a_{88}^{(9)}
\end{bmatrix}$$

$$h=\begin{bmatrix}u\\p\\v\end{bmatrix}=\begin{bmatrix}u_1\\u_2\\u_3\\u_4\\u_5\\u_6\\u_7\\u_8\\p_1\\p_2\\p_3\\p_4\\v_1\\v_2\\v_3\\v_4\\v_5\\v_6\\v_7\\v_8\end{bmatrix},\ b=0\ (20\times 1).$$

where $u^{*}=u$, $v^{*}=v$

And we get $\boldsymbol{b}=\boldsymbol{0}$

Here

$$a^{1}{}_{ij}=\iint T_i.\overline{u}.\frac{\partial T_j}{\partial x}+T_i.\overline{v}.\frac{\partial T_j}{\partial x}+\frac{1}{R_e}.\left[\frac{\partial T_i}{\partial x}\frac{\partial T_j}{\partial x}+\frac{\partial T_i}{\partial y}\frac{\partial T_j}{\partial y}\right].dA \qquad where \quad j=i=1,2,3,\ldots 8$$

$$a^{9}_{ij} = a^{1}_{ij}$$

$$a^{2}_{ij} = \iint T_i \frac{\partial S_j}{\partial x} dA \qquad as \quad i = j = 1,2,3,...8$$

$$a^{8}_{ij} = \iint T_i \frac{\partial S_j}{\partial y} dA \qquad as \quad i = j = 1,2,3,...8$$

$$a^{4}_{ij} = \iint S_i \frac{\partial T_j}{\partial x} dA \qquad as \quad i = j = 1,2,3,...8$$

$$a^{6}_{ij} = \iint S_i \frac{\partial T_j}{\partial y} dA \qquad as \quad i = j = 1,2,3,...8$$

Utilizing Local Coordinates for Derivatives and Integrals

Let $T(\varsigma \ \tau)$ the shape function concern the local coordinates. Permitted that the Global coordinates are x and y, therefore

$$\frac{\partial T}{\partial \varsigma} = \frac{\partial T}{\partial x} . \frac{\partial x}{\partial \varsigma} + \frac{\partial T}{\partial y} . \frac{\partial y}{\partial \varsigma}$$

$$\frac{\partial T}{\partial \tau} = \frac{\partial T}{\partial x} . \frac{\partial x}{\partial \tau} + \frac{\partial T}{\partial y} . \frac{\partial y}{\partial \tau}$$

$$\begin{bmatrix} \frac{\partial T}{\partial \varsigma} \\ \frac{\partial N}{\partial \tau} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \varsigma} & \frac{\partial y}{\partial \varsigma} \\ \frac{\partial x}{\partial \tau} & \frac{\partial y}{\partial \tau} \end{bmatrix} . \begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{bmatrix}$$

where

$$\begin{bmatrix} \frac{\partial x}{\partial \varsigma} & \frac{\partial y}{\partial \varsigma} \\ \frac{\partial x}{\partial \tau} & \frac{\partial y}{\partial \tau} \end{bmatrix}$$

and

$$J = JocobianMatrix$$

Translating GCS (global coordinate system) to LCS (local coordinate system) is,

$$\begin{bmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial T}{\partial \varsigma} \\ \frac{\partial T}{\partial \tau} \end{bmatrix} \tag{20}$$

The Jacobian Matrix Calculation

Assume that the shape functions in the local coordinates are

$$x = T_1.(\varsigma,\tau).x_1 + T_2.(\varsigma,\tau).x_2 + T_3.(\varsigma,\tau).x_3,\ldots + T_n.(\varsigma,\tau).x_n$$
$$y = T_1.(\varsigma,\tau).y_1 + T_2.(\varsigma,\tau).y_2 + T_3.(\varsigma,\tau).y_3,\ldots + T_n.(\varsigma,\tau).y_n$$
$$\frac{\partial x}{\partial \varsigma} = \frac{\partial T_1}{\partial \varsigma}.x_1 + \frac{\partial T_2}{\partial \varsigma}.x_2 +,\ldots + \frac{\partial T_n}{\partial \varsigma}.x_n$$
$$\frac{\partial y}{\partial \varsigma} = \frac{\partial T_1}{\partial \varsigma}.y_1 + \frac{\partial T_2}{\partial \varsigma}.y_2 +,\ldots + \frac{\partial T_n}{\partial \varsigma}.y_n$$
$$\frac{\partial x}{\partial \tau} = \frac{\partial T_1}{\partial \tau}.x_1 + \frac{\partial T_2}{\partial \tau}.x_2 +,\ldots + \frac{\partial T_n}{\partial \tau}.x_n$$
$$\frac{\partial y}{\partial \tau} = \frac{\partial T_1}{\partial \tau}.y_1 + \frac{\partial T_2}{\partial \tau}.y_2 +,\ldots + \frac{\partial T_n}{\partial \tau}.y_n$$

The obtained Jacobian matrix J is,

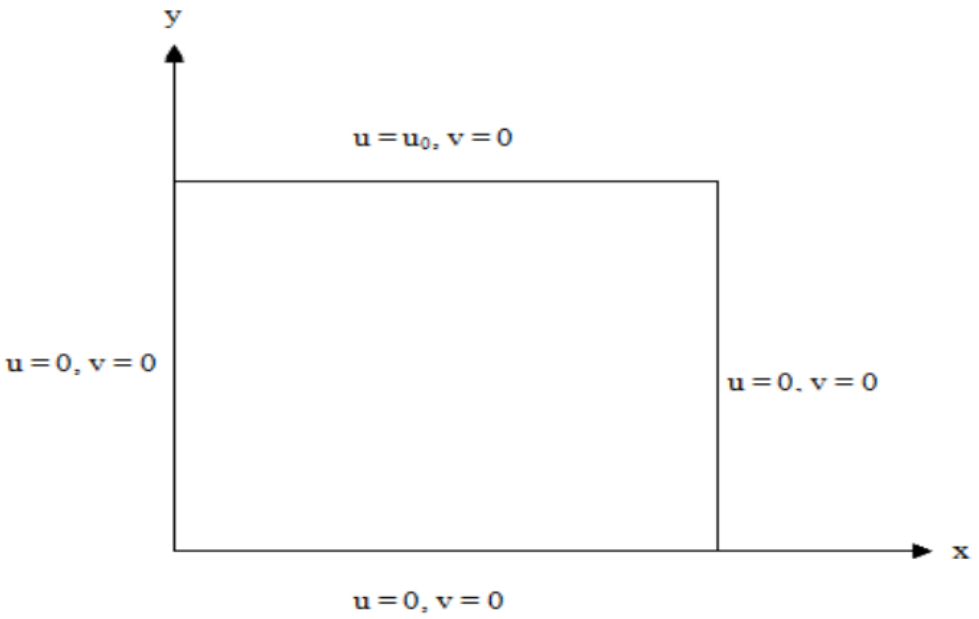
$$J = \begin{bmatrix} \frac{\partial x}{\partial \varsigma} & \frac{\partial y}{\partial \varsigma} \\ \frac{\partial x}{\partial \tau} & \frac{\partial y}{\partial \tau} \end{bmatrix} = \begin{bmatrix} \frac{\partial T_1}{\partial \varsigma} & \frac{\partial T_2}{\partial \varsigma} \cdots & \frac{\partial T_n}{\partial \varsigma} \\ \frac{\partial T_1}{\partial \tau} & \frac{\partial T_2}{\partial \tau} \cdots & \frac{\partial T_n}{\partial \tau} \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ . & . \\ . & . \\ . & . \\ x_n & y_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n}.\frac{\partial T_i}{\partial \varsigma}.x_i & \sum_{i=1}^{n}.\frac{\partial T_i}{\partial \varsigma}.y_i \\ \sum_{i=1}^{n}.\frac{\partial T_i}{\partial \tau}.x_i & \sum_{i=1}^{n}.\frac{\partial T_i}{\partial \tau}.y_i \end{bmatrix} \quad (21)$$

Assembly of the global system

The elemental equations were assembled once the governing equations for each element were shaped. This allowed the development of an equation system for the entire domain. The global system equations are created using the element equations and are composed of element connections, global coordinates, and degrees of freedom for nodes. By using the boundary, the global system of equations can be obtained.

Lid-driven cavity flow problem using FEM

The N-S equations for the 2-dimensional flow of an incompressible fluid in the absence of body forces are solved using Galerkin FEM. This approach works for the lid-driven cavity issue. Equations for the elements were developed using the eight-noded rectangular element. At the four corners, the pressure variable is located, and velocity is at the eight nodes. It is evident from the locations of the pressure and velocity components that this element has four unknowns for pressure and sixteen unknowns for velocities. Consequently, there are 20 unknown variables for pressure and velocities for each element. Whereas the bilinear interpolation function reflects pressure, the quadratic interpolation functions represent velocity components. Codes with finite elements were created for the lid-driven cavity problem. A well-known standard problem in the lid-driven cavity for viscous incompressible flow. The problem has two-dimensional geometry and simple boundary conditions. The fluid is enclosed inside the square domain, and both sides have Dirichlet boundary constraints applied. The velocity of the moving side is parallel to the side. To quicken the flow inside the hollow, the tangential velocity is supplied to the upper border. Regarding the remaining three walls with zero speed, no-slip regulations are in effect.



Where

$u^* = u$, $v^* = v$

Fig 1: Lid Driven Cavity

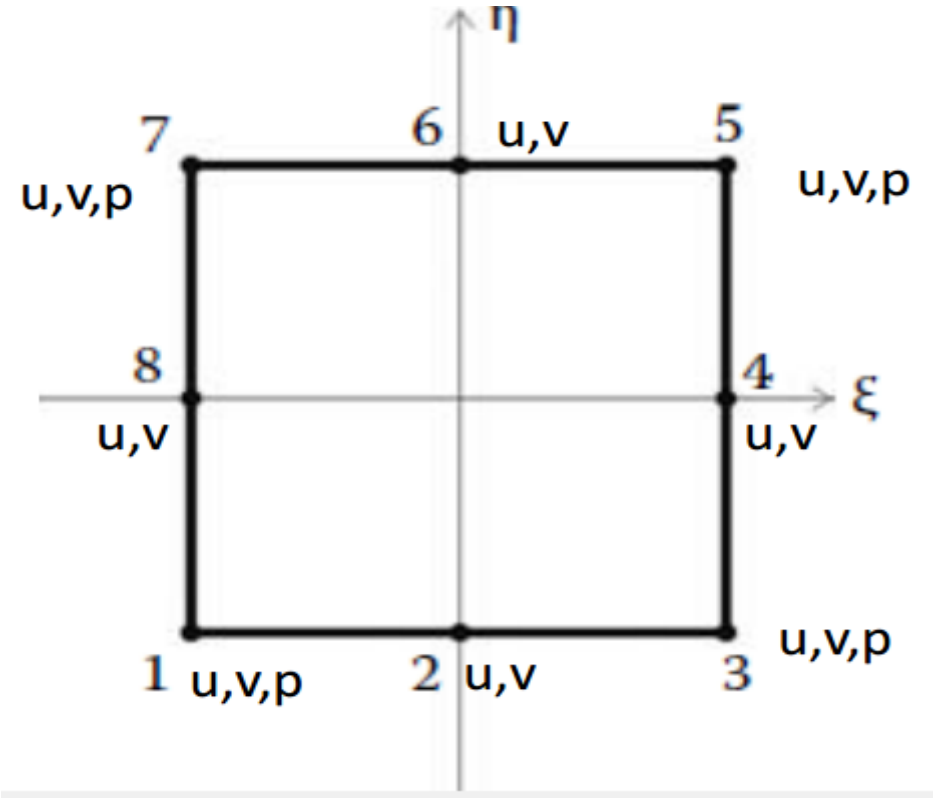


Fig 2: 8 nodes rectangular component

Where $u^* = u$, $v^* = v$

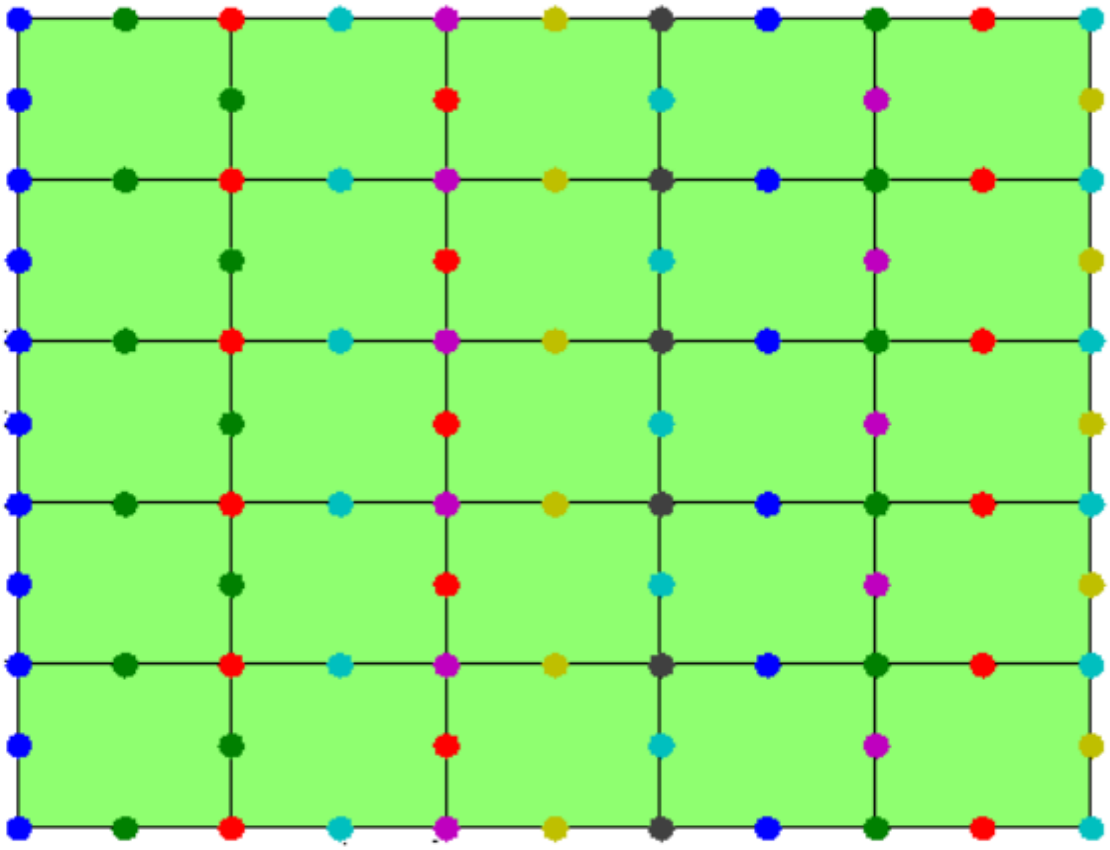


Fig 3: 8-node rectangular elements

Assembly of the global system and boundary conditions

By creating the governing equations for each element and then assembling them, it can construct a GDS equation for the domain. The element equations are used to build the LCS equation together with the global coordinates' elements and the global degree of freedom at nodes. The revised global system of equations is generated by applying the boundary conditions. A length-one cavity flow driven by a square lid was examined due to the boundary conditions. The vector field of the pressure model uses four nodal elements, while the vector field of the velocity model uses eight nodal elements.

Assumptions

Here, it is probable to obtain the pressure contour and velocity quiver map numerically at $Re = 1, 10, 50, 100, 500, 1000, 1200, 1400, 1500, \ldots$. This series of codes will continue for finite elements. The school of work that we performed yielded the results. The convergence is confirmed by increasing the tolerance.

Results and discussions

The numerical simulations were performed for Reynolds numbers 1, 10, 50, 100, 200, 500, 1000, 1200, 1400, and 1500 on a computational domain discretized with a 100-element mesh comprising 803 nodes. The results are presented in terms of velocity vector (quiver) fields and pressure contour distributions corresponding to each Reynolds number. All simulations were executed using a series of in-house finite element codes developed specifically for this study. A convergence criterion of 10^{-6} was imposed to ensure numerical stability and accuracy of the computed solutions.

As summarized in Table 1, it was observed that the number of iterations as well as the computational time increased considerably with higher Reynolds numbers. This trend highlights the greater numerical effort required to achieve convergence in simulations of flows at elevated Reynolds regimes.

Table 1: Five simulations for the cavity flow were run computationally.

Reynolds number	1	10	50	100	200	500	1000
Total iterations	21	22	26	29	35	47	147
Time for iterations (in seconds)	4.04	4.14	4.72	5.26	6.21	7.62	21.06

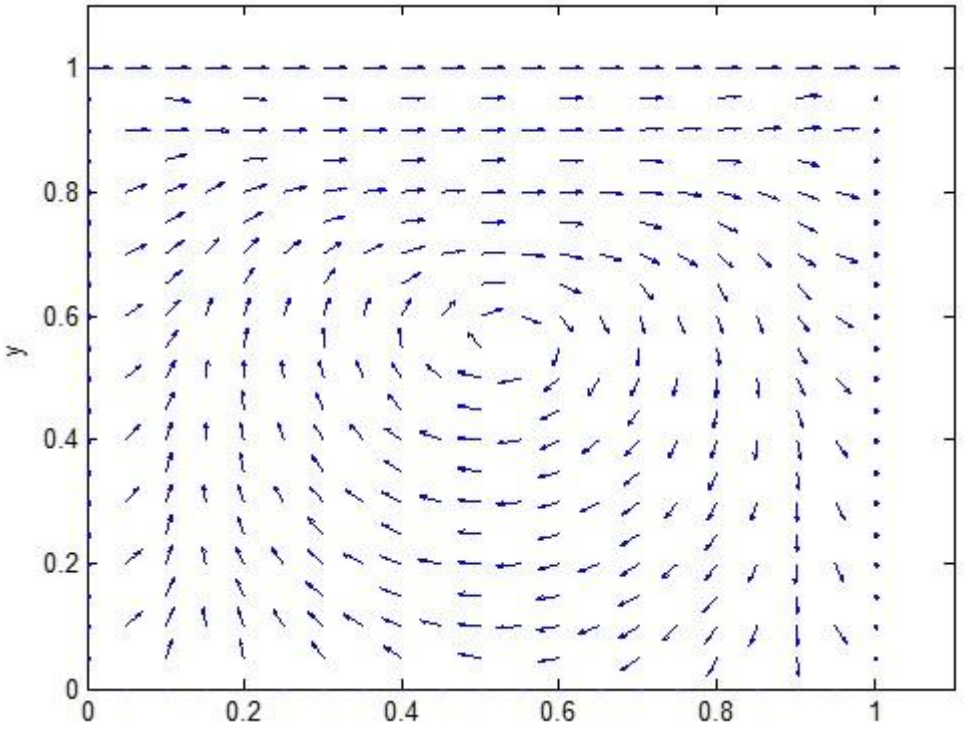


Fig 4: Velocity for $Re = 1$

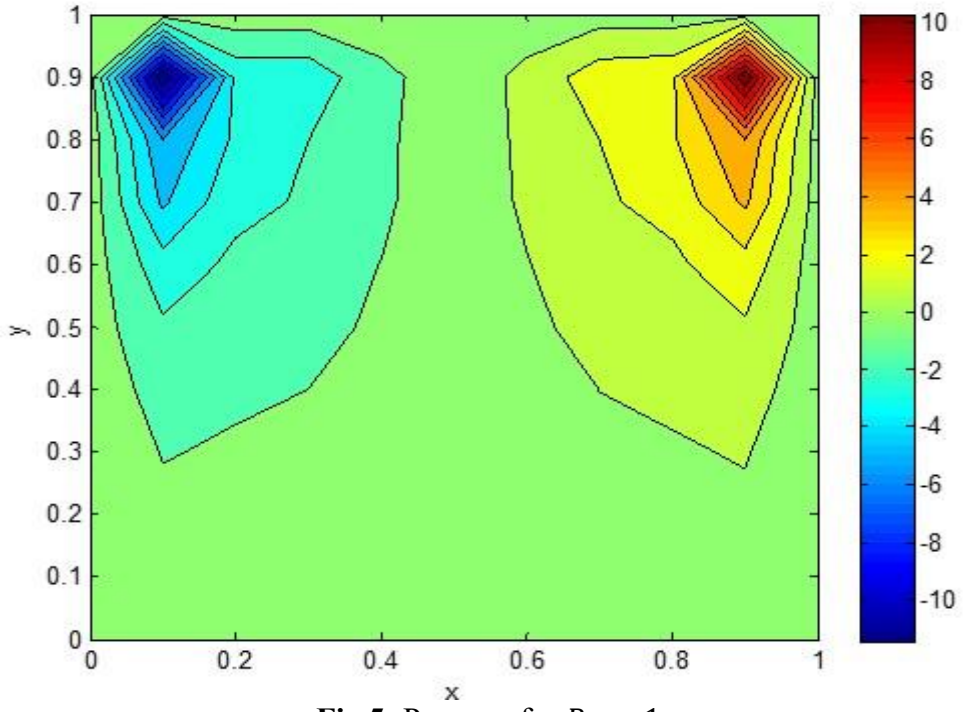


Fig 5: Pressure for $Re = 1$

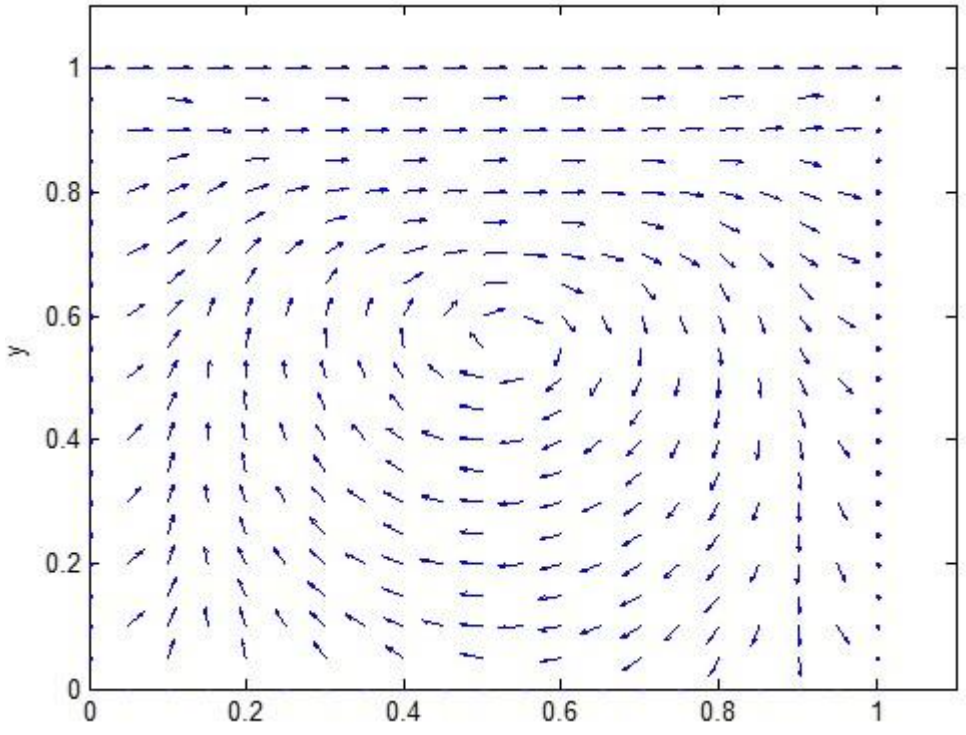


Fig 6: Velocity for $Re = 10$

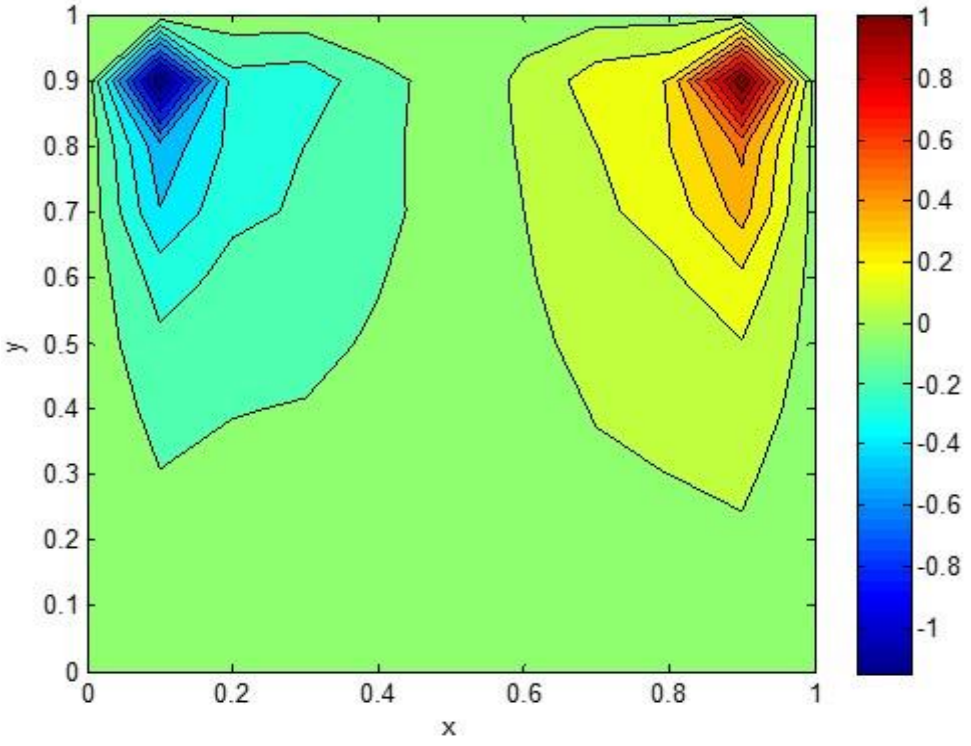


Fig 7: Pressure for $Re = 10$

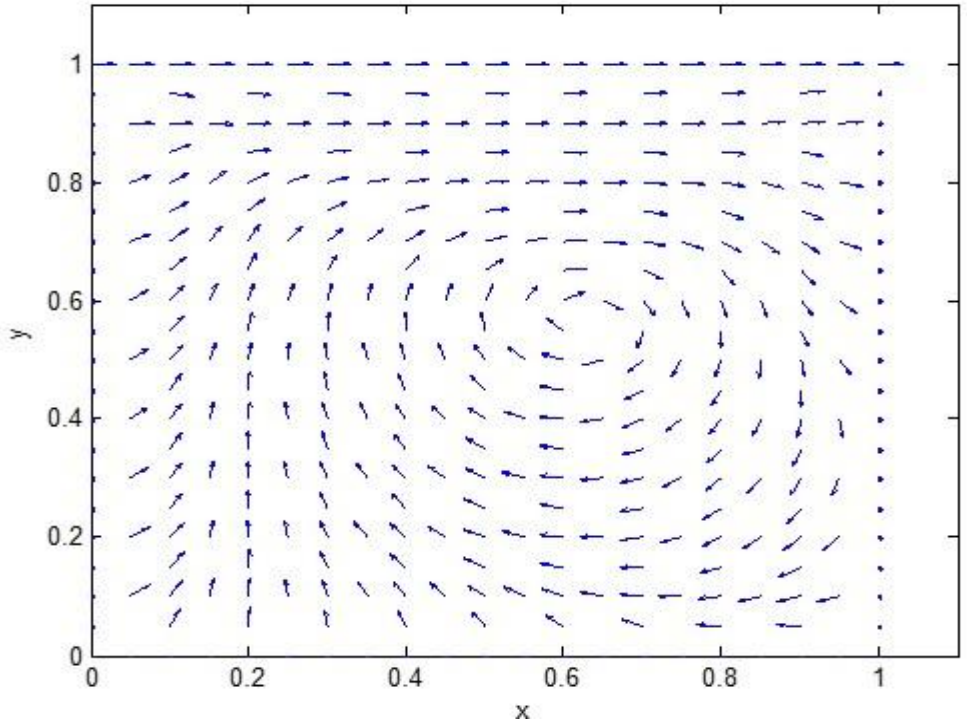


Fig 8: Velocity for $Re = 50$

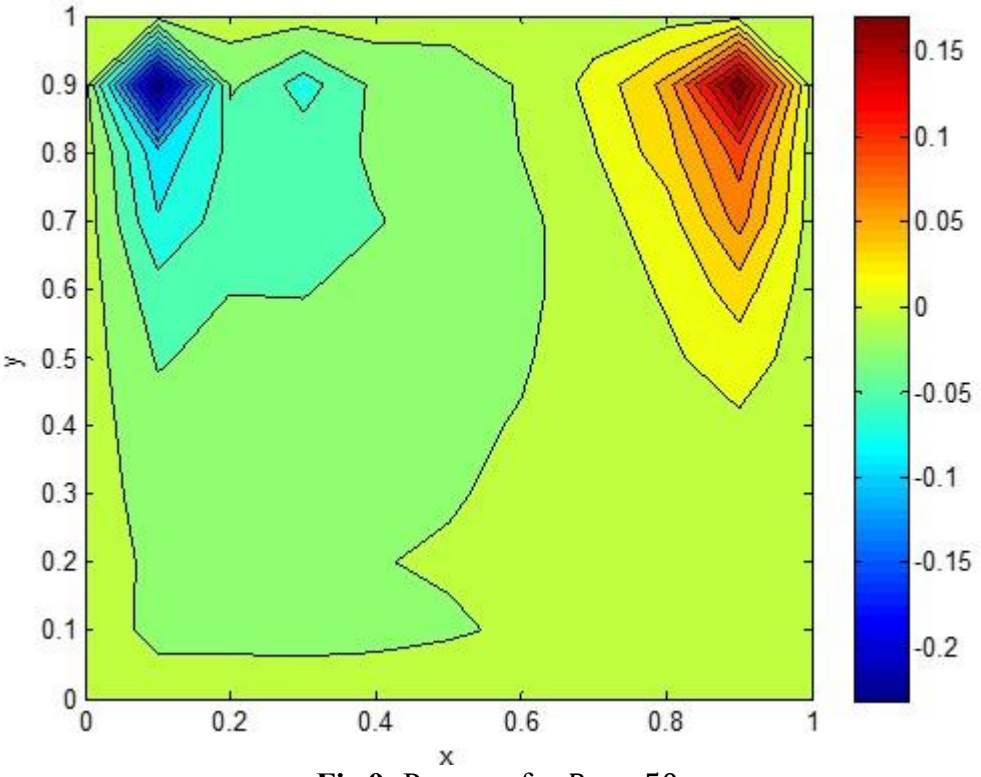


Fig 9: Pressure for $Re = 50$

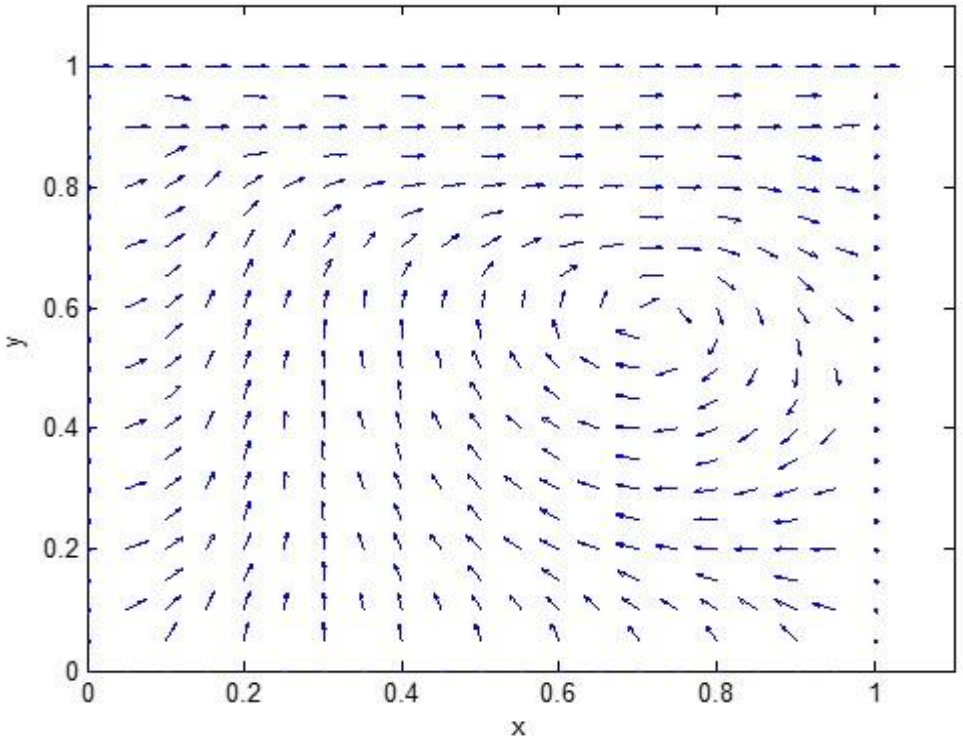


Fig 10: Velocity for $Re = 100$

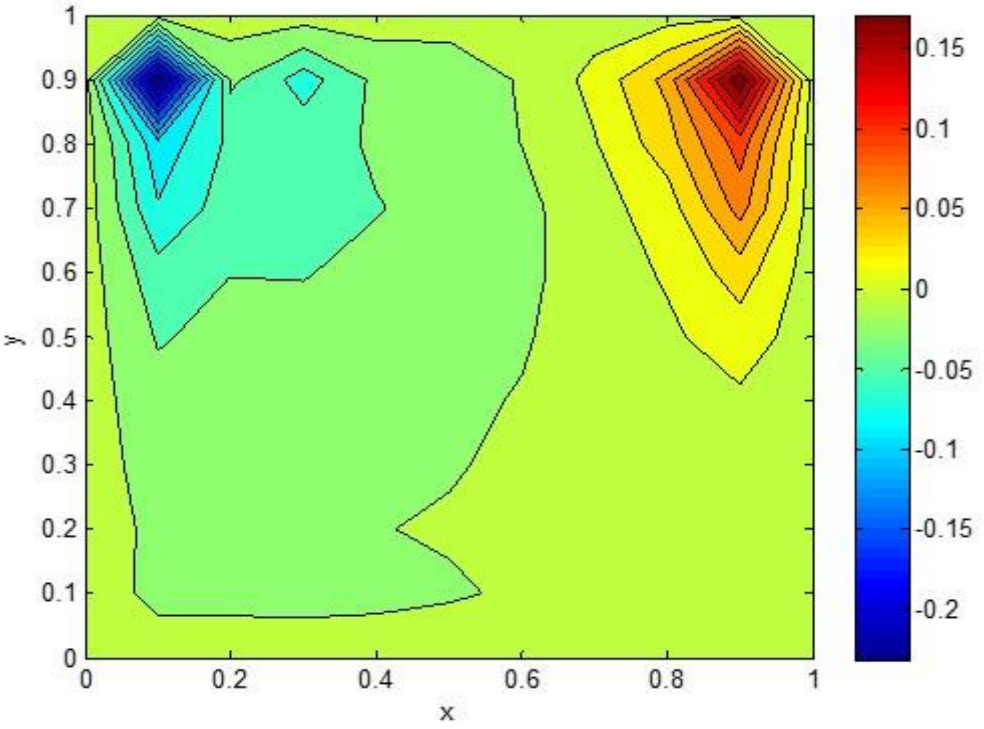


Fig 11: Pressure for $Re = 100$

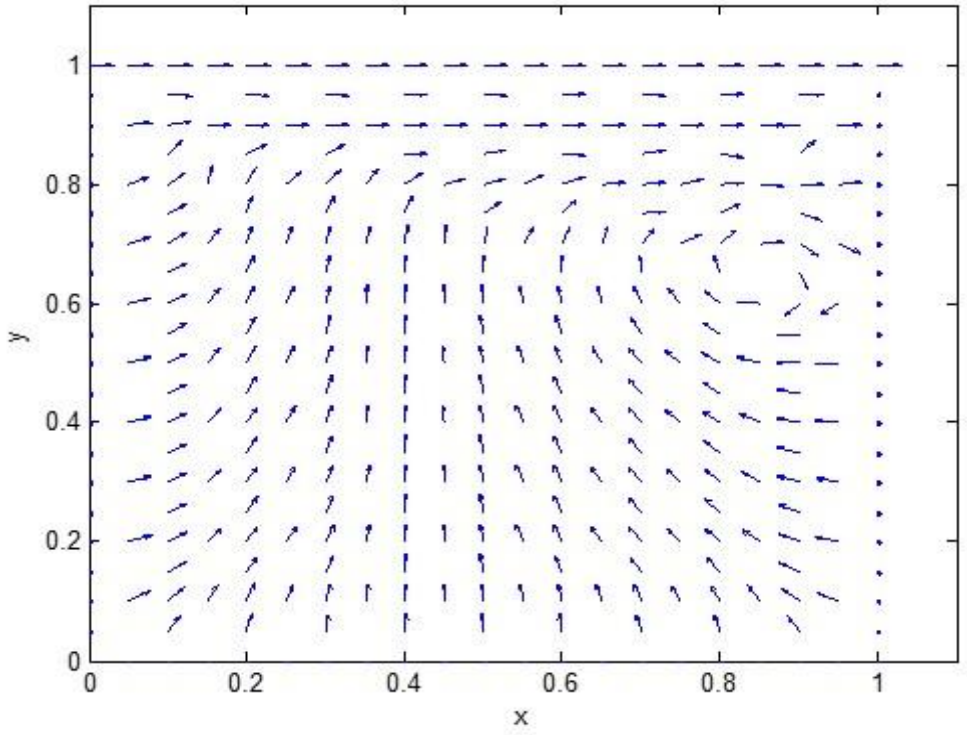


Fig 12: Velocity for $Re = 400$

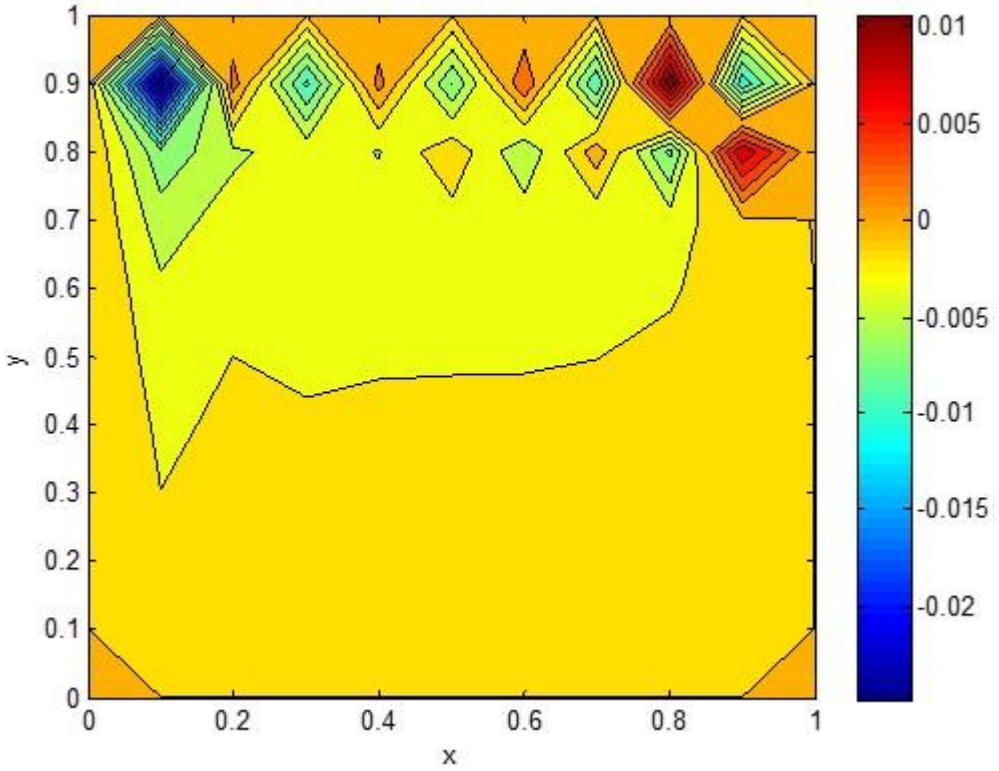


Fig 13: Pressure for $Re = 400$

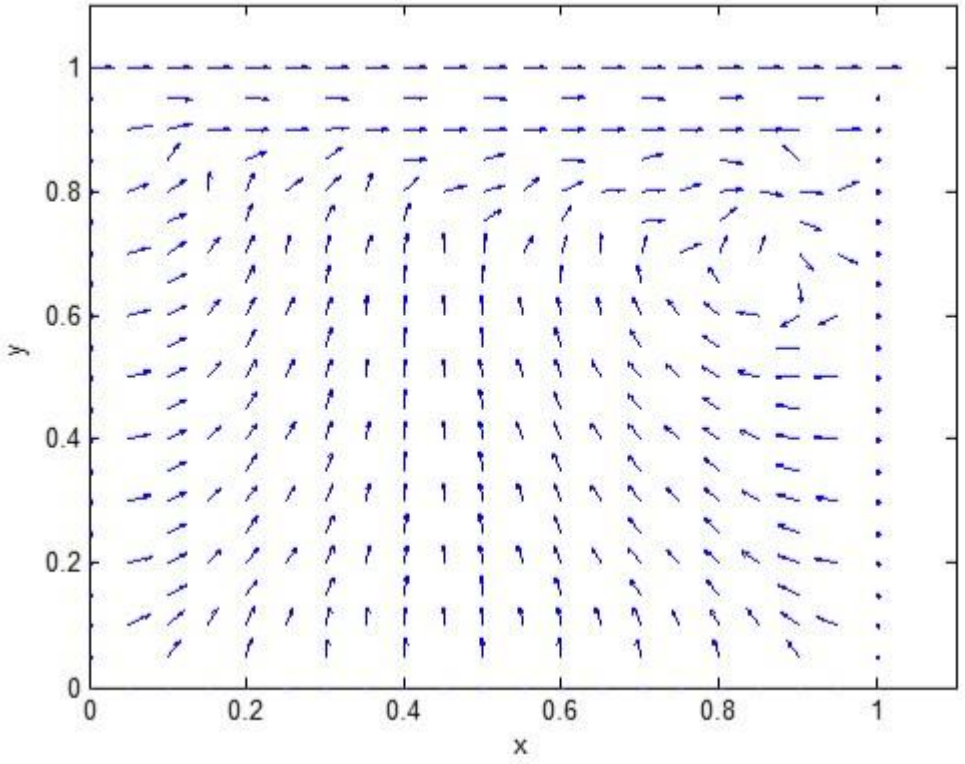


Fig 14: Velocity for $Re = 500$

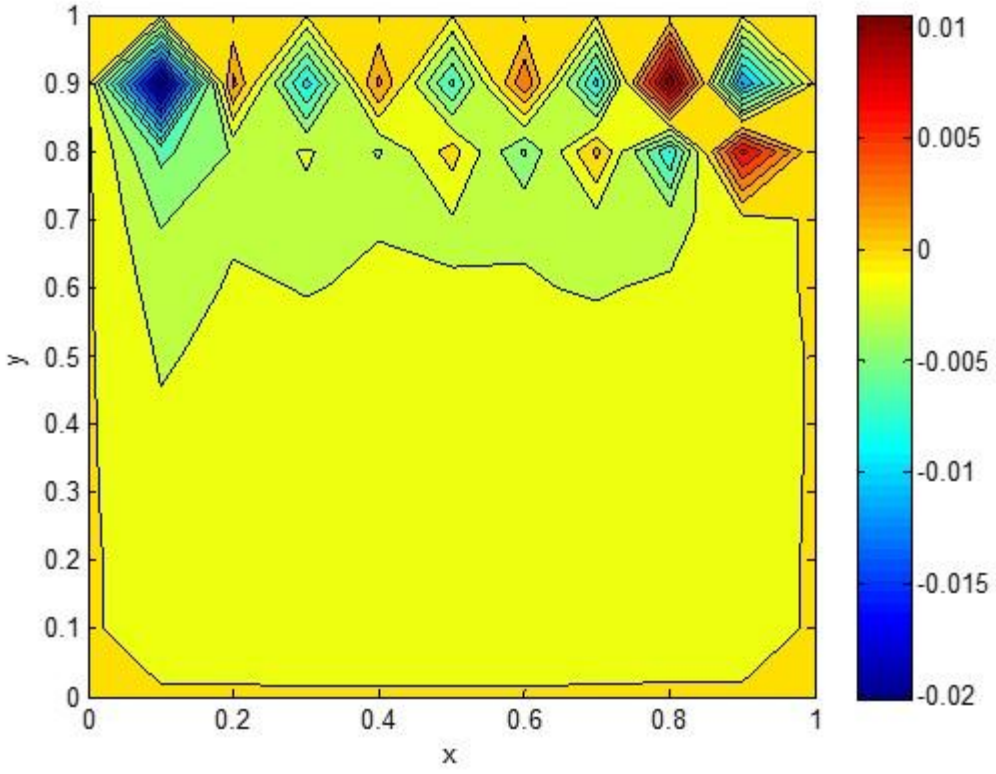


Fig 15: Pressure for $Re = 500$

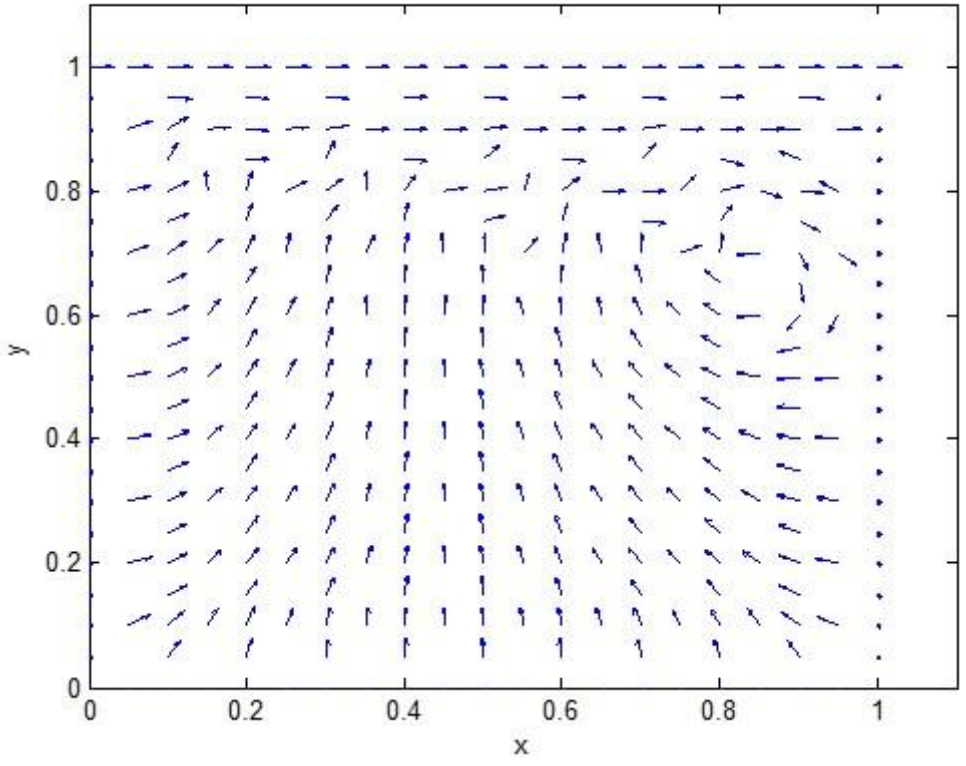


Fig 16: Velocity for $Re = 700$

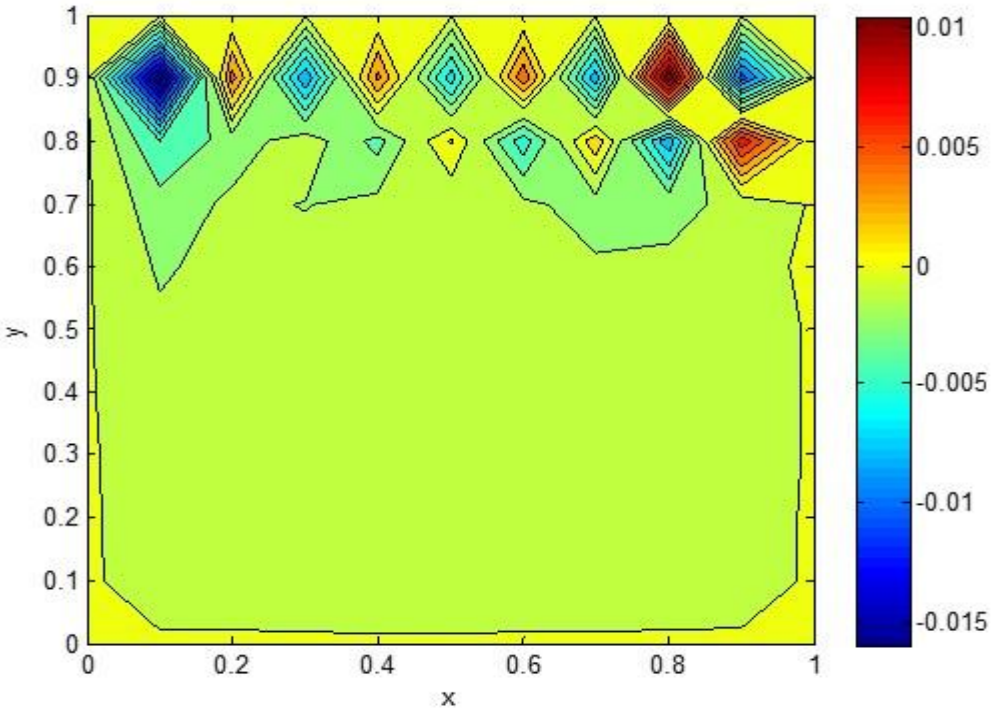


Fig 17: Pressure for $Re = 700$

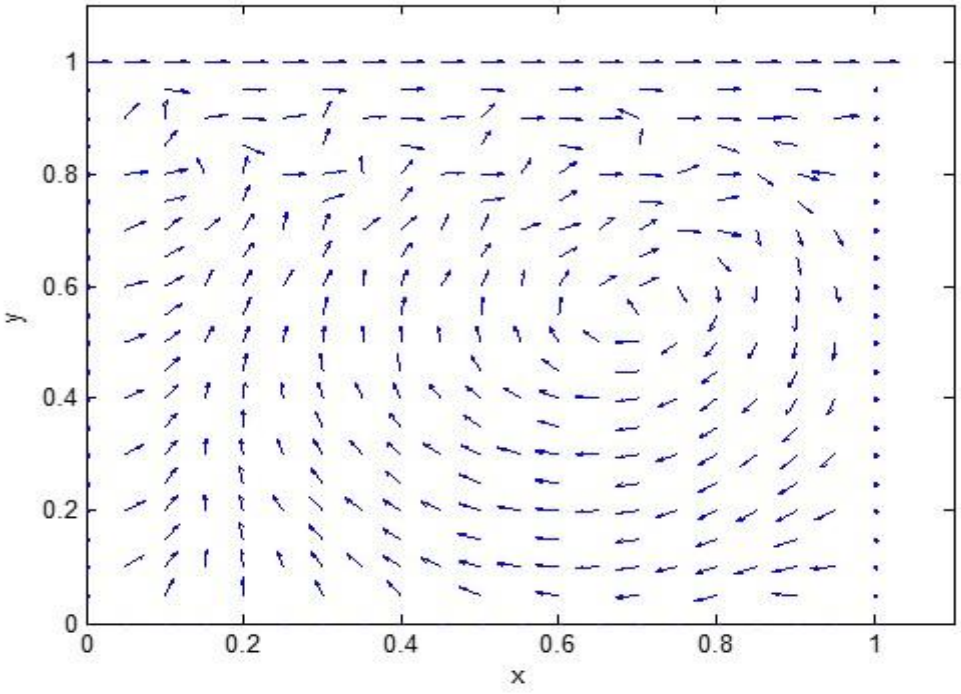


Fig 18: Velocity for $Re = 1000$

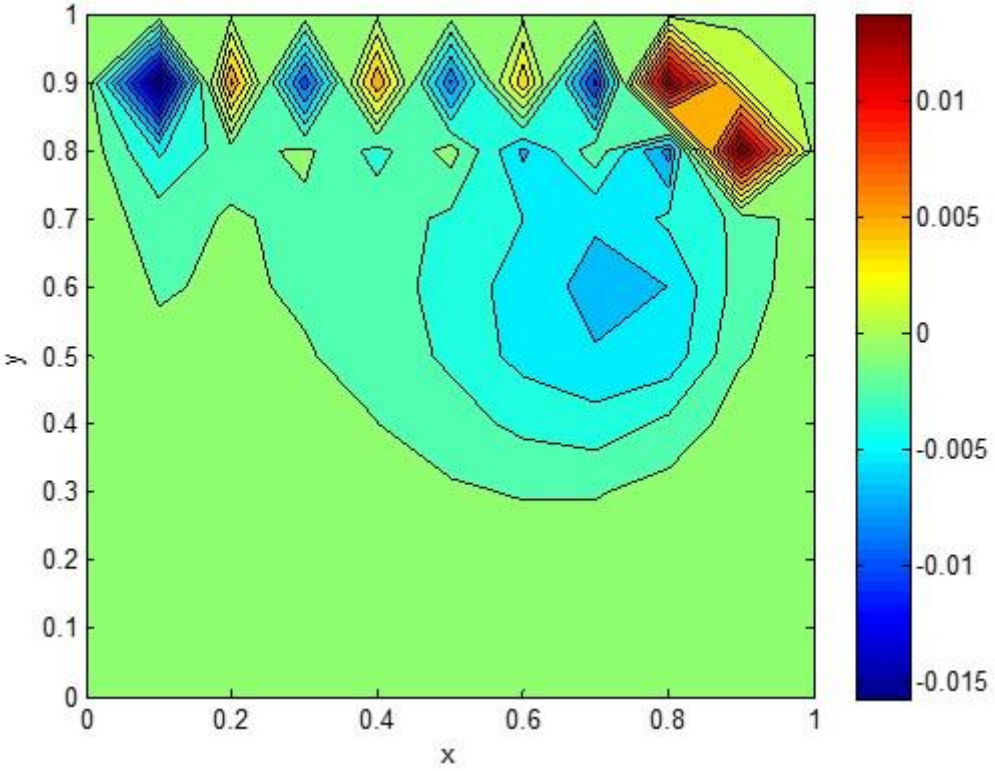


Fig 19: Pressure for $Re = 1000$

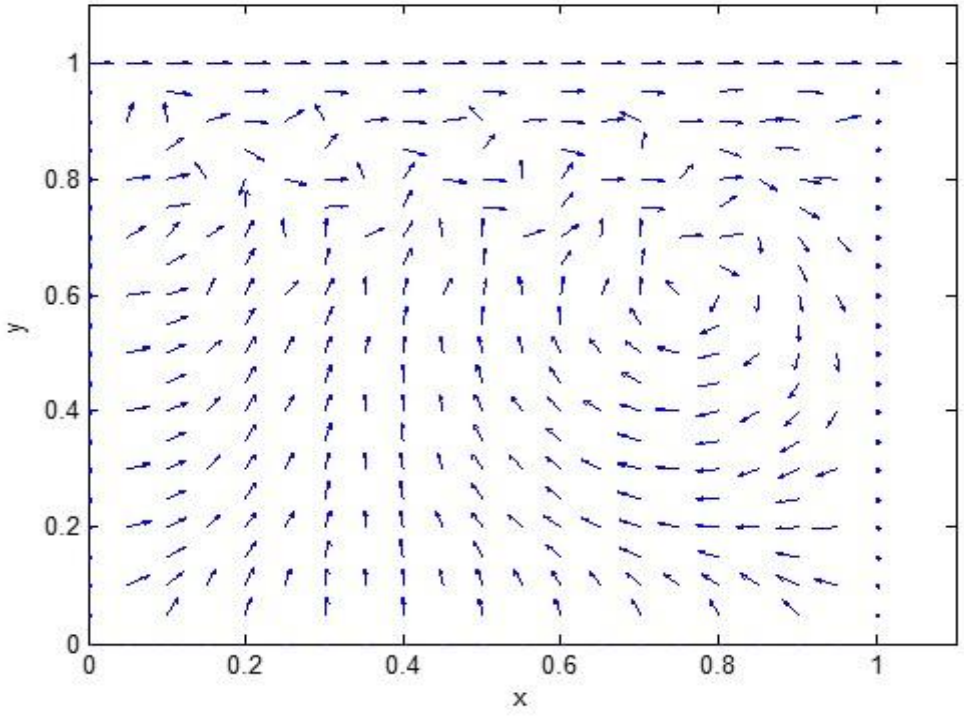


Fig 20: Velocity for $Re = 1200$

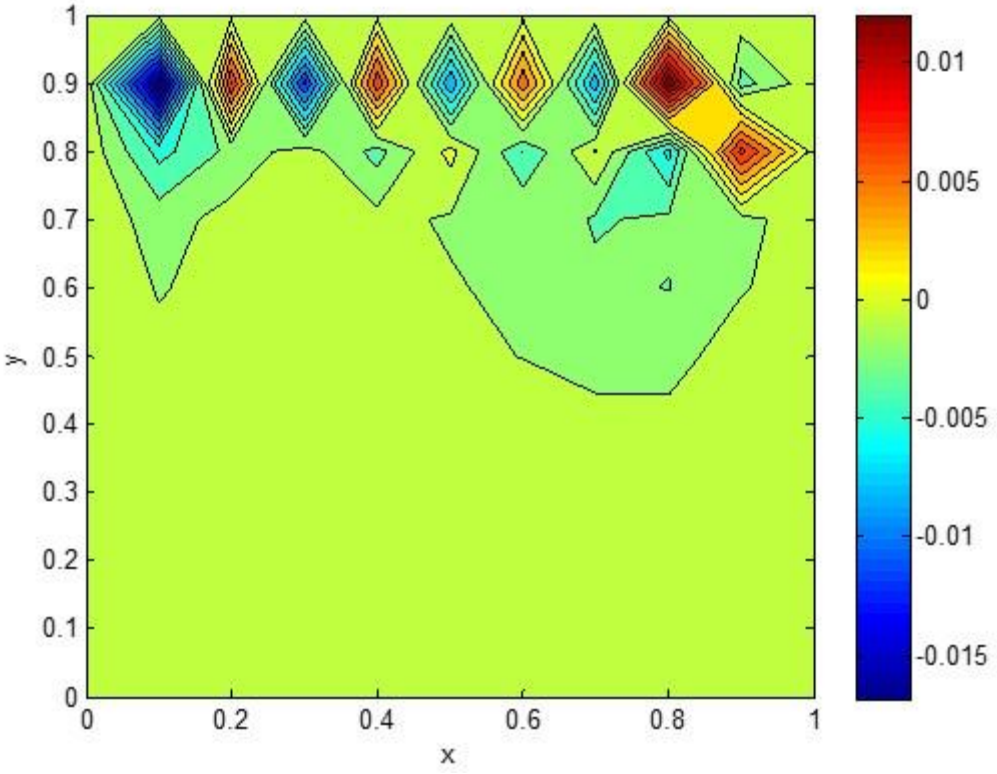


Fig 21: Pressure for $Re = 1200$

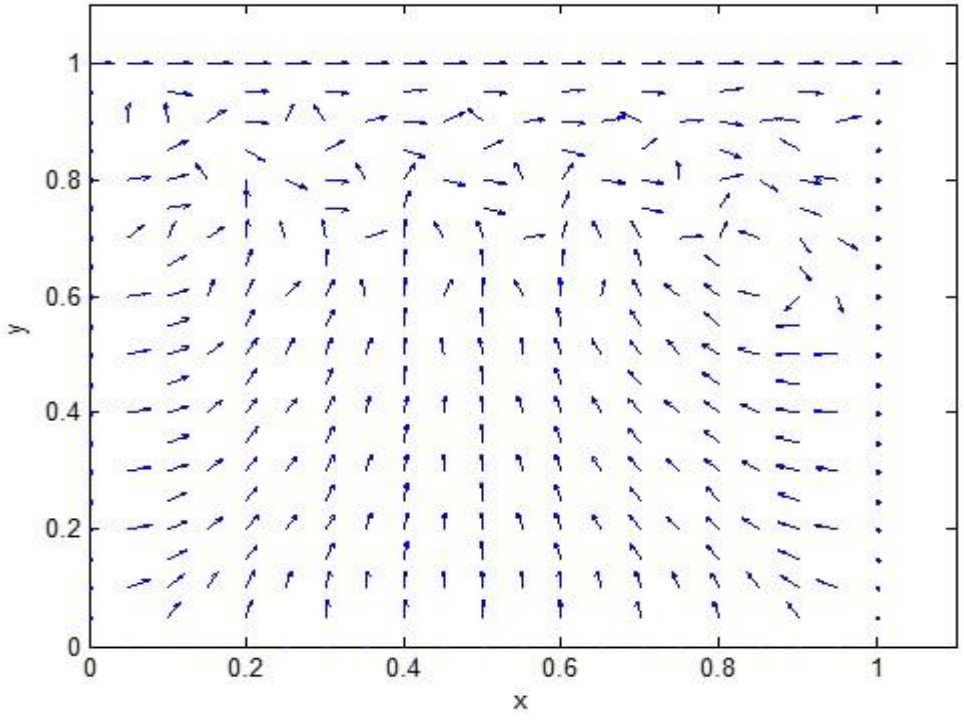


Fig 22: Velocity for $Re = 1400$

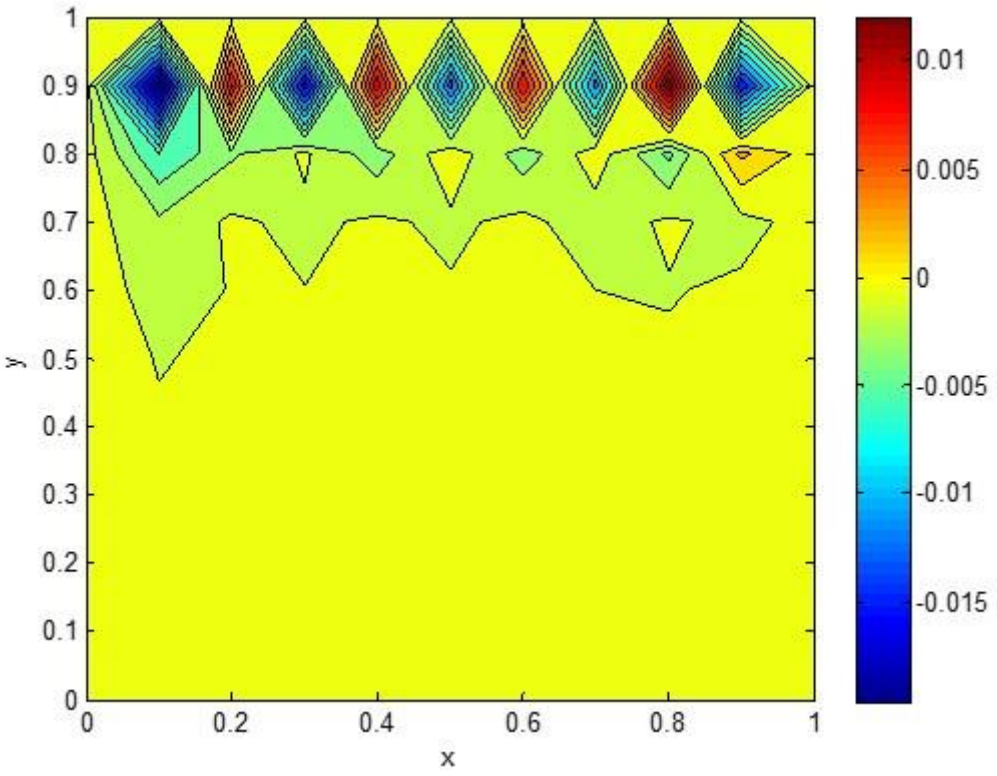


Fig 23: Pressure for $Re = 1400$

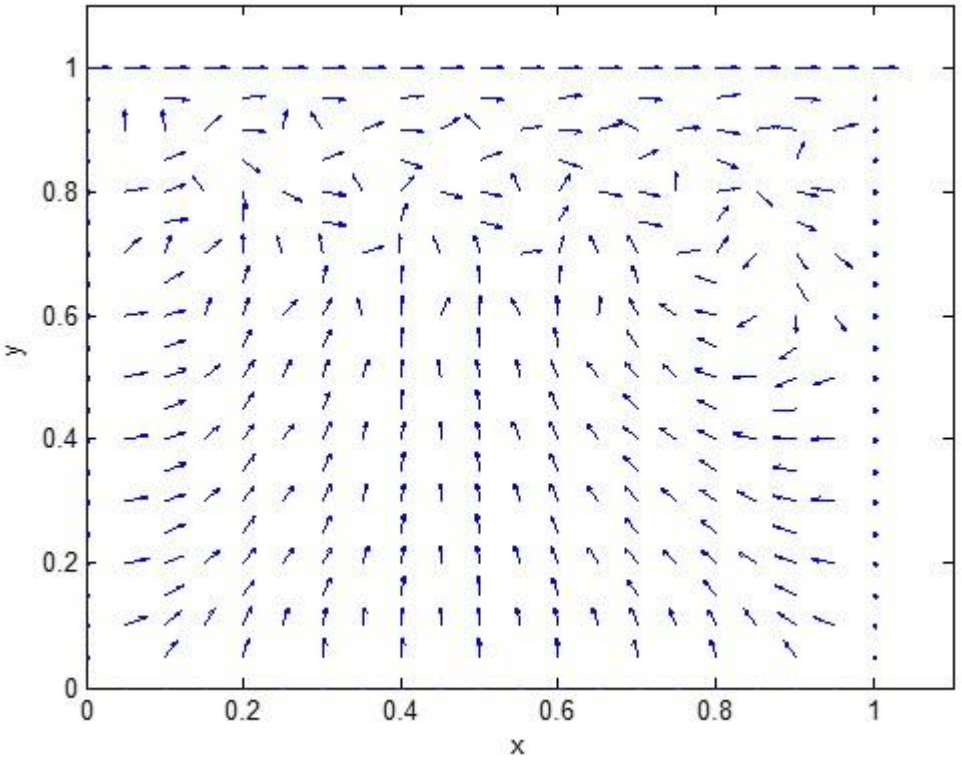


Fig 24: Velocity for $Re = 1500$

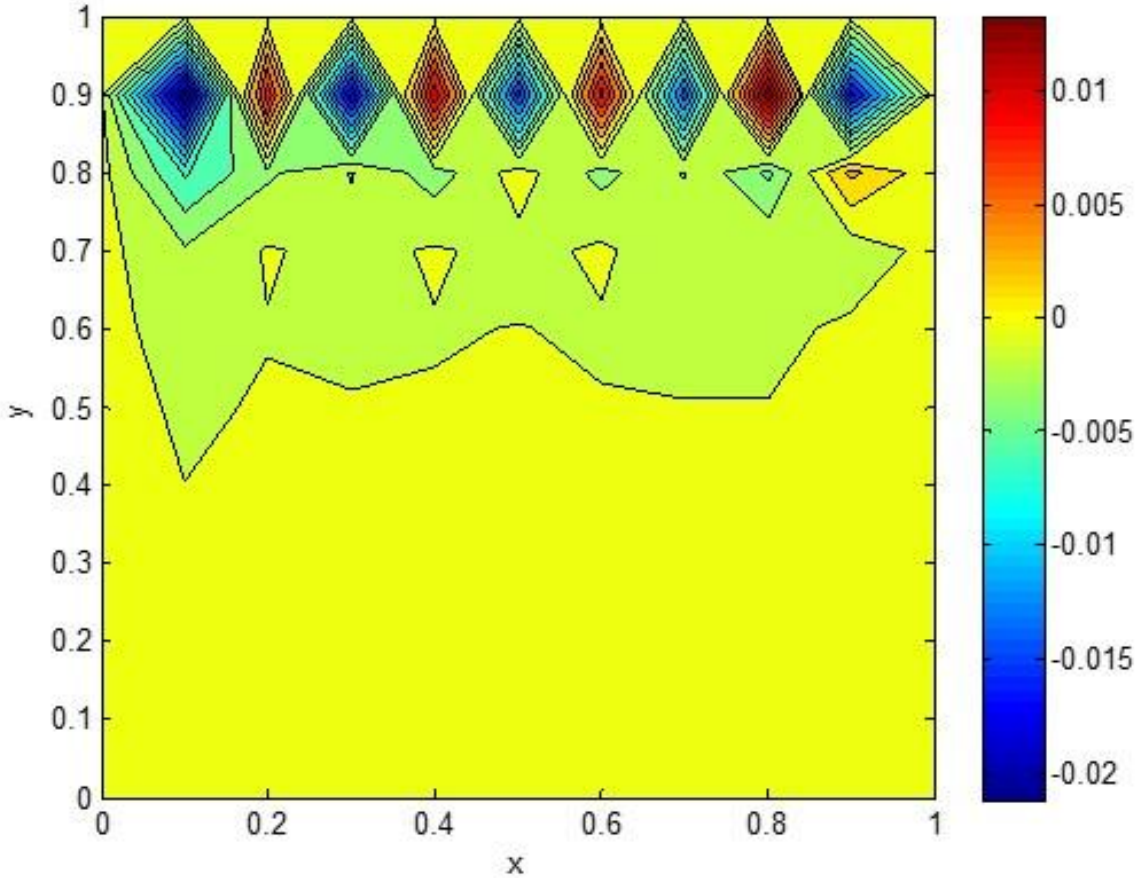


Fig 25: Pressure for $Re = 1500$

Conclusions

Used the square lid-driven cavity as a starting point to explain the 2D incompressible N-S equations for FEM. Dirichlet boundary criteria were applied to every border of each jurisdiction. Codes for finite element programming were developed to solve these equations. MATLAB is used to write these programming codes. The flow patterns are observed for multiple higher values of Re. The boundary condition and geometry codes are exceptional and incredibly effective.

Author Contributions: Haider Ali: Writing Original Draft, Review; Muhammad Amjad: Conceptualization; Tehreem: Software; Shah Jahan: Conceptualization; Muhammad Zaman: Validation; Aamir Hussain Khan: Validation; Muhammad Ramzan: Writing Original Draft.
Acknowledgments: Not Applicable
Declaration of Interest: The authors declare no conflict of interest.
Data Availability Statement: This manuscript has no associated data.
Funding: There is no funding associated with this manuscript.

References

[1] Cengel, Y. A., Cimbala, J. M., & Turner, R. H. (2014). Properties of fluids. *Fluid Mechanics: Fundamentals and Applications*, 37-73.

[2] Kwon, Y. W. (1991). Material nonlinear analysis of composite plate bending using a new finite element formulation. *Computers & structures*, *41*(5), 1111-1117.

[3] Jiajan, Wanchai (2010). The solution to incompressible Navier- Stokes equations by using the finite element method, MSc thesis, the University of Texas at Arlington.

[4] Ghia, U. K. N. G., Ghia, K. N., & Shin, C. T. (1982). High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of computational physics*, *48*(3), 387-411.

[5] Per-Olof Persson (2002). Implementation of Finite Element-Based Navier-Stokes Solver, 2.094 – Project, MIT.

[6] Glaisner, F. and Tezduyar, T.E. (1987). Finite Element Techniques for the Navier-Stokes Equations in the Primitive Variable Formulation and the Vorticity Streamfunction Formulation, Department of Mechanical Engineering, University of Houston.

[7] Taylor, C. and Hughes, T.G. (1981). Finite Element Programming of Navier-Stokes Equations, Swansea, U.K., Pineridge Press Ltd.

[8] Rhaman, M. M. and Helal, K. M. (2014).Numerical Simulations of Unsteady Navier-Stokes Equations for Incompressible Newtonian Fluids using FreeFem++ based on Finite Element Method, Annals of Pure and Applied Mathematics Vol. 6, No. 1, 70-84.

[9] Simpson, G. (2017). *Practical finite element modeling in earth science using matlab*. John Wiley & Sons.

[10] Khennane, Amar (2013). Introduction to Finite Element Analysis Using MATLAB and Abaqus, Taylor & Francis Group, LLC.

[11] Tsega, E. G., & Katiyar, V. K. (2018). Finite element solution of the two-dimensional incompressible navier-stokes equations using matlab. *Applications and Applied Mathematics: An International Journal (AAM)*, *13*(1), 35.

[12] Vajjala, K. S., Sengupta, T. K., & Mathur, J. S. (2020). Effects of numerical anti-diffusion in closed unsteady flows governed by two-dimensional Navier-Stokes equation. *Computers & Fluids*, *201*, 104479.

[13] Bedrossian, J., & Vicol, V. (2022). *The mathematical analysis of the incompressible Euler and Navier-Stokes equations: an introduction* (Vol. 225). American Mathematical Society.