MAE 766

Computational Fluid Dynamics

Spring 2019

Project 1

A FINITE ELEMENT PROGRAM FOR POTENTIAL FLOWS

Sujal Dave

MAE

200254193

1. Abstract

The projects aims at solving two cases for (i) internal flow in channel with bump and (ii) external flow around a cylinder, by writing a 2D incompressible potential solver using a linear finite element method on unstructured grids. The freestream velocity vector used the project is considered to be (1, 0). The mesh grid for each of the cases have been provided and the solver is written in FORTRAN. The post-processing graphs and contours have been plotted using TECPLOT360 software. The finite element solver is intended to augment the understanding of the finite element methods and to introduce the unstructured grid that can be used for numerically computing potential flows past arbitrary bodies. In order to enhance the knowledge about the convergence, a grid independence study is conducted to assess if the convergence can be achieved.

2. Introduction

A potential flow is enforced when the velocity field is described as the gradient of a scalar function, i.e. velocity potential. Since the curl of the gradient of a scalar is always zero, a potential flow is characterised by an irrotational velocity field. For a velocity potential ϕ, the velocity in a potential flow is,

$$\boldsymbol{v} = -\nabla\phi$$

In case of an incompressible flow the velocity potential satisfies the Laplace equation. Since we know that for the incompressible flow, the divergence of the velocity is zero.

$$\nabla . \boldsymbol{v} = 0$$
$$-\nabla^2\phi = 0$$

where, ∇^2 is the Laplace operator.

The finite element method used for solving the potential equation in the project is one of the most common numerical technique used in solving problems across engineering. The basic structure of solving an equation by the finite element method involves characterizing the equation by a variational formulation, then discretizing the domain, followed by enforcing solution algorithms and finally post-processing procedures. The discretization is conducted by creation of finite element meshes and defining basis functions on the reference element.

In this project, linear basis functions or Lagrangian functions will be used in order to solve the potential using the linear finite element method in the domain given. The governing equations for the problem in the project is as shown:

$$-\nabla^2\phi = 0 \qquad \text{in } \Omega$$

$$\frac{\partial\phi}{\partial n} = g = v_n = \boldsymbol{v} . \boldsymbol{n} = v_i n_i \qquad \text{on } \Gamma$$

Where, $\boldsymbol{\Omega}$ is a bounded open domain occupied by fluid, Γ its boundary and **n** is the outward normal unit vector to the Γ.

3. Description of methodology

3.1 Variational formulation and the weak form

The governing equation for the potential flow equation that is the Laplacian of the scalar potential is formulated in a weak form by using an arbitrary function v, that satisfies the boundary equations of the problem, later, on integrating both sides of the equation and applying divergence theorem and tensor identities, we arrive at the weak form of the equation which is given as,

$$\int_{\Omega} \nabla \phi . \nabla \mathrm{v} \, d\Omega = \int_{\Gamma} g v \, d\Gamma \qquad \forall v \in V$$

Where, V = {v: v is a continuous function on Ω}

3.2 Basis functions

Next, in order to represent the elements in the linear form, we introduce a set of basis functions $B_j \in V_h$, then we can write the finite approximation as, for N being the total points in the grid,

$$\sum_{i=1}^{N} \left(\int_{\Omega_h} \nabla B_i . \nabla B_j d\Omega \right) \phi_i = \int_{\Gamma} g \, B_j \, d\Gamma \ , \mathrm{j} = 1,2,3, \dots \dots , \mathrm{N}$$

In the matrix form, $A\phi = B$

A = a_{ij} is a matrix of dimension NxN

A (i,j) = a_{ij} = $\int_{\Omega_h} \nabla B_i . \nabla B_j d\Omega$: stiffness matrix.

B (j) = b_j = $\int_{\Gamma} g \, B_j \, d\Gamma$: load vector

Since this is a linear method, introduce, Barycentric coordinates λ_i,

$$\lambda_i = \frac{a_i x + b_i y + c_i}{D} \ , i = 1,2,3$$

Where,

$$a_i = y_j - y_k$$

$$b_i = -(x_j - x_k)$$

$$c_i = x_j y_k - x_k y_j$$

$$D = 2\Delta_{123} = c_1 + c_2 + c_3$$

Δ_{123} is the area of the triangle formed by local points 1,2,3.

Also, noting here, that,

$$\frac{d\lambda_i}{dx} = \frac{a_i}{D} \quad and \quad \frac{d\lambda_i}{dy} = \frac{b_i}{D}$$

For the method used in the project,

$$B_i = \lambda_i$$

3.3 Local and Global matrices

The stiffness and the load matrices as described in the earlier section are written out for each element (a_{ij} and b_j) and then a global matrix (A and B) are assembled. The basis functions over each element are evaluated and assembled into the local matrices which are then added according to the global indices into the global matrices. The local (element) matrices are,

$a_{ij}^{(e)}$: elemental stiffness matrices for each element 'e'

$b_i^{(f)}$: elemental load vector for each boundary face 'f'

Upon calculating these for every element in the domain, we assemble the local matrices into global matrices,

$$A\,(I,J) = \sum_e a_{ij}^{(e)}$$

$$B\,(J) = \sum_f b_i^{(f)}$$

Where, the capital letters correspond to the global variables and the small letters correspond to the local variables. Hence, now we obtain the equation,

$$A\phi = B$$

3.4 Solving the linear system $A\phi = B$

To solve the linear system obtained, a solver is written using the Gauss-Seidel method, the algorithm for which is as presented-

Input: $A, B, \phi - old, \phi - new, TOL\ (tolerance), N\ (maximum\ number\ of\ iterations), n\ (nodes)$
Step 1: Set $k = 1$
Step 2: while ($k \leq N$), do steps 3 to 6
Step 3 for $i = 1,2,3 \ldots .n$
$$(\phi - new)_i = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1}\left(A_{ij} * (\phi - new)_j\right) - \sum_{j=i+1}^{n}\left(A_{ij} * (\phi - old)_j\right) + B_i\right]$$
Step 4 If $\|(\phi - new) - (\phi - old)\| < TOL, then\ STOP$
Step 5 $k = k + 1$
Step 6 Set $\phi - old = \phi - new$
Step 7 Output $\phi - new$

3.5 Post-processing

On solving the linear system, we obtain the potential function ϕ at each point in the grid. As a first step we need to calculate the velocity potential at each element by the following equality,

$$\phi_e = \sum_{i=1}^{3} \phi_i B_i(x,y) = \sum_{i=1}^{3} \phi_i \lambda_i(x,y)$$

Then,

$$V_x = \frac{\partial \phi}{\partial x} = \sum_{i=1}^{3} \phi_i \frac{\partial \lambda_i}{\partial x}$$

$$V_y = \frac{\partial \phi}{\partial y} = \sum_{i=1}^{3} \phi_i \frac{\partial \lambda_i}{\partial y}$$

We know, that the velocity is piecewise constant on each element. Hence, now calculating the velocity on each point in the domain we use the following formula,

$$V_p = \frac{\sum_{e \in p} V_e w_e}{\sum_{e \in p} w_e}$$

Where, w_e is the area weight function of the element 'e'

4. Results

Putting together the velocities that we have found out for each node, we find out the x- and y-components and from that the total velocity for the test cases. Tecplot360 is used to read the output file with the coordinates, velocity potential and the total velocity magnitude which is used to plot the contours of the quantities. The following figures present the mesh, velocity potential, total velocity magnitude and the velocity vectors for the various test cases that were solved using the potential solver coded.

The velocity distributions along the top and bottom surfaces of the bodies which are tested are also presented in the graphs. The velocities at the surface were extracted by writing a code snippet wherein the points on boundary face with a certain flag were identified and the coordinates and velocity at the point were fed into a new file.

Subsequently, by applying the vector L^2 Norm, the error function was found out. The exact solution for the potential equation in polar coordinates is given by,

$$\phi(r,\theta) = Ur\left(1 + \frac{R^2}{r^2}\right)cos\theta$$

Transforming the polar coordinates to the Cartesian coordinates, we can rewrite the equation as,

$$\phi(x,y) = Ux\left(1 + \frac{R^2}{x^2 + y^2}\right)$$

Where, U is the x-component of the velocity and R is the radius of the cylinder.

The following is the short code written for the calculation of the error in each grid refinement

```
do i=1,npoin

r=(coord(1,i)*coord(1,i))+(coord(2,i)*coord(2,i))

exact(i)=vx*coord(1,i)*(1+(0.25/(r)))

diff=exact(i)-phinew(i)

eror=eror+(diff*diff)

end do

eror=sqrt(eror)/npoin
```

Case 1: Internal flow in a channel with bump

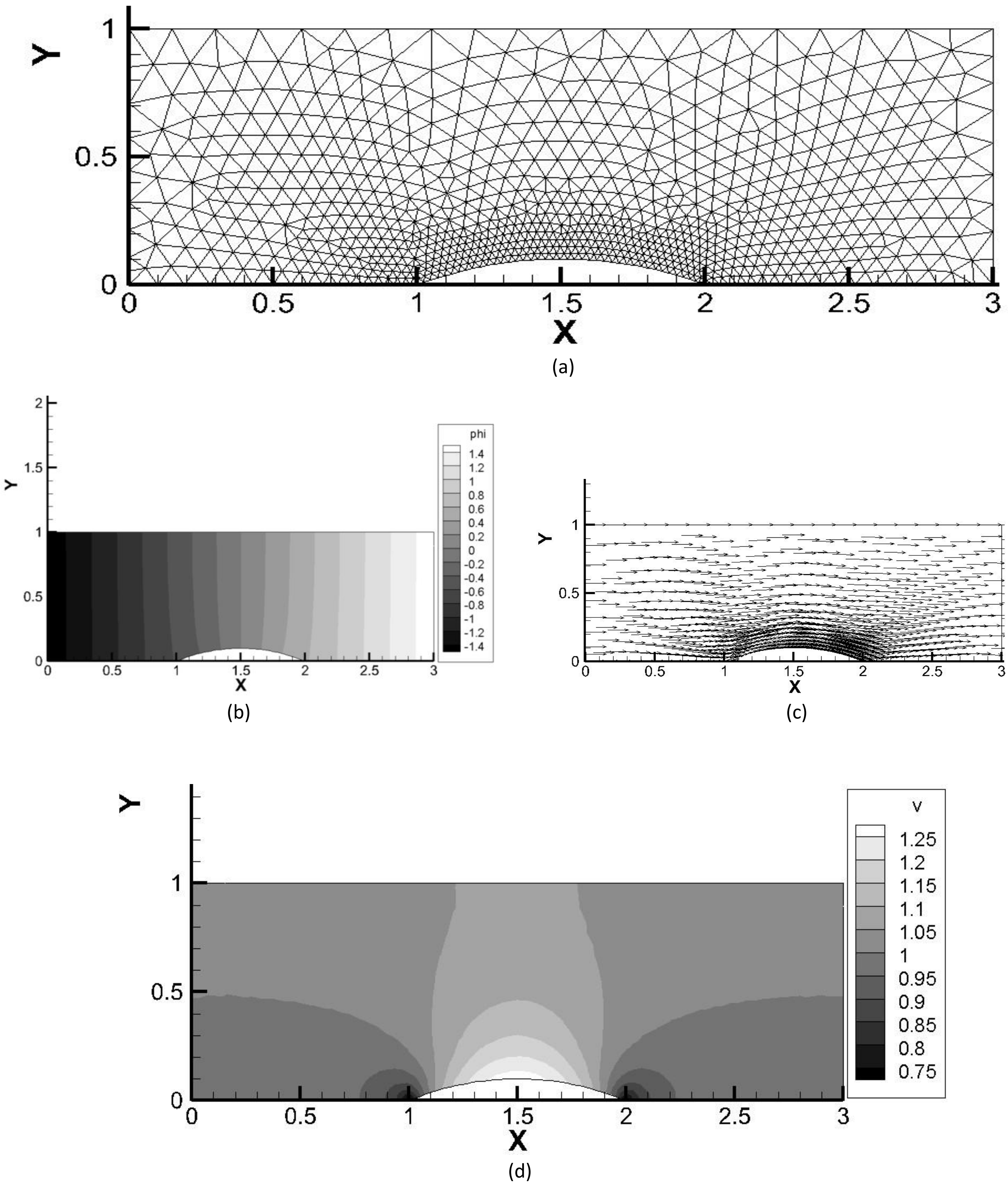


Figure 1: (a) Mesh (b) Velocity potential contours (c) Velocity vectors and (d) Velocity contours

Case 2: External flow past a cylinder

Coarse Mesh-

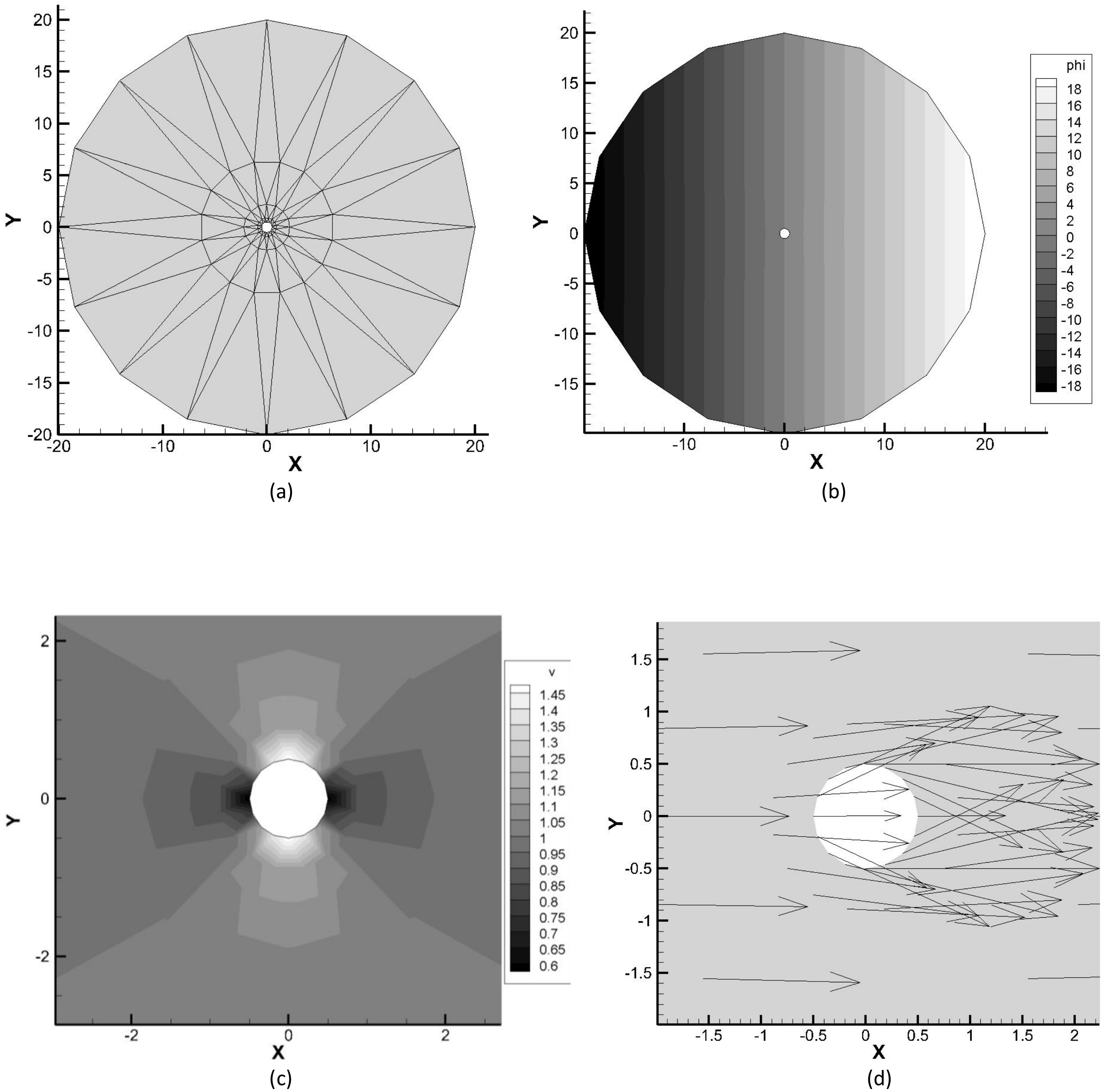


Figure 2: (a) Mesh (b) Velocity potential contours (c) Velocity contours and (d) Velocity vectors

Medium Mesh-

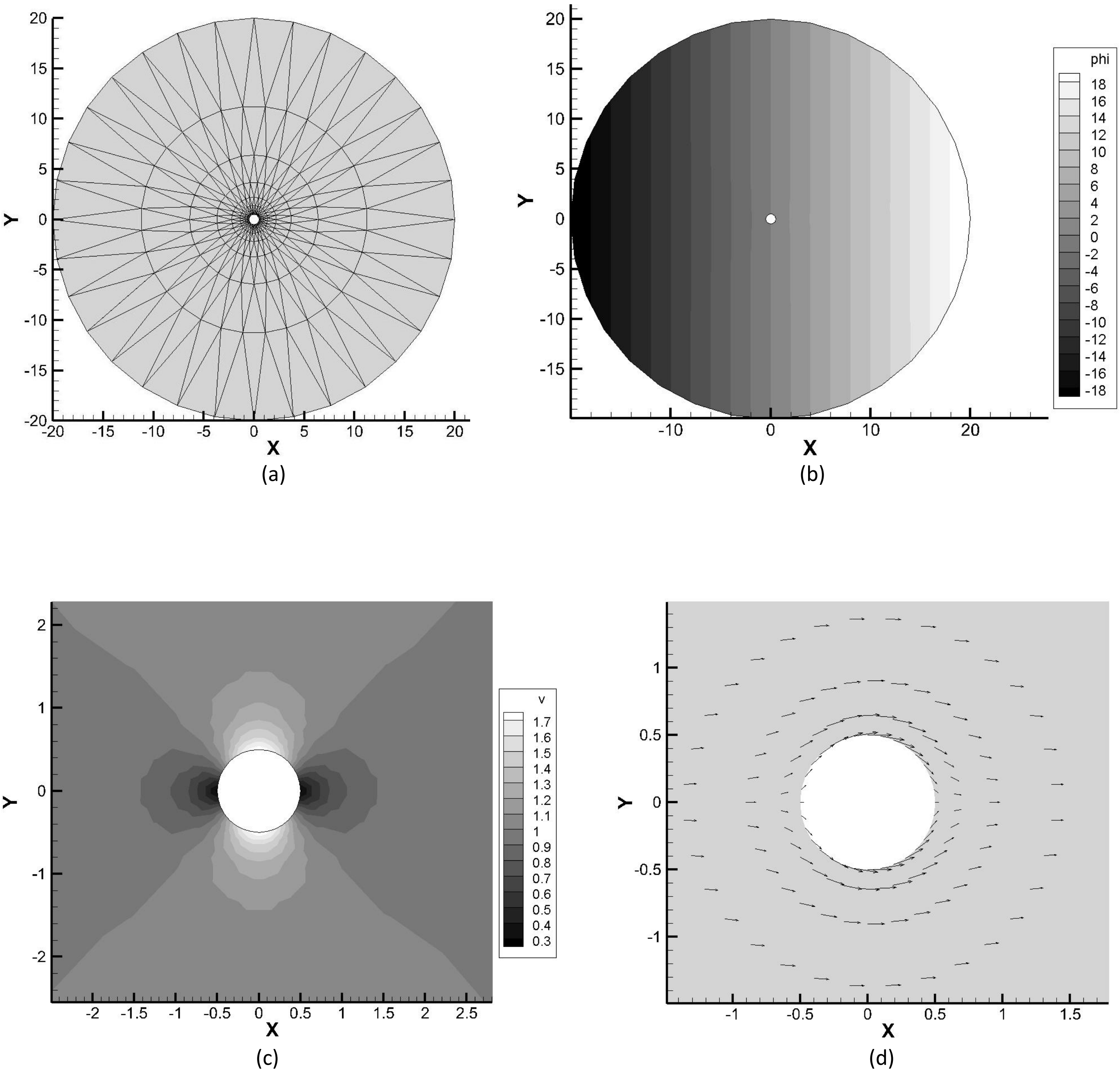


Figure 3: (a) Mesh (b) Velocity potential contours (c) Velocity contours and (d) Velocity vectors

Fine Mesh-

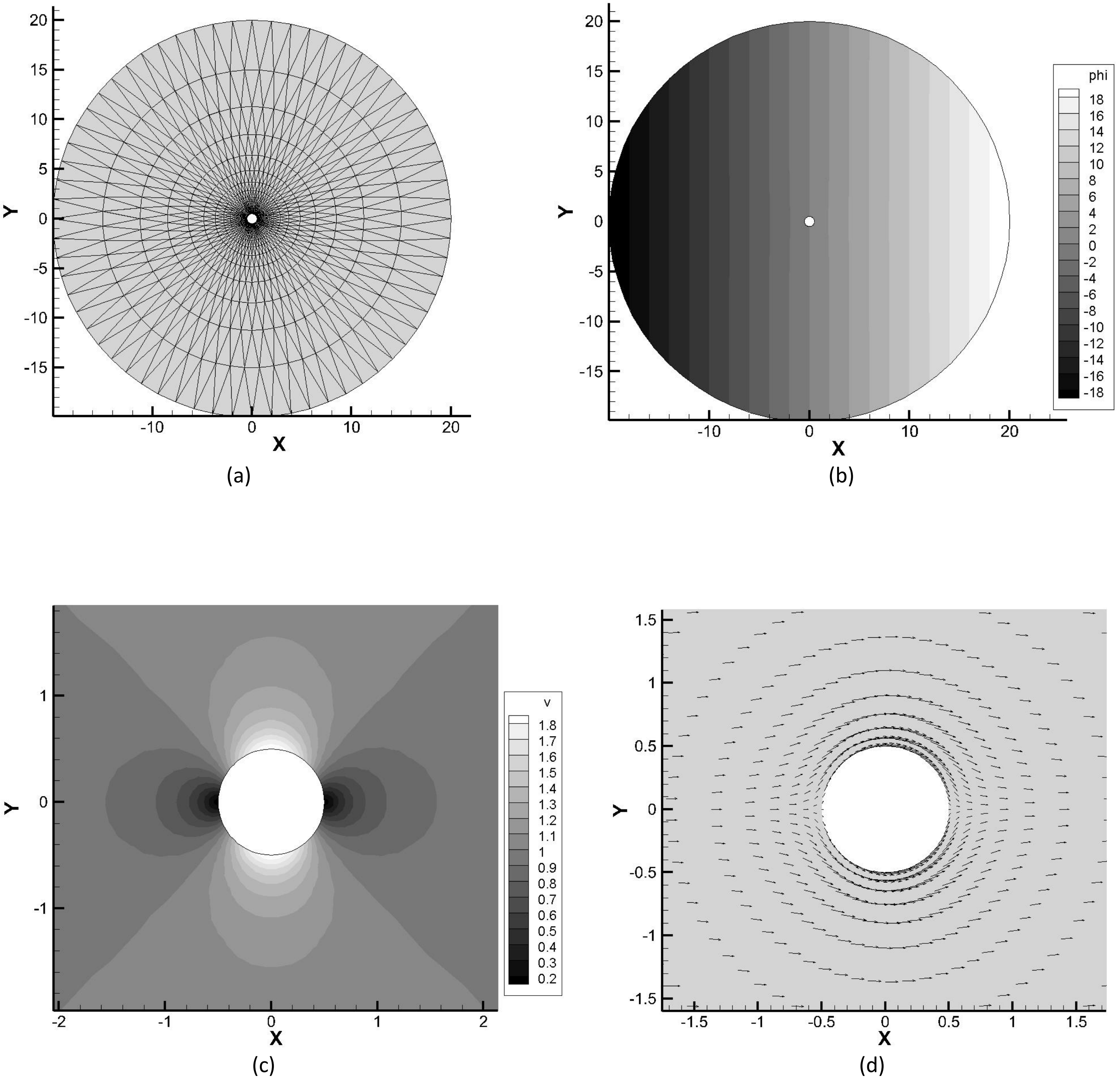


Figure 4: (a) Mesh (b) Velocity potential contours (c) Velocity contours and (d) Velocity vectors

Very Fine Mesh-

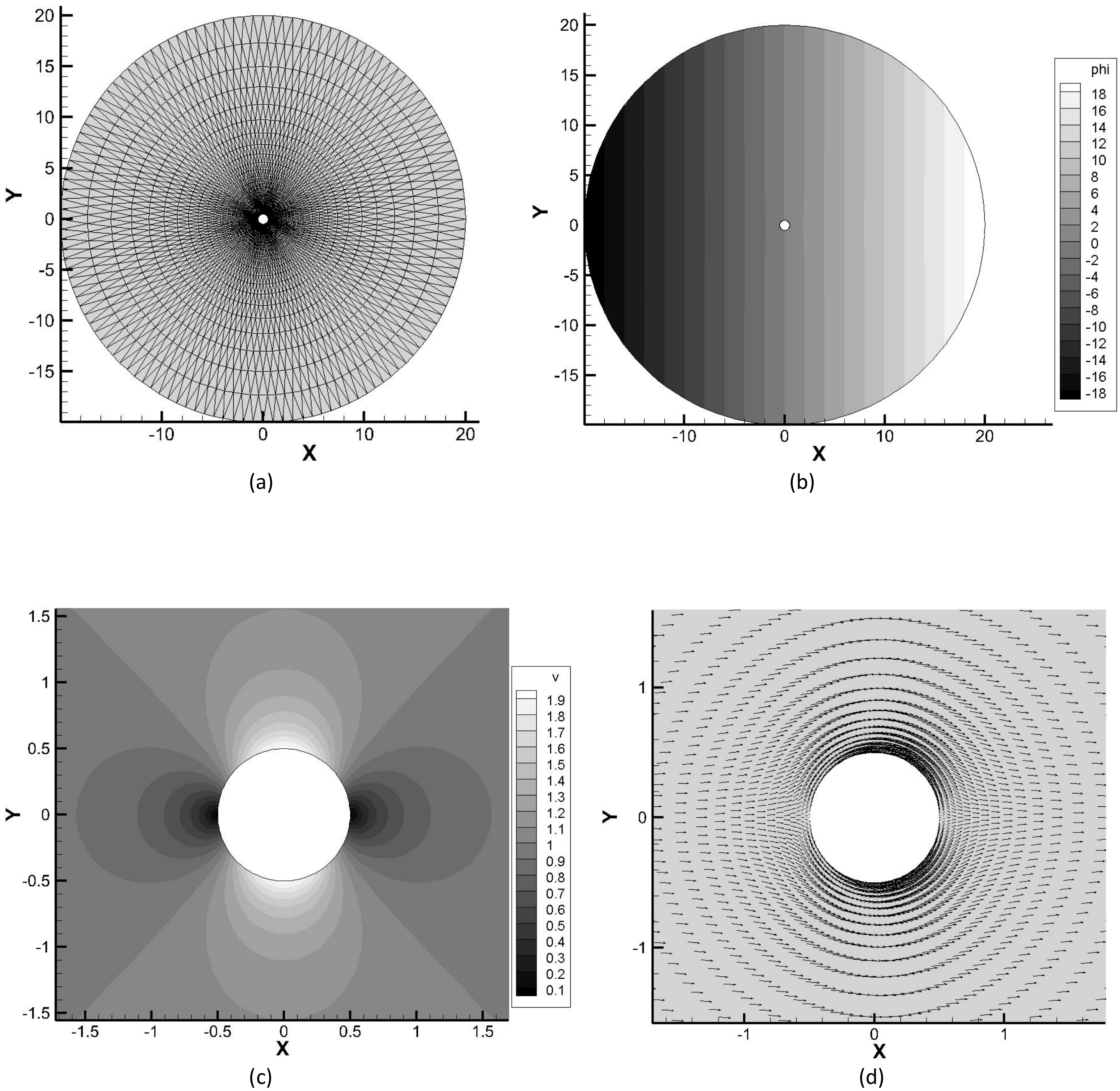
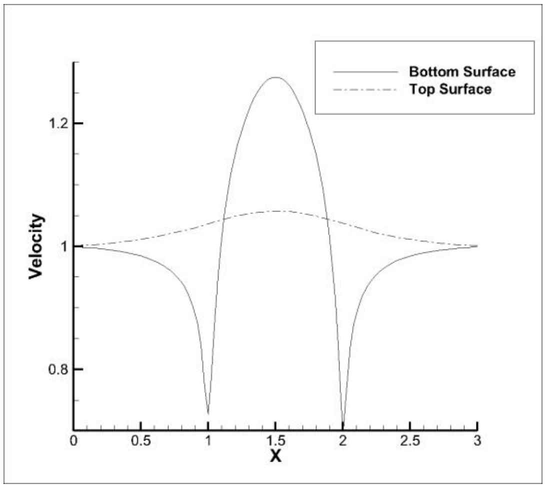
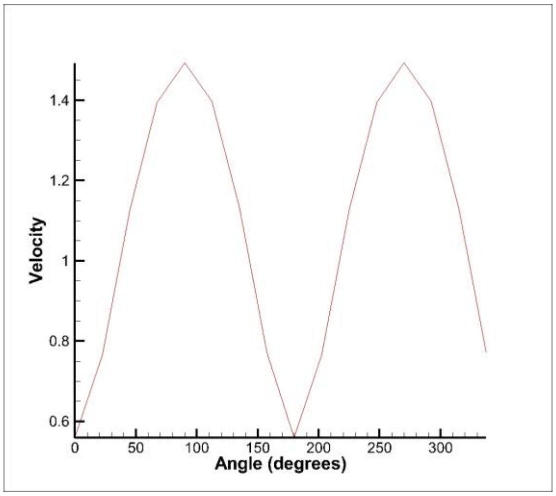


Figure 5: (a) Mesh (b) Velocity potential contours (c) Velocity contours and (d) Velocity vectors

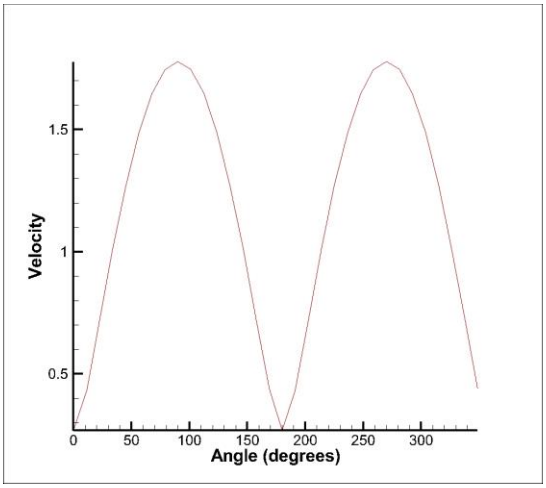
Velocity distribution over the surface



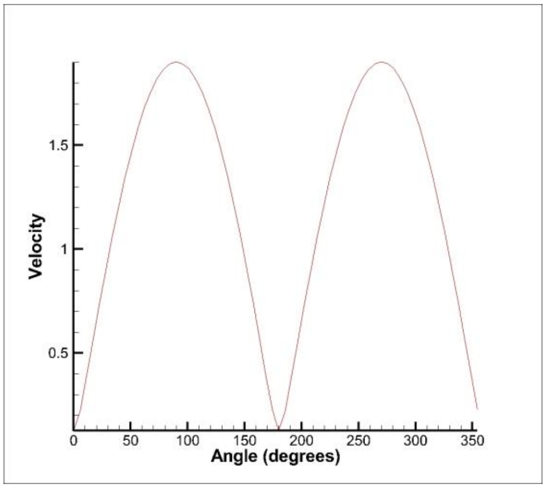
(a)



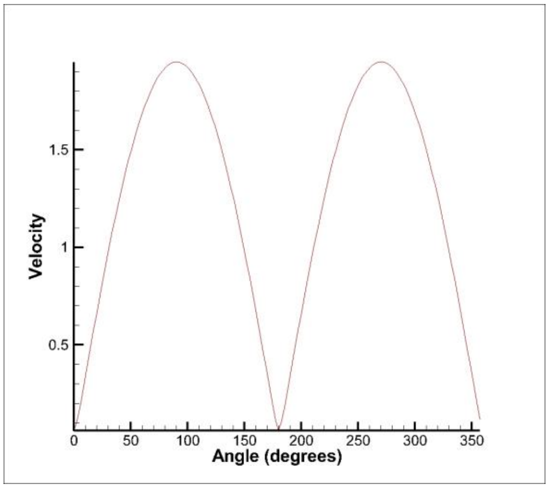
(b)



(c)



(d)



(e)

Figure 6: Velocity distributions on the surface of (a) channel with bnump (b) coarse mesh (c) medium mesh (d) fine mesh (e) very fine mesh

Mesh Type	Calculated Error
Coarse	5.2234875×10^{-3}
Medium	8.0532813×10^{-4}
Fine	1.5865992×10^{-4}
Very Fine	5.5334334×10^{-5}

Table 1: Error associated with the mesh provided

5. Conclusions

The tests on the various problem cases of internal and external flows using the potential linear finite element method solver were carried out successfully. The Gauss-Seidel converged according to the tolerance and the contours of velocity potential and total velocity were verified with previous results and are in good accordance. Hence, the aim of the project has been achieved which was to augment the understanding of finite element methods and an introduction to unstructured grid techniques. As evident from the data, as the mesh refinement is increasing, the error associated with the numerical analysis is also reducing drastically. Hence, we can see that the formal of convergence can be achieved.