Vol.6 (Iss. 3) Mathematics (2025), pp. 583-614 African Journal of Empirical Research http://ajernet.net ISSN 2709-2607

Solutions of Navier Stokes Equations for Dam Break Problem in Two Dimension Using Finite Element Method

Owen Mulinya Kizito*[1]
David Angwenyi[2]
Duncan Oganga[3]

[1,2,3]Department of Mathematics, Masinde Muliro University of Science and Technology, Kenya

*[1]Corresponding Author

https://doi.org/10.51867/ajernet.6.3.46

Abstract

A dam break is one of the most catastrophic events in hydraulic systems it happens when a dam suddenly fails, unleashing massive amounts of water in an uncontrolled rush. Even though water covers most of the Earth, water scarcity continues to be a serious challenge, especially in regions that rely heavily on irrigation. In response, many governments have invested in large scale dam construction to support food security by irrigating over 600,000 hectares of dry and semi-arid land. While dams are essential for water storage and agricultural productivity, they also come with significant risks. The enormous potential energy they store can lead to devastating environmental and social consequences if a failure occurs. This study focuses on modeling and simulating dam break scenarios using the two-dimensional Navier-Stokes equations, widely recognized for describing fluid behavior, solved through the Galerkin finite element method in MATLAB. The simulation considers steady-state, incompressible Newtonian fluids without body forces and applies the classic lid-driven cavity problem for benchmarking. To achieve accurate results, the study uses eight-noded rectangular elements, with quadratic interpolation for velocity and bilinear interpolation for pressure, resulting in 20 unknowns per element. The finite element method was selected over other numerical approaches because of its accuracy, especially when dealing with complex geometries. The simulation results align well with benchmark data across various Reynolds numbers, confirming the method's accuracy and reliability. These findings are valuable to the field of computational fluid dynamics (CFD), offering an effective way to simulate dam related fluid movement. More importantly, in the context of hydraulic engineering and disaster preparedness, the study provides critical insights into how dam failures evolve and how flood waters behave when released. This knowledge can inform smarter emergency planning, safer dam designs, and stronger public awareness for downstream communities, ultimately contributing to more resilient disaster risk reduction efforts.

keywords: Dam Break, Finite Element Method, Navier Stokes Equation, Fluid Mechanics

1 INTRODUCTION

A dam break is a structural calamity failure denoted by unexpected, swift, and uncontrolled release of enclosed water as a result of natural causes and human attributes. Dams store greater amount of potential energy which have a massive impact of a possible destruction on the civilian population and the environment. The failure of the Vajont Dam in Italy in 1963 led to the tragic loss of 2,600 lives

and in 1993 failure of Gouhou dam in China caused 300 deaths (1) and the 2018 Solai dam failure in Kenya leaving 2000 homeless and 48 dead (2). To minimize such risks, mitigation and prevention measure should be considered by prevailing information which increases safety awareness (3). Such information is obtained through the study of Navier-Stokes equations which are the governing equations in understanding Newtonian in-compressible fluids. The Navier-Stokes equations play a pivotal role in applied mathematics, describing the mechanics underlying a wide range of engineering and scientific phenomena. These equations are instrumental in modeling ocean currents, weather patterns, airflow around aircraft wings, and water flow in pipes. For this study, they are utilized to model a dam break. In both their simplified and complete forms, the Navier-Stokes equations are essential for vehicle and aircraft modeling, analyzing the behavior of dense liquids, studying pollution dispersion, designing power systems, and other fluid-related processes. This study specifically focuses on modeling the dynamics of a dam break.

Toro (2024)(4) presents two illustrative case studies relevant to dam-break mathematical modelling, combining numerical, theoretical, and experimental approaches. The first involves a two-dimensional circular dam with a known one-dimensional radial reference solution, allowing for rigorous evaluation of numerical methods. Dai et al. (2025)(5) used a shallow water model to examine how dam spacing, bed slope, and water depths affect dam-break flood dynamics. They found that larger dam spacing, steeper slopes, and higher upstream levels increase downstream impact pressure and velocity, while greater downstream depths reduce shear stress. The study adds valuable insight into downstream flood behavior often missed in prior research.

The Galerkin finite element formulation for the two-dimensional unsteady incompressible Navier-Stokes equations has been explored by (6) using a quadratic triangular element with six nodes. In this formulation, the pressure variable is positioned at the corner nodes, while the velocity components are located at all six nodes. The study utilized different meshes at various Reynolds numbers, with a focus on the artificial compressibility method. A Weak Galerkin Finite Element Method (WGFEM)(7) has been proposed for solving the Navier-Stokes equations (NSEs). The existence and uniqueness of the WGFEM solution for NSEs have been established. WGFEM delivers highly accurate numerical approximations for both the velocity and pressure fields, even at very high Reynolds numbers. A notable advantage of WGFEM is its flexibility in choosing the order of velocity and pressure compared to standard finite element methods.

Persson(8) implemented a finite element-based solver for the incompressible Navier-Stokes equations on unstructured two-dimensional triangular meshes, solving the lid-driven cavity flow problem for Reynolds numbers of 100, 500, 1000, and 2000. Glaisner et al. (9) discussed finite element procedures for the Navier-Stokes equations in both the primitive variable formulation and the vorticity stream-function formulations. They obtained steady-state solutions of lid-driven cavity flow using a velocity-pressure formulation with nine-node rectangular elements. Taylor (10) and Smith(11) employed eight-node rectangular element meshes to solve two-dimensional incompressible Navier-Stokes equations using the FORTRAN programming language. Rhaman et al. (12) presented a Galerkin finite element method to simulate fluid particle motion satisfying the unsteady Navier-Stokes equations through programming code developed in FreeFem++. Simpson (13) used nine-node rectangular elements with two degrees of freedom per node for finite element simulation of a coupled reaction-diffusion problem using MATLAB. Khennane (14) developed MATLAB codes for four-node and eight-node quadrilateral elements for the

linear elastic static analysis of two-dimensional problems using the finite element method.

Khani Aminjan and Domiri Ganji (2025)(15) conducted experimental and numerical studies on tangential input pressure-swirl atomizers, focusing on the effects of inlet pressure and Reynolds number. They found that increasing the Reynolds number from 33,832 to 198,524 led to notable increases in spray angle highlighting the Reynolds number's significant role in spray dynamics. Li, Klausner, and Mei (2025)(16) produced high-accuracy benchmark results for 2D lid-driven cavity flow at Reynolds numbers up to 30,000, demonstrating that with fine grids and careful error control, steady incompressible flow solutions can be achieved even at high Reynolds numbers offering valuable reference data for validating CFD models.

This study focuses on dam-break modelling using the finite element method (FEM) to solve the incompressible Navier-Stokes equations, with special attention to flow behavior across Reynolds numbers of 1, 50, 100, 200, and 1000. FEM offers high accuracy for simulating complex fluid dynamics, particularly in capturing transitional flow characteristics relevant to dam-break scenarios. By analyzing flow evolution at these Reynolds numbers, the study aims to enhance understanding of dam-break behavior and contribute to the development of reliable CFD tools for hydraulic engineering and flood risk mitigation.

2 INCOMPRESSIBLE NAVIER STOKES EQUATIONS

2.1 Governing Equations

The governing equations are the steady incopressible Navier-Stokes equations. These equations are the energy, momentum, and continuity equations in their most conservative version. The collection of nonlinear partial differential equations with velocities as components make up the momentum equations. Under isothermal condition, just the continuity equation and the incompressible Navier Stokes equations need to be solved (17). In cartesian coordinates, the 2-D Naviers Stoke equation are represented as follows

Continuity equation

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.1}$$

Momentum equations

x-component

$$\rho\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial u^2}{\partial y^2}\right) \tag{2.2}$$

y-component

$$\rho\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) = -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial v^2}{\partial y^2}\right) \tag{2.3}$$

where u and v represents the velocity components in x and y, p is the static pressure, ρ is the density, and μ is the viscosity . It should be noted that the body forces do not appear in (2) and (3) because

they are grouped with the pressure terms.

Using L and V as a characteristic length and velocity respectively, the dimensionless variables can be defined as

$$x^* = \frac{x}{L}, \quad y^* = \frac{y}{L}, \quad u^* = \frac{u}{V}, v^* = \frac{v}{V} \quad and \quad p^* = \frac{p}{\rho V^2}. \tag{2.4}$$

The governing equations, Equation 1 - 3 can be written in the dimensionless form without the astrix(*) as:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.5}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial u^2}{\partial y^2}\right) \tag{2.6}$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial v^2}{\partial y^2}\right) \tag{2.7}$$

where $Re = \frac{\rho VL}{\mu}$ is the Reynolds number.
The equations clearly contain a nonlinear term, specifically arising from the convective inertia term. The presence of nonlinearity adds complexity to the problem, necessitating the application of suitable formulations for resolution.

2.2 Finite Element Method

Step 1: Discretizing into weak form, the weak forms of equations (5) -(7) are expressed as follows.

$$\int\int_{\Omega} M\left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} +\right] dA = 0 \tag{2.8}$$

$$\int\int_{\Omega} N\left[u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial u^2}{\partial y^2}\right)\right] dA = 0 \tag{2.9}$$

$$\int\int_{\Omega} N\left[u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial v^2}{\partial y^2}\right)\right] dA = 0 \tag{2.10}$$

Where M and N are weight functions that will be equated in the Ritz- Galerkin finite element models. The first weight function represents pressure and the second represents velocities components.

Step 2: Selection of the approximated function, the dependent variables u, v and p are clearly approximated by expansions form.

$$u = \sum_{i=1}^{8} N_i u_i \tag{2.11}$$

$$v = \sum_{i=1}^{8} N_i v_i \tag{2.12}$$

$$p = \sum_{i=1}^{4} M_i p_i \tag{2.13}$$

Where M and N are vectors of Interpolation functions,u and v are the vectors of nodal values of velocity components, and p are the vectors of nodal values of pressure. From the expansion of the approximate function (11)-(13)as weight functions. They can be substituted into the weak form of (8)-(10) and integrated using by part integration . As a result, the solution to the incompressible Navier Stokes equation in finite element form is given below.

$$\int\int_{\Omega} M_i \left[\sum_{i=1}^{8} \frac{\partial N_j}{\partial x} u_j + \sum_{i=1}^{8} \frac{\partial N_j}{\partial y} v_j \right] dA = 0 \tag{2.14}$$

Equation 9 can be written as

$$\begin{aligned} \int\int_{\Omega} N_i \Bigg[\left(\sum_{k=1}^{8} N_k u_k \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j \right) + \left(\sum_{k=1}^{8} N_k v_k \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} u_j \right) + \frac{\partial M_i}{\partial x} p_i \\ - \frac{1}{Re} \left(\sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial x^2} u_j + \sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial y^2} u_j \right) \Bigg] dA = 0 \end{aligned} \tag{2.15}$$

Using Gauss Divergence Theorem (18), we have

$$\begin{aligned} &\int\int_{\Omega} \frac{1}{Re} \left(N_i \sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial x^2} u_j + N_i \sum_{j=1}^{8} \frac{\partial^2 N_j}{\partial y^2} u_j \right) dA \\ &= - \frac{1}{Re} \left(\int\int_{\Omega} \frac{\partial N_i}{\partial x} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j dA + \int\int_{\Omega} \frac{\partial N_i}{\partial y} \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} u_j dA \right) \\ &\quad + \frac{1}{Re} \int_{\Gamma} N_i \sum_{j=1}^{8} \frac{\partial N_j}{\partial n} u_j dS = 0 \end{aligned} \tag{2.16}$$

Where Γ is the boundary of the element Ω,$n = (n_x, n_y)$is the unit outward normal vector to the element and $\frac{\partial N_j}{\partial n} = \frac{\partial N_j}{\partial n} n_x + \frac{\partial N_j}{\partial n} n_y$ is the directional derivative of N_j in the direction normal to the boundary Γ. Hence we have

$$\begin{aligned} \int\int_{\Omega} N_i \Bigg[\left(\sum_{k=1}^{8} N_k u_k \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j \right) + \left(\sum_{k=1}^{8} N_k v_k \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} u_j \right) + \frac{\partial M_i}{\partial x} p_i \\ + \frac{1}{Re} \left(\frac{\partial N_i}{\partial x} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j + \frac{\partial N_i}{\partial y} \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} u_j \right) \Bigg] dA \\ - \frac{1}{Re} \int_{\Gamma} N_i \sum_{j=1}^{8} \frac{\partial N_j}{\partial n} u_j dS = 0 \end{aligned} \tag{2.17}$$

For boundary condition, we have

$$Dirichilet \int\int_{\Omega} N_i \left[\left(\sum_{k=1}^{8} N_k u_k \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j \right) + \left(\sum_{k=1}^{8} N_k v_k \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} u_j \right) + \frac{\partial M_i}{\partial x} p_i + \frac{1}{Re} \left(\frac{\partial N_i}{\partial x} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j + \frac{\partial N_i}{\partial y} \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} u_j \right) \right] dA = 0 \tag{2.18}$$

Applying similar similar procedure for Equation (10) we get

$$\int\int_{\Omega} N_i \left[\left(\sum_{k=1}^{8} N_k u_k \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} v_j \right) + \left(\sum_{k=1}^{8} N_k v_k \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} v_j \right) + \frac{\partial M_i}{\partial y} p_i + \frac{1}{Re} \left(\frac{\partial N_i}{\partial x} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} v_j + \frac{\partial N_i}{\partial y} \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} v_j \right) \right] dA = 0 \tag{2.19}$$

Table 1. Coefficient matrix formulas of incompressible Navier-Stokes equations

Abbr	**Formulas**	
a_{ij}^1	$\int\int_{\Omega} \left[N_i N_k u_k \frac{\partial N_j}{\partial x} + N_i N_k u_k \frac{\partial N_j}{\partial x} + \frac{1}{Re} \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) \right] dA$	i,j= 1,2,...8
a_{ij}^2	$\int\int_{\Omega} N_i \frac{\partial M_j}{\partial x} dA$	i=1,2,...8 j=1,2,3,4
a_{ij}^4	$\int\int_{\Omega} M_i \frac{\partial N_j}{\partial x} dA$	j=1,2,...8 i=1,2,3,4
a_{ij}^6	$\int\int_{\Omega} M_i \frac{\partial N_j}{\partial y} dA$	j=1,2,...8 i=1,2,3,4
a_{ij}^6	$\int\int_{\Omega} N_i \frac{\partial M_j}{\partial y} dA$	i=1,2,...8 j=1,2,3,4

Equation 14, which is the continuity equation in its weak form, and Equations 18 and 19, which are the momentum equations, can be represented in matrix form based on Table 1. This table contains the coefficients of the formulas a_{ij}^1, a_{ij}^2, a_{ij}^4, a_{ij}^6, a_{ij}^8, and a_{ij}^9. The order is arranged as u-p-v, as used by Smith (11). Thus, we obtain Equation 20.

$$\begin{bmatrix} a_{ij}^1 & a_{ij}^2 & 0 \\ a_{ij}^4 & 0 & a_{ij}^6 \\ 0 & a_{ij}^8 & a_{ij}^9 \end{bmatrix} \begin{bmatrix} u \\ p \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.20}$$

or

$$Ah = b \tag{2.21}$$

2.2.1 Quadratic Rectangular element(8-nodes)

The quadratic rectangular element (8-nodes) is chosen as the interpolation functions to directly compute the finite element matrix form using the exact integral formula(19).Let us denote an element by Ω. Shape functions for the rectangular elements are expressed in terms of local coordinates ξ and η where $\xi = 2(x - x_c)/l_x$ and $\eta = 2(y - y_c)/l_y$. Here, $(x_c, y_c)$is the centroid of the element, and l_x, l_y represent its length in x and y-direction.

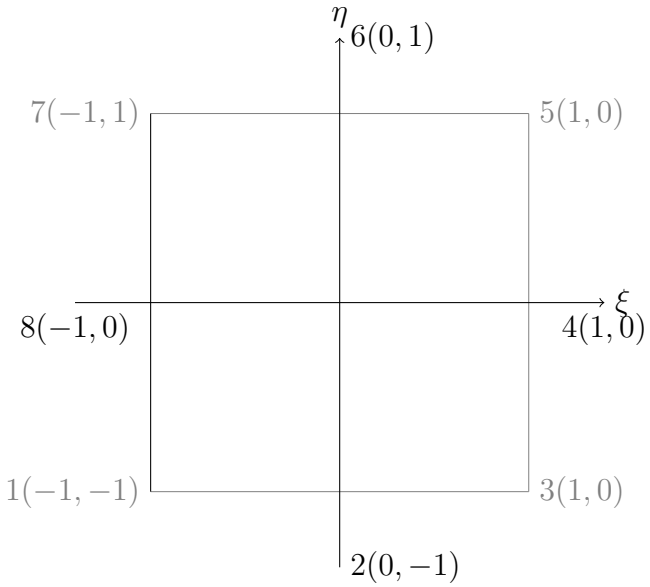

Fig. 2.1. Eight noded rectangular element.

This element comprises 16 unknowns for velocities and 4 unknowns for pressure. The expansions of dependent variables u, v and p are expressed as.

$$u = \sum_{i=1}^{8} N_i u_i \tag{2.22}$$

$$v = \sum_{i=1}^{8} N_i v_i \tag{2.23}$$

$$p = \sum_{i=1}^{4} M_i p_i \tag{2.24}$$

In order to construct the finite element matrix easily by using the exact integral formula, the interpolation functions are expressed in (22),(23) and (24). Using Figure (1) as reference to the local coordinates nodes or nodes 1,2,3,4,5,6,7,8. The general form of shape functions for 4-noded bilinear rectangular element(considering corner nodes) using local coordinates is

$$M = a + b\xi + c\eta + d\xi\eta \tag{2.25}$$

The general form of shape functions for 8-noded quadratic rectangular element(considering all nodes) using local coordinates is

$$N = a + b\xi + c\eta + d\eta^2 + e\xi\eta + f\eta^2 + g\xi^2\eta + h\xi\eta^2 \tag{2.26}$$

Using Kronecker -delta property (20) of shape functions, from equation (25) and (26)shape functions for 8-noded and 4-noded rectangular elements are

$$\begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \\ N_8 \end{bmatrix} = \begin{bmatrix} -\frac{1}{4}(1-\xi)(1-\eta)(1+\xi+\eta) \\ \frac{1}{2}(1-\xi^2)(1-\eta) \\ -\frac{1}{4}(1+\xi)(1-\eta)(1-\xi+\eta) \\ \frac{1}{2}(1+\xi)(1-\eta^2) \\ -\frac{1}{4}(1+\xi)(1+\eta)(1-\xi-\eta) \\ \frac{1}{2}(1+\xi^2)(1+\eta) \\ -\frac{1}{4}(1-\xi)(1+\eta)(1+\xi-\eta) \\ \frac{1}{2}(1-\xi^2)(1-\eta^2) \end{bmatrix} \tag{2.27}$$

And

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4}(1-\xi-\eta+\xi\eta) \\ \frac{1}{4}(1+\xi-\eta-\xi\eta) \\ \frac{1}{4}(1+\xi+\eta+\xi\eta) \\ \frac{1}{4}(1-\xi+\eta-\xi\eta) \end{bmatrix} \tag{2.28}$$

2.2.2 Derivatives and integrals using local coordinates

let N(ξ, η)be a shape function in terms of local coordinates. If x and y are the global coordinates, then

$$\frac{\partial N}{\partial \xi} = \frac{\partial N}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial N}{\partial y}\frac{\partial y}{\partial \xi}$$
$$\frac{\partial N}{\partial \eta} = \frac{\partial N}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial N}{\partial y}\frac{\partial y}{\partial \eta}$$

$$\begin{bmatrix} \frac{\partial N}{\partial \xi} \\ \frac{\partial N}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial N}{\partial x} \\ \frac{\partial N}{\partial y} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial N}{\partial \xi} \\ \frac{\partial N}{\partial \eta} \end{bmatrix} = J \begin{bmatrix} \frac{\partial N}{\partial x} \\ \frac{\partial N}{\partial y} \end{bmatrix}$$

where

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

is the Jacobian matrix of the tranformation of the global coordinate system to local coordinate system. then, we have

$$\begin{bmatrix} \frac{\partial N}{\partial x} \\ \frac{\partial N}{\partial y} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial N}{\partial \xi} \\ \frac{\partial N}{\partial \eta} \end{bmatrix} \tag{2.29}$$

2.2.3 Computing the Jacobian Matrix

Let $N_1(\xi,\eta), N_2(\xi,\eta), ..., N_n(\xi,\eta)$ be the shape functions for an element in local coordinates. If$(x_1,y_1),(x_2,y_2),..$ are the global coordinates of the nodes of the elements and (x,y) is the global coordinate of a point on the element then

$$x = N_1(\xi,\eta)x_1 + N_2(\xi,\eta)x_2 + ... + N_n(\xi,\eta)x_n$$
$$y = N_1(\xi,\eta)y_1 + N_2(\xi,\eta)y_2 + ... + N_n(\xi,\eta)y_n$$

Then

$$\frac{\partial x}{\partial \xi} = \frac{\partial N_1}{\partial \xi}x_1 + \frac{\partial N_2}{\partial \xi}x_2 + ... + \frac{\partial N_n}{\partial \xi}x_n$$
$$\frac{\partial y}{\partial \xi} = \frac{\partial N_1}{\partial \xi}y_1 + \frac{\partial N_2}{\partial \xi}y_2 + ... + \frac{\partial N_n}{\partial \xi}y_n$$
$$\frac{\partial x}{\partial \eta} = \frac{\partial N_1}{\partial \eta}x_1 + \frac{\partial N_2}{\partial \eta}x_2 + ... + \frac{\partial N_n}{\partial \eta}x_n$$
$$\frac{\partial y}{\partial \eta} = \frac{\partial N_1}{\partial \eta}y_1 + \frac{\partial N_2}{\partial \eta}y_2 + ... + \frac{\partial N_n}{\partial \eta}y_n$$

Hence the Jacobian matrix of the transformation J is

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \cdots & \frac{\partial N_n}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \cdots & \frac{\partial N_n}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ . & . \\ . & . \\ . & . \\ x_n & y_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} \frac{\partial N_i}{\partial \xi}x_i & \sum_{i=1}^{n} \frac{\partial N_i}{\partial \xi}y_i \\ \sum_{i=1}^{n} \frac{\partial N_i}{\partial \eta}x_i & \sum_{i=1}^{n} \frac{\partial N_i}{\partial \eta}y_i \end{bmatrix} \tag{2.30}$$

2.2.4 Picard linearization method

Several methods exist to address the nonlinearity issue in this study, and Picard iteration will be employed. Due to the nonlinearity, the resulting set of algebraic equations cannot be solved in a single step, necessitating an iterative approach. In such iterative solutions, nonlinear terms can be linearized in various ways.The non linear terms will be replaced with

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} \longrightarrow \bar{u}\frac{\partial u}{\partial x} + \bar{v}\frac{\partial u}{\partial y},$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} \longrightarrow \bar{u}\frac{\partial v}{\partial x} + \bar{v}\frac{\partial v}{\partial y}$$

where $\bar{u}$ and $\bar{v}$ are approximate values of velocity components. We assume starting values $u_{01}, u_{02}, ..., u_{08}$ and $v_{01}, v_{02}, ..., v_{08}$ for the element and

$$\bar{u} = \sum_{k=1}^{8} N_k u_{0k},$$

$$\bar{v} = \sum_{k=1}^{8} N_k v_{0k},$$

The iteration process continues by replacing u_{0k} and $v_{0k}, k = 1, 2, ...8$, by the average velocity component values from the previous two iterations until tolerance is satisfied.Therefore equations 18 and 19 can be written as

$$\int\int_{\Omega} N_i \left[\left(\bar{u} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j \right) + \left(\bar{v} \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} u_j \right) + \frac{\partial M_i}{\partial x} p_i + \frac{1}{Re} \left(\frac{\partial N_i}{\partial x} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} u_j + \frac{\partial N_i}{\partial y} \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} u_j \right) \right] dA = 0 \tag{2.31}$$

$$\int\int_{\Omega} N_i \left[\left(\bar{u} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} v_j \right) + \left(\bar{v} \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} v_j \right) + \frac{\partial M_i}{\partial y} p_i + \frac{1}{Re} \left(\frac{\partial N_i}{\partial x} \sum_{j=1}^{8} \frac{\partial N_j}{\partial x} v_j + \frac{\partial N_i}{\partial y} \sum_{j=1}^{8} \frac{\partial N_j}{\partial y} v_j \right) \right] dA = 0 \tag{2.32}$$

This makes the new coefficient for $a_{ij}^1 and a_{ij}^9$ to be represented as

$$\int\int_{\Omega} \left[N_i \bar{u} \frac{\partial N_j}{\partial x} + N_i \bar{v} \frac{\partial N_j}{\partial x} + \frac{1}{Re} \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) \right] dA$$

2.2.5 Meshing

A structured 10 by 10 mesh is chosen, and sixteen elements are designated as blocks within the mesh. These blocks are strategically placed to simulate the presence of barriers or structural elements that influence the flow dynamics during a dam break scenario, in order to relate the steady Navier-Stokes equations to the dam break problem. Eighty-four elements remain accessible for the fluid flow simulation. There are 309 nodes in the mesh, and each one is essential to describing the pressure and velocity fields in the simulation area. The mesh's nodes are successively numbered in the $x-$direction to provide a methodical and well organized method of node indexing. Every node's degrees of freedom are arranged as $u - p - v$. This arrangement adheres to the standard set by (11).

In this setup, the upper compartment of the mesh is assumed to contain water, reflecting the initial condition of a reservoir or a similar body of water. The lower compartment is treated as a void, representing an empty space that will be filled by the flowing water once the dam break occurs. This compartmentalization is crucial for accurately simulating the dynamic interaction between the water and the void, capturing the rapid changes in velocity and pressure that characterize a dam break event.

3 RESULT AND DISCUSSIONS

After formulating the governing equations for each element, the next step involved assembling these equations to construct the global system of equations for the entire domain. This process incorporated

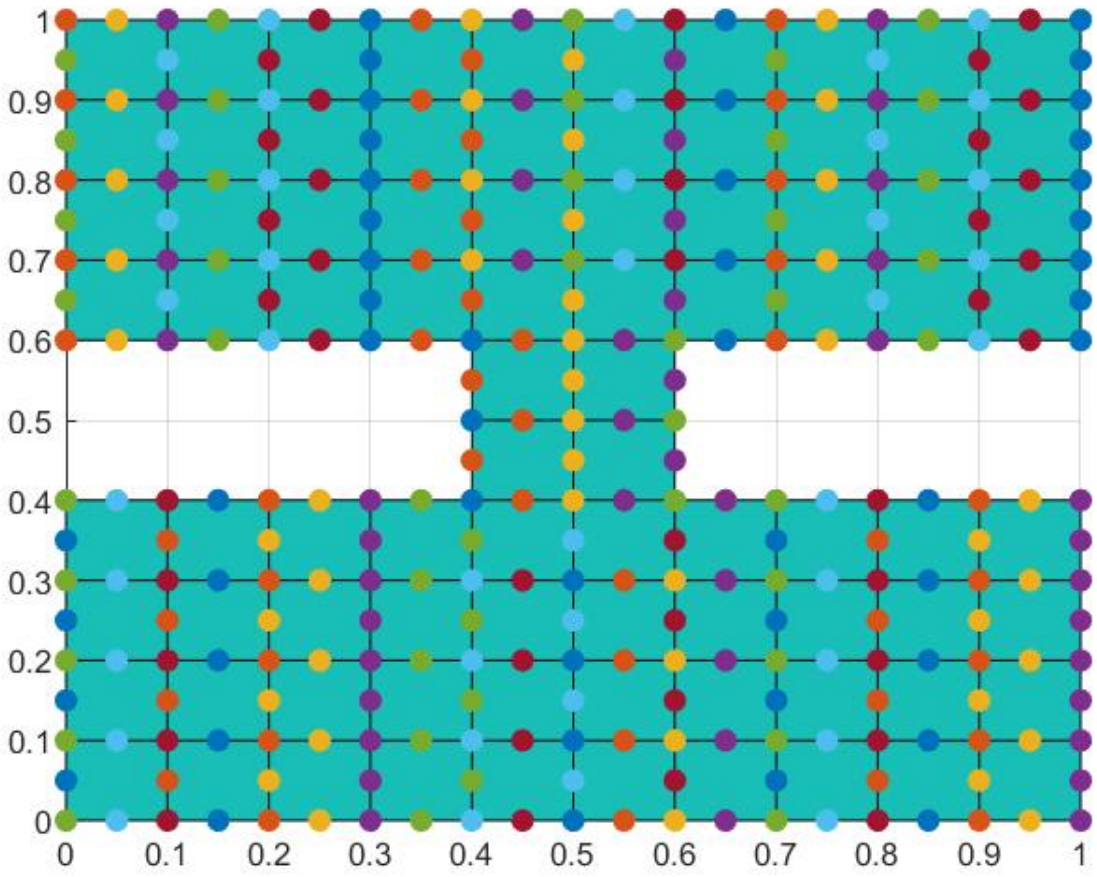

Fig. 2.2. Eight-noded rectangular elements Mesh with 309 nodes.

element equations, global node coordinates, element connectivity, and the global degrees of freedom associated with each node. By applying the necessary boundary conditions, a modified global system of equations was obtained. The MATLAB codes utilized for the solution process are provided in the appendix.

To illustrate the dam break flow problem discussed in this paper, we analyze a square lid-driven cavity flow with a unit length. With minor modifications, the upper four elements represent the dam, while the lower four elements simulate a void. The flow is driven by a unit vertical velocity applied at the top boundary, whereas the velocities along the remaining two opposite boundaries are set to zero. The velocity field is discretized using eight-node elements, while the pressure field is modeled using four-node elements. Simulations are conducted for Reynolds numbers of 1, 10, 50, 100, 200, 500, and 1000, employing a uniform mesh of 84 elements and 731 nodes, consisting of 319 nodes for u-velocity, 133 nodes for pressure, and 319 nodes for v-velocity. Additionally, 16 elements are adjusted into Block F, as illustrated in Figure 2. The numerical results are presented using quiver plots for velocity and contour plots for pressure at different Reynolds numbers. The simulations were performed using a finite element approach, ensuring a convergence tolerance. The chosen mesh configuration enables an in-depth analysis of velocity components and pressure distribution within the cavity, particularly in the context of the dam break problem. The results are visualized through velocity streamline plots and pressure contours, highlighting the influence of the mesh structure on the numerical solutions. For each Reynolds number, multiple iterations were conducted, each utilizing different time steps measured in seconds. The longest iteration required 23 seconds to complete.

Highly Laminar and Viscous-Dominated Flow: At Reynolds number 1, the flow remains smooth, slow-moving, and highly viscous. The dam break does not generate significant disturbances, and the velocity streamlines exhibit a well-ordered circulation pattern, with fluid motion dominated by viscous forces. The pressure distribution remains stable and symmetrical, with minimal gradients.

Stable Laminar Flow with Slight Disturbances: At Reynolds number 10, the flow still retains its laminar nature, but the fluid begins to move more freely as inertia effects start to emerge. The velocity streamlines indicate a stable vortex formation, and the pressure contours display clear yet mild variations. The dam's break effect is visible but not intense, as viscous forces still dominate.

Stronger Flow Circulation, Laminar but More Dynamic: At Reynolds number 50, the flow remains laminar, but circulation intensifies, with increased velocity gradients near the dam break. The velocity streamlines begin to stretch further, and pressure variations become more noticeable. The void left by the dam exhibits a smoother refilling motion, though turbulence is not yet present.

Transition to Turbulence, Developing Instabilities: At Reynolds number 100, the flow begins to exhibit unsteady characteristics. The velocity streamlines show oscillatory behavior, and small-scale vortices emerge near the dam break area. Pressure variations increase, and shear effects along the boundary layers become more evident. This represents the transition point between laminar and turbulent flow.

Onset of Turbulence, Vortex Formation Increases: At Reynolds number 200,the dam break generates a more chaotic motion, with multiple vortices appearing and evolving dynamically. The velocity field

becomes more complex, showing increased fluctuations and instability. Pressure contours display larger gradients, indicating stronger momentum interactions.

Fully Turbulent Flow, Strong Shear and Mixing:At Reynolds number 500, the flow is now highly turbulent, characterized by irregular velocity distributions and intense mixing. Large eddies and shear layers form near the dam break, contributing to rapid flow fluctuations. The pressure field shows large variations, reflecting high-energy interactions between fluid layers.

Highly Unstable and Turbulent, Dominated by Inertia: At Reynolds number 1000, the flow exhibits chaotic behavior, with continuous vortex shedding and energy dissipation. The velocity field is highly unsteady, with violent fluid motion and intense shear effects. Pressure variations become highly irregular, indicating a fully developed turbulent regime, with minimal influence from viscosity.

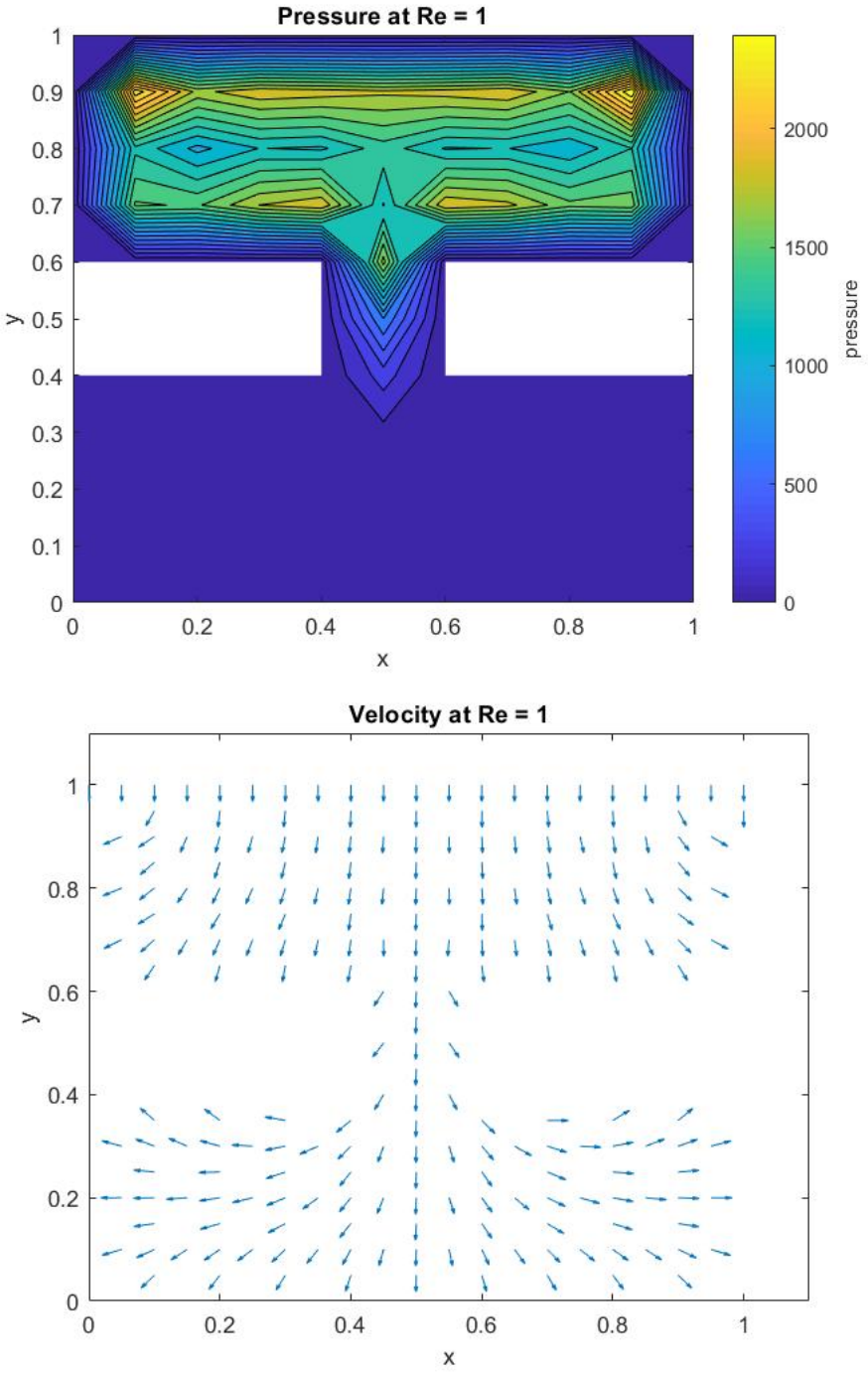

Fig. 3.1. Pressure contours and velocity stream line at Reynolds 1.

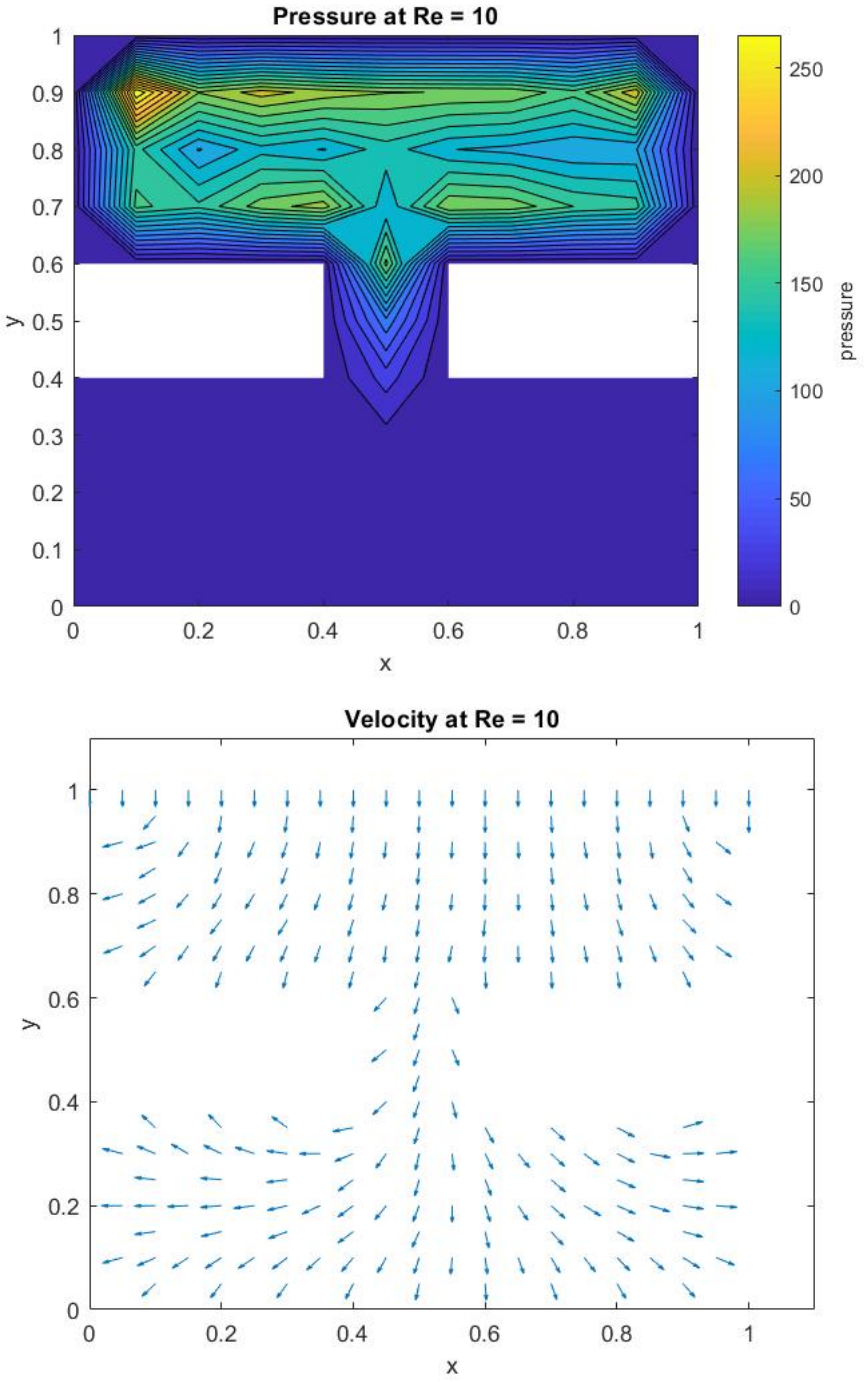

Fig. 3.2. Pressure contours and velocity stream line at Reynolds 10.

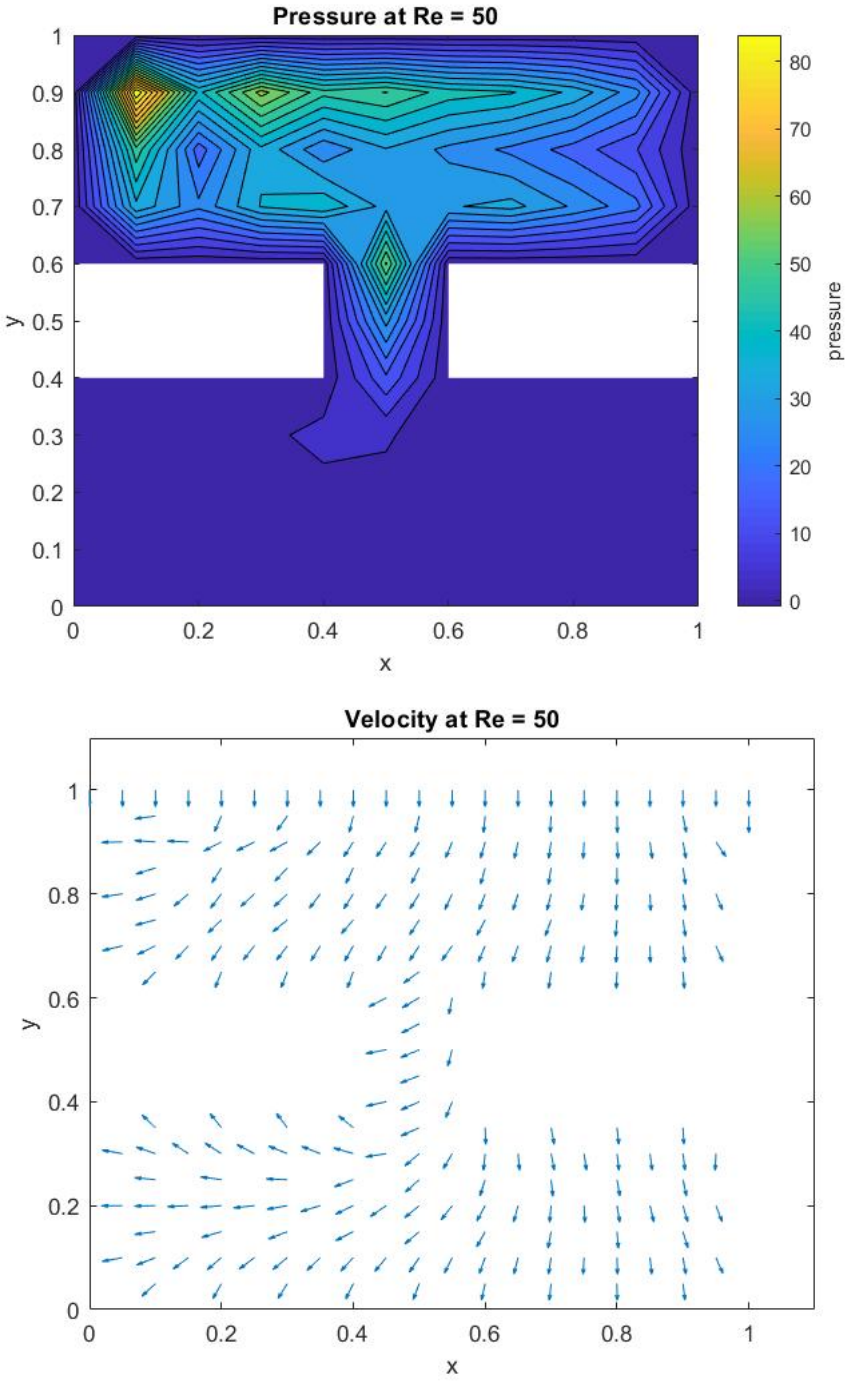

Fig. 3.3. Pressure contours and velocity stream line at Reynolds 50.

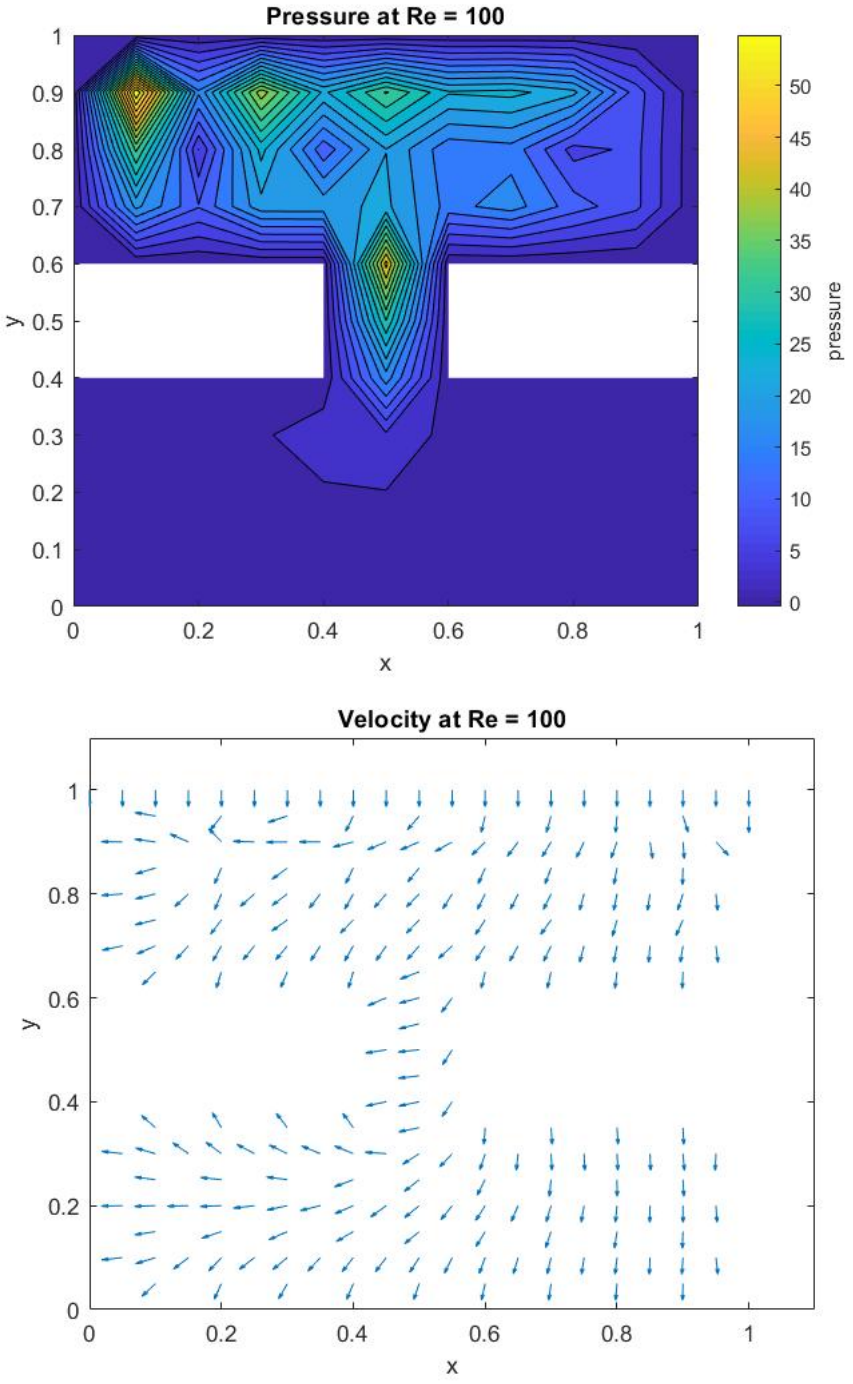

Fig. 3.4. Pressure contours and velocity stream line at Reynolds 100.

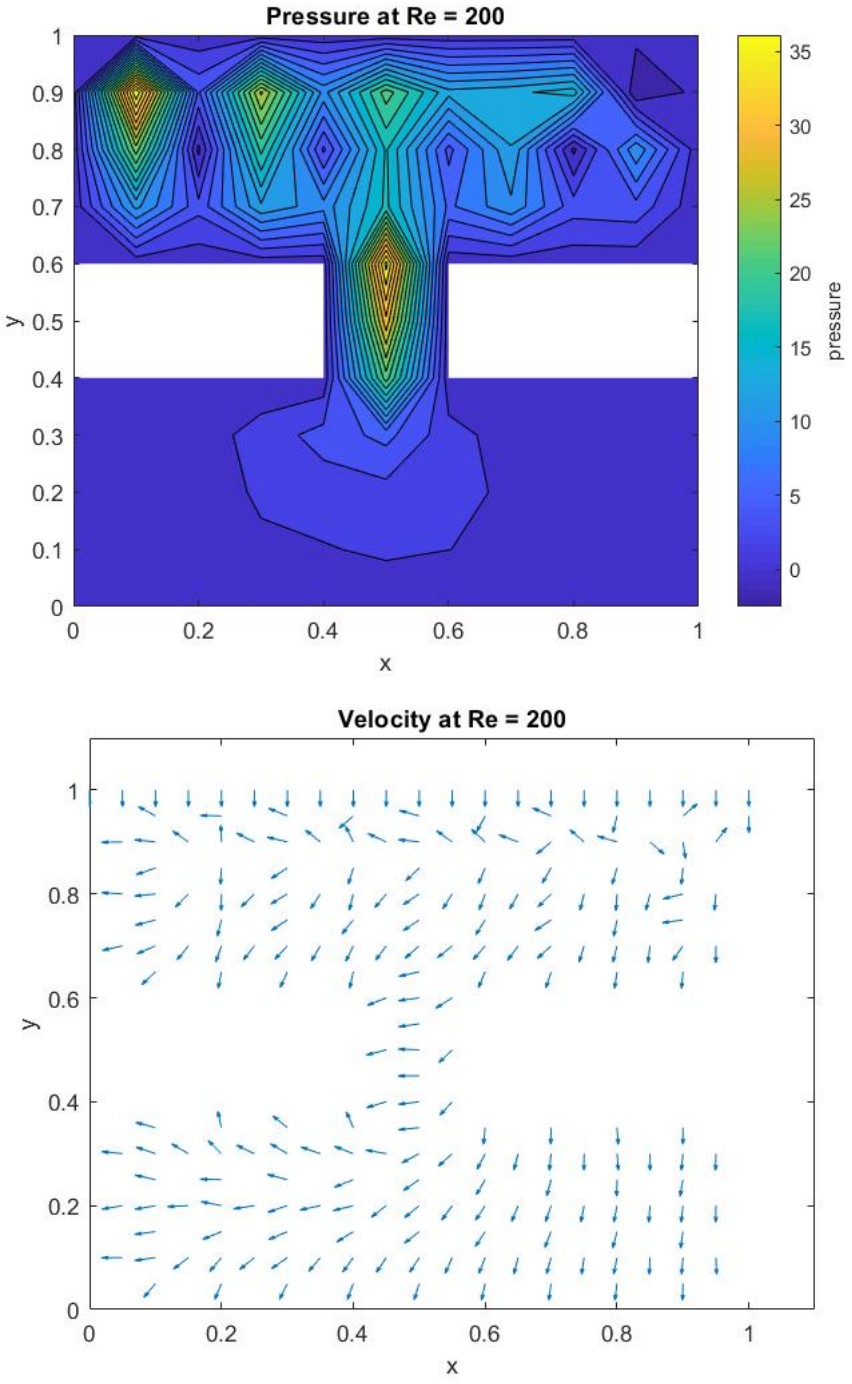

Fig. 3.5. Pressure contours and velocity stream line at Reynolds 200.

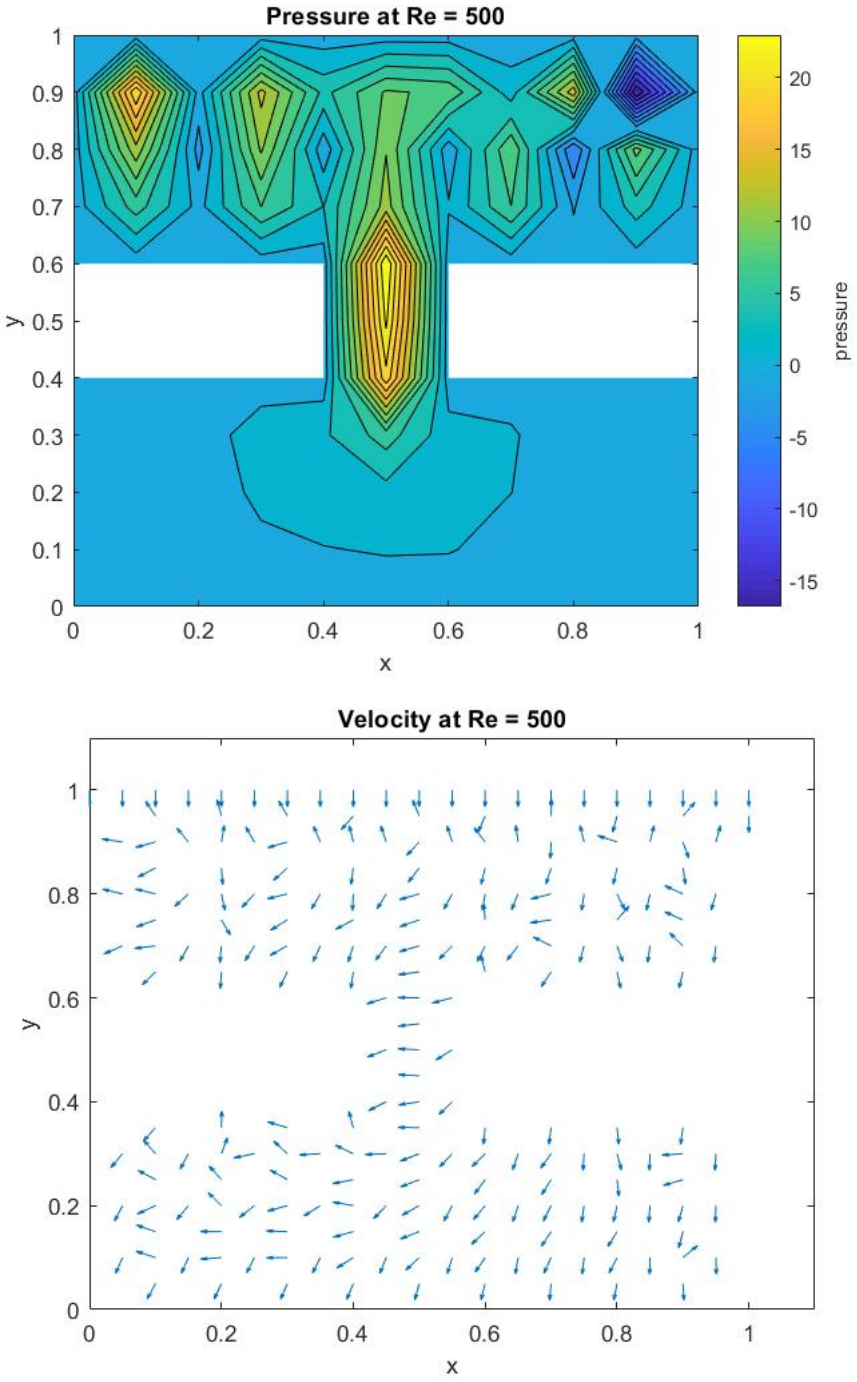

Fig. 3.6. Pressure contours and velocity stream line at Reynolds 500.

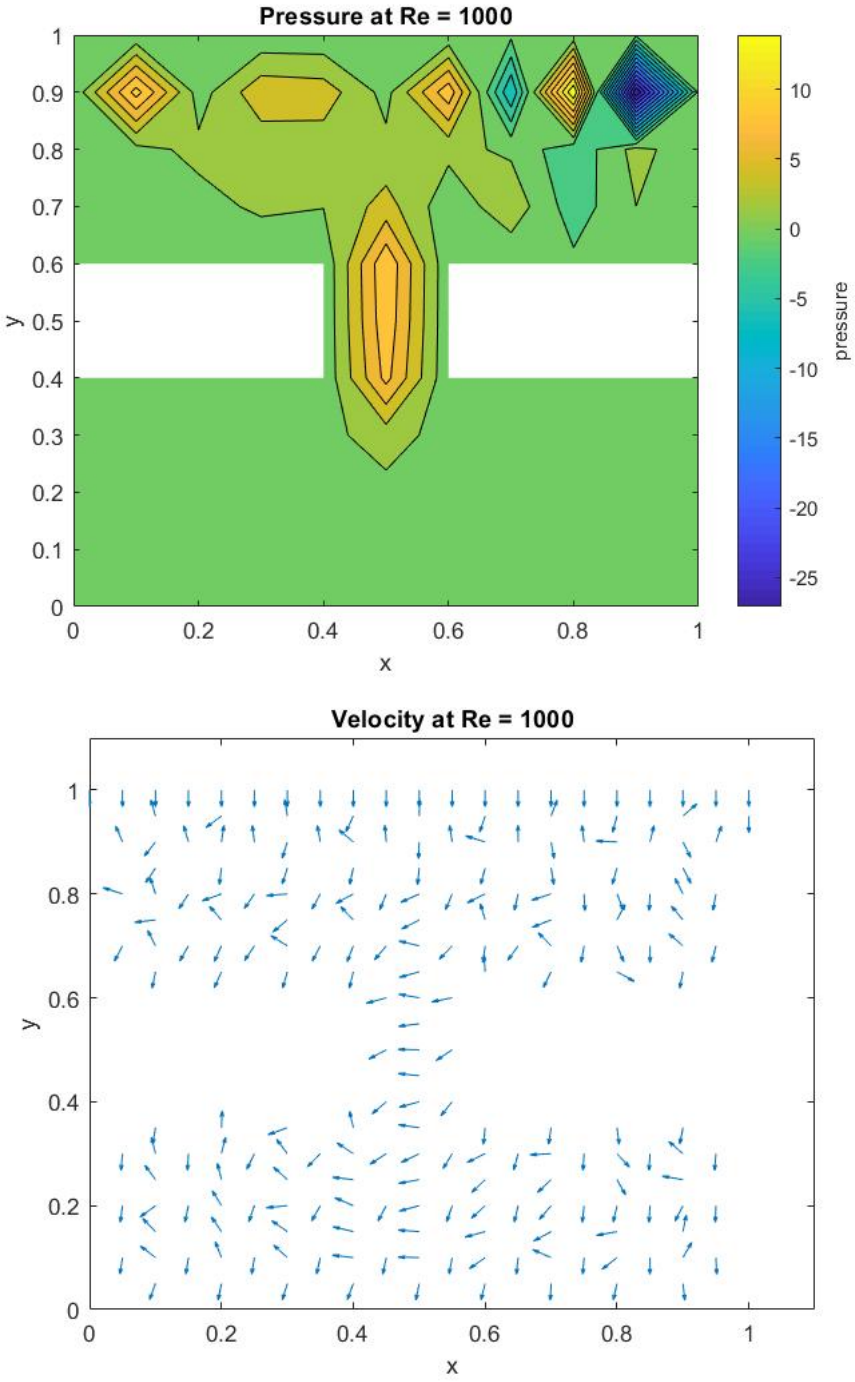

Fig. 3.7. Pressure contours and velocity stream line at Reynolds 1000.

4 Conclusions

In this paper, we explored the application of the finite element method, employing unequal-order interpolation for velocity and pressure within the conservative formulation of the two-dimensional, steady, incompressible Navier-Stokes equations. To achieve this, the coefficient matrices for both the continuity and momentum equations were derived by directly transforming the interpolation functions of a quadratic rectangular element.

To solve the steady incompressible Navier-Stokes equations, we implemented an iterative approach at each time step, utilizing the Picard linearization method to handle the nonlinear system. The finite element codes for these computations were developed in MATLAB, ensuring accuracy through convergence checks on velocity components and pressure. The program consists of a main module supported by multiple subprograms, which, with some modifications, can be adapted to address similar fluid flow challenges.

To validate the numerical method, we applied it to a well-known benchmark problem: the square lid-driven cavity flow. Dirichlet boundary conditions were enforced along all domain boundaries. For the dam-break scenario, we used a 10×10 mesh, incorporating barriers at specific points in the center of the left and right sections. The domain was structured into two distinct sections: the upper portion representing the dam and the lower portion serving as the open space where the released water flows. The numerical results were compared to the experimental results (21).

Author contributions

All authors discussed the results and contributed to the final manuscript.

Acknowledgments

The authors would like thank the reviewers for their valuable comments to improve our paper.

Financial disclosure

None reported.

Conflict of interest

The authors declare no potential conflict of interests.

References

[1] You, L., Li, C., Min, X., and Xiaolei, T. (2012). *Review of dam-break research of earth-rock dam combining with dam safety management.* Procedia Engineering, 28, 382-388.

[2] Murunga, P. A. (2019). *Influences of Perceived Crisis Response Strategies on Organizational Reputation*: *A Case of Solai Dam Tragedy in Kenya* : United States International University-Africa.

[3] Ahmadi, S. M., and Yamamoto, Y. (2021). *A New Dam-Break Outflow-Rate Concept and Its Installation to a Hydro-Morphodynamics Simulation Model Based on FDM (An Example on Amagase Dam of Japan).* Water, 13(13), 1759.

[4] Toro, E. F. (2024). *Computational Algorithms for Shallow Water Equations. Springer.*

[5] Dai, S., Wang, Y., Guo, J., Song, R., Shi, Z., Yang, S., ... and Jin, S. (2025). *Numerical simulation of dynamic process of dam-break flood and its impact on downstream dam surface. Water Resources Management*, 1-31.

[6] Jiajan, W. (2010). *Solution to incompressible Navier stokes equations by using finite element method*: The University of Texas at Arlington.

[7] Zhang, J., Zhang, K., Li, J., Wang, X. (2018). A weak Galerkin finite element method for the Navier-Stokes equations. Commun. Comput. Phys, 23(3), 706-746.

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

[9] Glaisner, F. (1987). *Finite element techniques for the Navier-Stokes equations in the primitive variable formulation and the vorticity stream-function formulation.*

[10] Taylor, C., and Hughes, T. G. (1981). *Finite element programming of the Navier-Stokes equations* (Vol. 14). Swansea: Pineridge press

[11] Smith, I. M., Griffiths, D. V., and Margetts, L. (2013). *Programming the finite element method. John Wiley and Sons.*

[12] 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, 6(1), 70-84.

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

[14] Khennane, A. (2013). *Introduction to finite element analysis using MATLAB® and abaqus.* CRC Press.

[15] Khani Aminjan, K., and Domiri Ganji, D. (2025). Experimental and numerical study on inlet pressure and Reynolds number in tangential input pressure-swirl atomizer. Arabian Journal for Science and Engineering, 1-20.

[16] Li, L., Klausner, J. F., Mei, R. (2025). Flow structures in two-dimensional lid-driven cavity flow: Benchmark numerical results for steady flows. European Journal of Mechanics-B/Fluids, 114, 204313.

[17] Reddy, J. N. (2019). *Introduction to the finite element method*. McGraw-Hill Education.

[18] Benci, V., and Baglini, L. L. (2014). *A generalization of Gauss' divergence theorem*. arXiv preprint arXiv:1406.4349.

[19] Tsega, E. G., and 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.

[20] Meyer, R. A., Swartworth, W., Woodruff, D. P. (2025). Understanding the Kronecker Matrix-Vector Complexity of Linear Algebra. arXiv preprint arXiv:2502.08029.

[21] Fraccarollo, L., and Toro, E. F. (1995). *Experimental and numerical assessment of the shallow water model for two-dimensional dam-break type problems.* Journal of hydraulic research, 33(6), 843-864.

A FINITE ELEMENT PROGRAMMING CODES

Main Program

```
clear all;
close all;
clc;

Nx=10; Ny=10; x0=0; xf=1; y0=0; yf=1;
Re=[1 10 50 100 200 500 1000]; % Reynolds numbers

for j=1:length(Re)
    tol=1e-6;
    maxit=1000;
    erru=1; errv=1;

    vnd=(Nx+1)*(Ny+1) - 8;
    mnd=(Nx+1)*Ny+Nx*(Ny+1) - 24;
    nd=vnd+mnd;
    neq=2*nd+vnd;

    u0=ones(nd,1);
    v0=zeros(nd,1);

    celem=connective(Nx,Ny);
    elem8=[celem(:,1:8)];
    elem4=[celem(:,9:12)];

    gdof=fungdof(Nx,Ny,x0,xf,y0,yf);
    nel=length(elem8(:,1));
    node=anode481(Nx,Ny,x0,xf,y0,yf);

    ne=length(celem(:,1));
    iter = 0;
    tic

    while (or(erru>tol, errv>tol) & iter<=maxit)
        iter = iter+1;

        A=assemblymatrix(Nx,Ny,x0,xf,y0, yf,u0,v0,Re(j));
        b=assemblyfluidvector(Nx,Ny,x0,xf,y0,yf,Re(j));

        dx=(xf-x0)/Nx; dy=(yf-y0)/Ny;
        dofa = [32,33,53,54,64,65,85,86,96,97,117,118];
```

```
    bDof=[1:22,dofa,128:137,141:150,152,153,157,158,160:169,173:182,dofa+16
    bDofb = [331,332,342,343];
    bdof = [bDof,310:321,bDofb,353:358,360:365,367:372,374:379,bDofb+58,411

    nb=10*(Nx+Ny)+80;
    bv=zeros(nb,1);
    bv(end:-1:end-21)=-100;

    upv=zeros(neq,1);
    nc=length(bdof);

    for i=1:nc
        id=bdof(i);
        upv(id)=bv(i);
    end

    freeNodes=setdiff (1:neq,bdof);
    b=b-A*upv;

    % Solve the system
    upv(freeNodes) = A(freeNodes,freeNodes) \ b(freeNodes);

    % Post-processing
    u=upv(1:nd);
    p=upv(nd+1:nd+vnd);
    v=upv(nd+vnd+1:neq);

    erru= max(abs(u-u0));
    errv= max(abs(v-v0));

    uvel=(u+u0)/2;
    vvel=(v+v0)/2;

    u0=uvel;
    v0=vvel;
end

fprintf('u-velocity v-velocity\n');
velocity=[u v];
disp(velocity);

fprintf('pressure\n');
disp(p);
```

```
    figure;
    node=anode481(Nx, Ny, x0, xf, y0, yf);
    x=node(:,1);
    y=node(:,2);

    L = sqrt(u.^2 + v.^2);
    quiver(x, y, u./L, v./L, 0.4);
    axis([0 1.1 0 1.1]);
    xlabel('x');
    title(['Velocity at Re = ', num2str(Re(j))]);
    ylabel('y');
    view(0,90);
end

toc;
```

Sub Program

```
function gdof=fungdof (Nx,Ny,x0,xf, y0, yf)
    vnd=(Nx+1)* (Ny+1) - 8;
    mnd=(Nx+1) *Ny+Nx* (Ny+1) - 24;
    NN=vnd+mnd;
    celem=connective(Nx,Ny);
    elem8=[celem(:,1:8)];
    pnode=zeros(Nx*Ny,4);
    for k=1:Nx
        pnode(k,1)=NN+k;
        pnode(k,2)=NN+k+1;
        pnode(k,3)=pnode(k,2)+Nx+1;
        pnode(k,4) =pnode(k,3)-1;
    end
      for k=Nx+1:Nx*Ny - 60
          pnode(k,1:4) =pnode(k-Nx, 1:4) +Nx+1;
      end

    pnode (41,1:2)=pnode(5,1:2)+44;
    pnode (41,3:4)=pnode(5,3:4)+ 40;
    pnode (42,1:2)=pnode(6,1:2)+ 44;
    pnode (42,3:4)=pnode(6,3:4)+40;
    pnode (43,1:2)=pnode(5,1:2)+ 51;
    pnode (43,3:4)=pnode(5,3:4)+47;
    pnode (44,1:2)=pnode(6,1:2)+51;
    pnode (44,3:4)=pnode(6,3:4)+47;
  for k=45:Nx*Ny - 16
      pnode (k, 1:4)=pnode (k-44,1:4)+ 58;
```

```
  end
     for e = 1:length(celem(:,1))
         dofCounter = 0;
        % u velocity DOFs
        for i= 1:8
            dofCounter = dofCounter + 1;
            gdof(e,dofCounter) = elem8(e,i);
        end
        % pressure DOFs
        for i = 1:4
            dofCounter = dofCounter + 1;
            gdof(e,dofCounter) = pnode(e,i);
         end
        % v velocity DOFs
        for i =1:8
            dofCounter = dofCounter + 1;
            gdof(e,dofCounter) =2*NN-mnd + elem8(e,i);
        end
    end
```

Sub Program

```
function Aelem=fluidelemmatrix (coord8, coord4,u0,v0,Re)
Aelem=zeros(20,20);
A1=zeros(8,8);
A2=zeros(8,4);
A3=zeros(8,8);
A4=zeros(4,8);
A5=zeros(4,4);
A6=zeros(4,8);
A7=zeros(8,8);
A8=zeros (8,4);
A9=zeros(8,8);
Nx=5; Ny=5; x0=0; xf=1; y0=0; yf=1;
celem=connective(Nx,Ny);
elem8=[celem(:,1:8)];
elem4=[celem(:,9:12)];
nel=length(elem8(:,1));
u0=ones(8,1);
v0=zeros(8,1);
xp=[-sqrt(0.6) 0 sqrt(0.6)];
wp=[5/9 8/9 5/9];
[g1,g2]=meshgrid(xp,xp);
gp=[g2(:) g1(:)];
```

```
[w1,w2]=meshgrid(wp,wp);
w=w2(:).* w1(:);
ngp=length(gp);
for k=1:ngp
    ksi=gp(k,1);
    eta=gp (k,2);
    N(1,k)=-0.25*(1-ksi)* (1-eta)* (1+ksi+eta);
    N(2,k)=0.5* (1-ksi.^2)* (1-eta);
    N(3,k)=-0.25*(1+ksi)*(1-eta) * (1-ksi+eta);
    N(4,k)=0.5*(1+ksi) *(1-eta.^2);
    N(5,k)= -0.25*(1+ksi)*(1+eta)*(1-ksi-eta);
    N(6,k)=0.5*(1-ksi.^2)* (1+eta);
    N(7,k)= -0.25* (1-ksi) * (1+eta)* (1+ksi-eta);
    N(8,k)= 0.5* (1-ksi)* (1-eta.^2);
    dN (1,1,k)=0.25*(1-eta)* (2*ksi+eta);
    dN (1,2,k)=-ksi*(1-eta);
    dN (1,3,k)=0.25*(eta-1)*(eta-2*ksi);
    dN(1,4,k)=0.5*(1-eta.^2);
    dN(1,5,k)=0.25*(eta+1)*(eta+2*ksi);
    dN(1,6,k)=-ksi*(eta+1);
    dN (1,7,k)=-0.25*(1+eta) *(eta-2*ksi);
    dN (1,8,k)=0.5*(-1+eta.^2);
    dN (2,1,k)=0.25*(1-ksi) * (2*eta+ksi);
    dN (2,2,k)=0.5* (-1+ksi.^2);
    dN (2,3,k)=0.25*(1+ksi) * (2*eta-ksi);
    dN(2,4,k)= -eta* (1+ksi);
    dN (2,5,k)=0.25*(1+ksi) * (ksi+2*eta);
    dN (2,6,k)=0.5*(1-ksi.^2);
    dN (2,7,k)=0.25*(1-ksi) * (2*eta-ksi);
    dN(2,8,k)= eta*(ksi-1);
    M(1,k)=0.25*(1-ksi) *(1-eta);
    M(2,k)=0.25* (1+ksi) * (1-eta);
    M(3,k)=0.25*(1+ksi) * (1+eta) ;
    M(4,k)=0.25*(1-ksi) * (1+eta) ;

    dM(1,1,k)=-0.25* (1-eta);
    dM(1,2,k)=0.25* (1-eta);
    dM (1,3,k)=0.25* (1+eta);
    dM(1,4,k)=-0.25* (1+eta);


    dM(2,1,k)=-0.25* (1-ksi);
    dM(2,2,k)=-0.25* (1+ksi);
    dM(2,3,k)= 0.25* (1+ksi);
    dM(2,4,k) =0.25*(1-ksi);
```

```
end


for k=1:ngp
    ubar = 0;
    vbar = 0;
    for i= 1:8
        ubar = ubar + N(i,k)*u0(i);
        vbar = vbar + N(i,k) *v0(i);
    end
    Jacob8 (:,:)= dN(:,:,k)*coord8(:,:);%jacobian at kth gp
    detJacob8=abs(det(Jacob8));
    gdN=Jacob8 (:,:)\dN(:,:,k);%2X8 change of variable in der. at kth gp
    gdN1=gdN(1,:);
    gdN2=gdN(2,:);
    for i=1:8
        for j=1:8
            A1(i,j)=A1(i,j) +w(k) * (N(i,k) *ubar*gdN1(j)...
                +N(i,k) *vbar*gdN2(j)...
                +(1/Re) * (gdN1 (i) *gdN1 (j) +gdN2 (i) *gdN2 (j))) *detJacob8;
        end
    end
    A9=A1;
    Jacob4 (:,:)=dM(:,:,k) *coord4(:,:);
    gdM=Jacob4 (:,:)\dM(:,:,k);

    gdM1=gdM(1,:);
    gdM2=gdM(2,:);
    detJacob4 = abs (det (Jacob4));
    for i=1:8
        for j=1:4
            A2 (i,j) =A2 (i,j) +w(k) *N(i,k) *gdM1(j)*detJacob4;
            A8 (i,j) =A8 (i,j) +w(k) *N(i,k) *gdM2(j)*detJacob4;
        end
    end
    Jacob4 (:,:)=dM(:,:,k) *coord4(:,:);
    gdM=Jacob4 (:,:)\dM(:,:,k);
    gdM1=gdM(1,:);
    gdM2=gdM(2,:);
    detJacob4=abs(det(Jacob4)) ;
    for i=1:4
        for j=1:8
            A4(i,j)= A4 (i,j) +w(k) *M(i,k) *gdN1 (j) *detJacob4;
            A6 (i,j) =A6 (i,j) +w(k)*M(i,k) *gdN2 (j) *detJacob4;
        end
```

```
    end
end
Aelem=[A1 A2 A3;A4 A5 A6;A7 A8 A9];
```

sub Program

```
function belem=localfluidvector(celem)
n=length (celem(1,:));
belem =zeros(2*n-4,1);
```

Sub Programl

```
function A=assemblymatrix(Nx,Ny,x0,xf,y0, yf,u0,v0,Re)
vnd=(Nx+1)* (Ny+1) - 8;
mnd=(Nx+1) *Ny+Nx* (Ny+1) - 24;
nd=vnd+mnd; %veliocity nodes
neq=2*nd+vnd; % total number of degree of freedom
celem=connective (Nx,Ny);
elem4=[celem(:,9:12)];
elem8=[celem(:,1:8)];
node=anode481(Nx,Ny,x0,xf,y0,yf);
gdof=fungdof(Nx,Ny, x0, xf,y0, yf);
n=2* length(celem(1,:))-length (elem4(1,:));
ne=length(celem(:,1)); %number of elements

A=zeros (neq,neq);
% invel=zeros (neq,i);
% invei (1:nd)=1:
%v0=invel (nd+vnd+i : neq) ;
for e=1:ne
    e
    Aelem=fluidelemmatrix(node(elem8(e,:),:),...
        node(elem4(e,:),:),u0(elem8(e,:),:),v0(elem8(e,:),:),Re);
    for i= 1:n
        for j = 1:n
            A(gdof(e,i),gdof(e,j))= A(gdof(e,i),gdof(e,j))+Aelem(i,j);
        end
     end
end
```

Sub Program

```
function b=assemblyfluidvector (Nx,Ny,x0,xf,y0, yf, Re)
vnd=(Nx+1)* (Ny+1) - 8;
mnd= (Nx+1) *Ny+Nx *(Ny+1) - 24;
nd=vnd+mnd;
```

```
neq=2*nd+vnd;
celem=connective(Nx,Ny);
elem4=[celem(:,9:12)];
elem8=[celem(:,1:8)];
node=node48(Nx,Ny,x0,xf, y0, yf);
n=2*length (celem(1,:))-4;
ne=length (celem(:,1));
gdof=fungdof(Nx,Ny,x0,xf,y0, yf );
b=zeros (neq,1);
for e = 1:ne
    belem=localfluidvector(celem) ;
    for i = 1:n
        b(gdof(e,i))= b(gdof (e,i))+belem(i);
    end
end
```

Mesh Generating Code

```
close all;
clear all:
clc
Nx=10;Ny=4;x0=0;xf=1;y0=0;yf=0.4;
dx=(xf-x0)/Nx;dy=(yf-y0)/Ny;
x1=x0:dx/2:xf;
x2=x0:dx:xf;
y1=y0:dy:yf;
y2=dy/2:dy:yf;
x3=x0:dx:xf;
y3=y0:dy:yf;
[X1,Y1] =meshgrid(x1,y1);
[X2,Y2] =meshgrid(x2,y2);
[X3,Y3] =meshgrid(x3,y3);
u1=0.004*ones (length (x1),length(y1));
u2=0.004*ones (length (x2),length(y2));
u3=0.004*ones(length (x3),length(y3));
Nx=10;Ny=4;x01=0;xf1=1.0;y01=0.6;yf1=1.0;
dx=(xf1-x01)/Nx;dy=(yf1-y01)/Ny;
x11=x01:dx/2:xf1;
x21=x01:dx:xf1;
y11=y01:dy:yf1;
y21=0.6+dy/2:dy:yf1;
x31=x01:dx:xf1;
y31=y01:dy:yf1;
[X11,Y11] =meshgrid(x11,y11);
[X21,Y21] =meshgrid(x21,y21);
[X31,Y31] =meshgrid(x31,y31);
```

```
u11=0.004*ones (length (x11),length(y11));
u21=0.004*ones (length (x21),length(y21));
u31=0.004*ones (length (x31),length(y31));
Nx=2;Ny=2;x02=0.4;xf2=0.6;y02=0.4;yf2=0.6;
dx=(xf2-x02)/Nx;dy=(yf2-y02)/Ny;
x12=x02:dx/2:xf2;
x22=x02:dx:xf2;
y12=y02:dy:yf2;
y22=0.4+dy/2:dy:yf2;
x32=x02:dx:xf2;
y32=y02:dy:yf2;
[X12,Y12] =meshgrid(x12,y12);
[X22,Y22] =meshgrid(x22,y22);
[X32,Y32] =meshgrid(x32,y32);
u12=0.004*ones (length (x12),length(y12));
u22=0.004*ones (length (x22),length(y22));
u32=0.004*ones (length (x32),length(y32));
figure
surf (x3, y3,u3', 'FaceColor','interp');
hold on
surf (x31, y31,u31', 'FaceColor','interp');
hold on
surf (x32, y32,u32', 'FaceColor','interp');
view(0,90)
P1=plot3 (X2,Y2,u2','.', 'markersize',25);
p2=plot3 (X1,Y1,u1','.', 'markersize',25);
hold on
P3=plot3 (X21,Y21,u21','.', 'markersize',25);
p4=plot3 (X11,Y11,u11','.', 'markersize',25);
p5=plot3 (X22,Y22,u22','.', 'markersize',25);
p6=plot3 (X32,Y32,u32','.', 'markersize',25);
axis([x0 xf1 y0 yf1 0 0.01]);
```

.....