Mini Project - Topic 02

Topic: *Velocity Profile of airflow in underground drivages*

ABSTRACT:

- This report presents a **comprehensive analysis of airflow behaviour within underground mine cross-cut regions, focusing on the utilisation of brattice sails to direct airflow.** Brattice sails are vital ventilation control devices used in underground mining to ensure efficient airflow distribution, thereby enhancing safety and working conditions for miners.
- The study employs **Computational Fluid Dynamics (CFD) modelling techniques** to investigate the impact of brattice sail length on fluid flow behaviour in cross-cut regions. Through iterative simulations based on the **Navier-Stokes equations and Finite Difference Method (FDM)**, the research evaluates various brattice sail lengths to understand ventilation air behaviour and determine optimal sizes for efficient contaminant removal from unventilated mine areas.
- Key aspects covered include the algorithmic framework, which employs **iterative solvers to solve the Navier-Stokes equations**, considering the influence of brattice sails on airflow dynamics. The study also discusses the **importance of brattice design and placement for optimising ventilation efficiency** and contaminant control within underground mines.
- Results highlight the **dependency of streamlines on the magnitude of the main airstream velocity**, with significant enhancements observed in airflow patterns with brattice usage. Moreover, the influence of brattice length on y-velocity in cross-cut regions underscores the **importance of brattice design parameters in improving ventilation mechanisms**.
- Overall, this study provides valuable insights into predicting airflow behaviours in mine sections employing brattices and underscores the importance of optimising brattice design and placement for enhanced ventilation strategies and safety standards in underground mining operations. **Future research directions involve expanding the model to explore additional brattice configurations and locations within mine workings**, contributing to advancements in underground mine ventilation systems and safety protocols.

INTRODUCTION:

- Efficient airflow management is crucial for ensuring the safety and well-being of miners in underground mining environments. Brattice sails, lightweight ventilation control devices, play a pivotal role in directing airflow to specific areas within mines, improving ventilation where it may be lacking. This project utilises Computational Fluid Dynamics (CFD) modelling techniques to analyse airflow behaviour in underground mine cross-cut regions, focusing on the influence of brattice sails.

- By employing CFD simulations, the project aims to gain insights into the impact of brattice sail length on airflow dynamics and optimise ventilation strategies for efficient contaminant removal. Using Python programming language and libraries such as NumPy and Matplotlib, the project simulates fluid flow in a 2D domain, applying the Finite Difference Method (FDM) to discretize the Navier-Stokes equations governing fluid flow.

- Through the analysis of various brattice sail lengths, the project seeks to provide valuable insights into ventilation airflow dynamics, guiding the design and implementation of brattice sails for improved air circulation and contaminant control within underground mine environments. Ultimately, the project endeavours to contribute to the enhancement of ventilation strategies and safety standards in underground mining operations, ensuring the well-being of miners and the efficiency of mining operations.

LITERATURE REVIEW:

Navier-Stokes Equations: The Navier-Stokes equations govern the motion of fluid substances and are fundamental in fluid dynamics. They provide a mathematical description of how velocity, pressure, temperature, and density vary in space and time within a fluid. The Navier-Stokes equations are a set of partial differential equations that describe the motion of viscous fluid substances. They are derived from the principles of conservation of mass and conservation of momentum, and they govern the evolution of fluid velocity and pressure in space and time.

Conservation of Mass (Continuity Equation):

$$\nabla \cdot \mathbf{v} = 0$$

This equation represents the conservation of mass, stating that the divergence of the velocity field ($\mathbf{v}$) is equal to zero, indicating that the flow is incompressible.

Conservation of Momentum (Momentum Equations):

For the x-component:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{1}{\rho}\nabla p + \nu \nabla^2 u + \mathbf{f}_x$$

For the y-component:

$$\frac{\partial v}{\partial t} + (u \cdot \nabla)v = -\frac{1}{\rho}\nabla p + \nu \nabla^2 v + \mathbf{f}_y$$

For the z-component:

$$\frac{\partial w}{\partial t} + (u \cdot \nabla)w = -\frac{1}{\rho}\nabla p + \nu \nabla^2 w + \mathbf{f}_z$$

where:

- $\mathbf{v} = (u, v, w)$ is the velocity vector,
- p is the pressure,
- ρ is the density of the fluid,
- ν is the kinematic viscosity,
- $\mathbf{f}_x$, $\mathbf{f}_y$, and $\mathbf{f}_z$ represent external body forces in the x, y, and z directions, respectively.

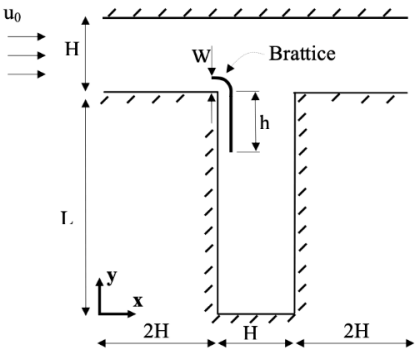
These equations describe the time evolution of velocity and pressure fields in a fluid, accounting for convection, pressure gradients, viscous forces, and external forces. They are nonlinear and coupled, making their analytical solution difficult in most cases, necessitating the use of numerical methods for practical simulations.

Finite Difference Method (FDM): FDM is a numerical technique used to solve partial differential equations (PDEs) by approximating derivatives using finite differences. It is widely used in CFD due to its simplicity and effectiveness in discretizing the domain into a grid and solving the governing equations iteratively. FDM is particularly suitable for simple geometries and regular grids, making it a good choice for initial simulations and educational purposes.

Iterative Solver: The algorithm employs an iterative solver to numerically solve the Navier-Stokes equations over a series of time steps. By iteratively updating the velocity and pressure fields, the algorithm converges to a solution that satisfies the governing equations and boundary conditions.

VISUALISATION:

Schematic Diagram of Channel:



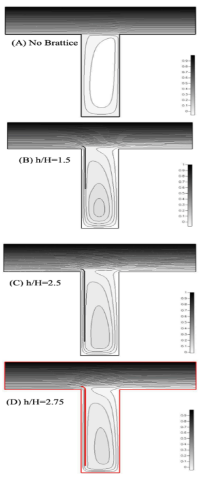


Figure 4. Stream Function patterns (ψ/ψ_{max}) in the cross-cut region at Re=3.2×10^5

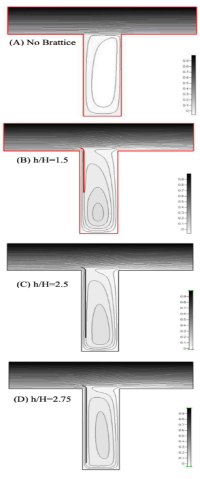


Figure 5. Stream Function patterns (ψ/ψ_{max}) in the cross-cut region at Re=1.6×10^5

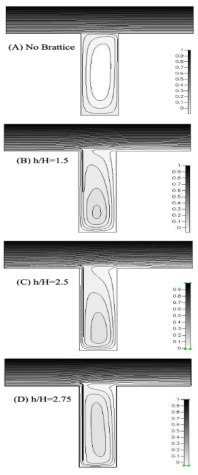


Figure 6. Stream Function patterns (ψ/ψ_{max}) in the cross-cut region at Re=0.3×10^5

PROPOSED METHODOLOGY:

The algorithm used in the provided code is a basic iterative solver for the Navier-Stokes equations using the Finite Difference Method (FDM). Below is an outline of the algorithmic steps:

Initialization:

- Define the parameters of the simulation such as grid size, domain size, viscosity, density, time step size, and number of time steps.
- Initialise velocity (u, v), pressure (p) fields, and boundary conditions.

Main Loop:

- Iterate over time steps (nt):

Momentum Equations:

- Solve the momentum equations for each grid point within the domain.
- Calculate the new velocity components (u, v) using the previous time step's velocity values (un, vn).
- Apply the Finite Difference Method to discretize the convective, diffusive, and pressure gradient terms in the Navier-Stokes equations.

Brattice Sail Influence:

- Check if the current grid point is within the influence of the brattice sail.
- Modify the velocity components within the region influenced by the brattice sail if necessary.

Pressure Equation:

- Solve the pressure equation to ensure divergence-free flow.
- Update the pressure field (p) based on neighbouring pressure values.

Boundary Conditions for Pressure:

- Apply boundary conditions to the pressure field to maintain consistency with the specified boundary conditions.

Visualisation:

- Plot the results, including contour plots of velocity magnitude and streamlines, using Matplotlib.

Repeat:

- Repeat the process for the specified number of time steps.

The algorithm iteratively solves the **Navier-Stokes equations** for fluid flow in a 2D domain while considering the influence of the brattice sail within the specified region. This algorithm is implemented using nested loops to traversE.

Explanation:

- **Import Libraries**: The code starts by importing necessary libraries, including `numpy` for numerical computations and `matplotlib.pyplot` for plotting.

- **Define Parameters**: Key parameters such as grid size (`nx`, `ny`), domain size (`Lx`, `Ly`), grid spacing (`dx`, `dy`), viscosity (`nu`), density (`rho`), time step size (`dt`), and number of time steps (`nt`) are defined.

- **Initialize Variables**: Arrays for velocity in x-direction (`u`), velocity in y-direction (`v`), and pressure (`p`) are initialised with zeros. These arrays represent the flow field variables.

- **Define Boundary Conditions**: Inlet velocity (`u[:, 0]`) is set to 1.0, representing the fluid entering the domain. No-slip boundary conditions are applied at the outlet (`u[:, -1] = 0.0`) and all other boundaries (`v[0, :]`, `v[-1, :]`). These conditions specify that the fluid does not slip along the boundaries.

- **Main Loop - Time Integration**: The main loop iterates over time steps (`nt`). Within each time step, the velocity and pressure fields are updated based on the Navier-Stokes equations.

- **Solve Momentum Equations**: Within each time step, the momentum equations (in x and y directions) are solved using finite difference approximations. These equations represent the conservation of momentum and include convection, diffusion, and external forces (gravity).

- **Solve Pressure Equation:** The pressure equation, derived from the continuity equation, is solved to ensure divergence-free flow. The pressure field is updated based on neighbouring pressure values.

- **Apply Boundary Conditions for Pressure:** Pressure boundary conditions are applied to ensure the pressure gradient is zero at inlet, outlet, bottom, and top boundaries.

- **Plot Results**: After the simulation completes, the velocity magnitude contour plot and streamlines are generated using `matplotlib`.

MATERIALS AND TOOLS USED:

The software, tools, and algorithms used:

Python: Python is a high-level programming language commonly used for scientific computing and numerical simulations. It offers libraries such as NumPy and Matplotlib, which are used extensively in this code.

NumPy: NumPy is a Python library for numerical computing. It provides support for multidimensional arrays and matrices, along with mathematical functions to operate on these arrays efficiently. In this code, NumPy arrays represent the velocity field (`u`, `v`), and pressure field (`p`), and perform array operations.

Matplotlib: Matplotlib is a plotting library for Python. It provides a MATLAB-like interface for creating static, interactive, and animated visualisations in Python. In this code, Matplotlib is used to visualise the results of the simulation, including contour plots of velocity magnitude and streamlines.

Finite Difference Method: The Finite Difference Method (FDM) is a numerical technique used to solve partial differential equations (PDEs) by discretizing the domain into a grid and approximating derivatives using finite differences. In this code, FDM is used to discretize the Navier-Stokes equations and solve them iteratively over time.

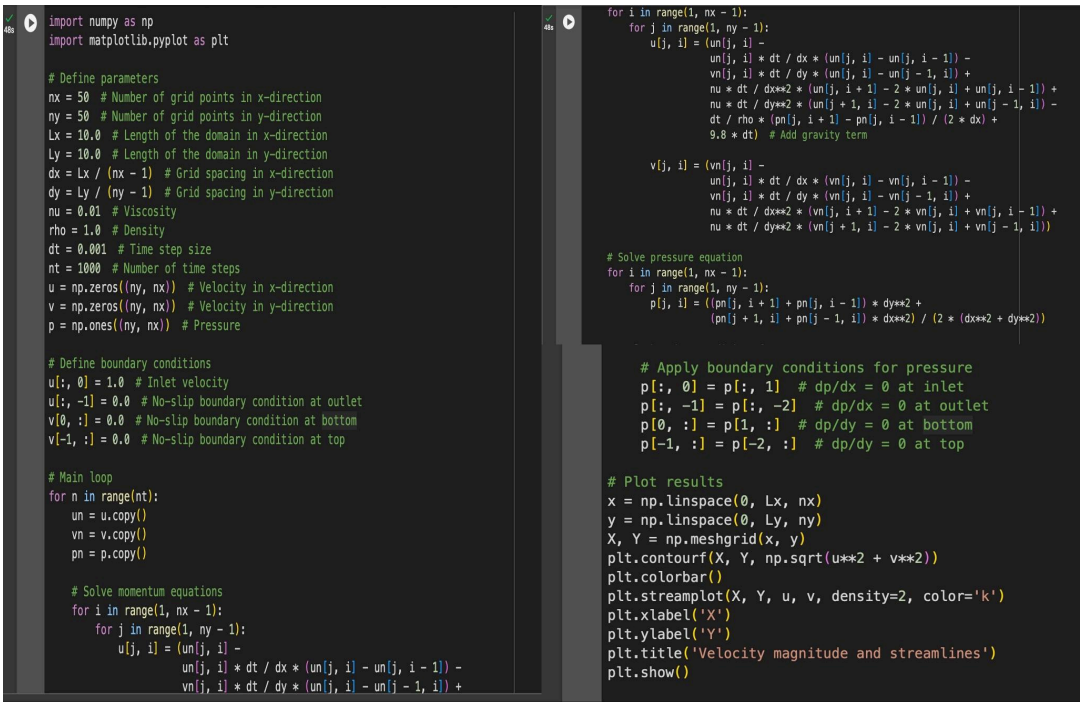
Navier-Stokes Equations: The Navier-Stokes equations describe the motion of fluid substances. They are a set of partial differential equations that govern the conservation of momentum and mass for fluid flow. In this code, the Navier-Stokes equations are discretized and solved numerically to simulate fluid flow in the domain.

Boundary Conditions: Boundary conditions are used to specify the behaviour of the flow at the domain boundaries. In this code, no-slip boundary conditions are applied at the top, bottom, and outlet boundaries, while an inlet velocity boundary condition is specified at the inlet.

Overall, the code utilises Python and NumPy, and Matplotlib libraries to implement a simple CFD simulation using the Finite Difference Method to solve the Navier-Stokes equations for fluid flow in a 2D domain.

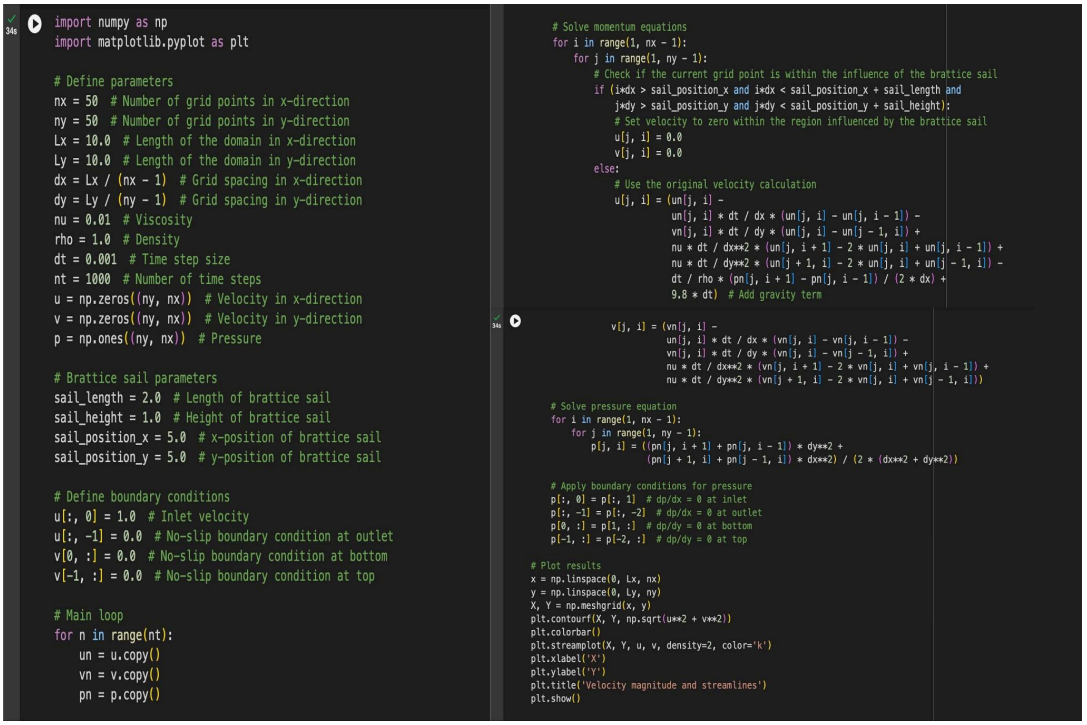
CODING:

Without using Brattice Sail as a parameter:



```python
import numpy as np
import matplotlib.pyplot as plt

# Define parameters
nx = 50  # Number of grid points in x-direction
ny = 50  # Number of grid points in y-direction
Lx = 10.0  # Length of the domain in x-direction
Ly = 10.0  # Length of the domain in y-direction
dx = Lx / (nx - 1)  # Grid spacing in x-direction
dy = Ly / (ny - 1)  # Grid spacing in y-direction
nu = 0.01  # Viscosity
rho = 1.0  # Density
dt = 0.001  # Time step size
nt = 1000  # Number of time steps
u = np.zeros((ny, nx))  # Velocity in x-direction
v = np.zeros((ny, nx))  # Velocity in y-direction
p = np.ones((ny, nx))  # Pressure

# Define boundary conditions
u[:, 0] = 1.0  # Inlet velocity
u[:, -1] = 0.0  # No-slip boundary condition at outlet
v[0, :] = 0.0  # No-slip boundary condition at bottom
v[-1, :] = 0.0  # No-slip boundary condition at top

# Main loop
for n in range(nt):
    un = u.copy()
    vn = v.copy()
    pn = p.copy()

    # Solve momentum equations
    for i in range(1, nx - 1):
        for j in range(1, ny - 1):
            u[j, i] = (un[j, i] -
                       un[j, i] * dt / dx * (un[j, i] - un[j, i - 1]) -
                       vn[j, i] * dt / dy * (un[j, i] - un[j - 1, i]) +
                       nu * dt / dx**2 * (un[j, i + 1] - 2 * un[j, i] + un[j, i - 1]) +
                       nu * dt / dy**2 * (un[j + 1, i] - 2 * un[j, i] + un[j - 1, i]) -
                       dt / rho * (pn[j, i + 1] - pn[j, i - 1]) / (2 * dx) +
                       9.8 * dt)  # Add gravity term

            v[j, i] = (vn[j, i] -
                       un[j, i] * dt / dx * (vn[j, i] - vn[j, i - 1]) -
                       vn[j, i] * dt / dy * (vn[j, i] - vn[j - 1, i]) +
                       nu * dt / dx**2 * (vn[j, i + 1] - 2 * vn[j, i] + vn[j, i - 1]) +
                       nu * dt / dy**2 * (vn[j + 1, i] - 2 * vn[j, i] + vn[j - 1, i]))

    # Solve pressure equation
    for i in range(1, nx - 1):
        for j in range(1, ny - 1):
            p[j, i] = ((pn[j, i + 1] + pn[j, i - 1]) * dy**2 +
                       (pn[j + 1, i] + pn[j - 1, i]) * dx**2) / (2 * (dx**2 + dy**2))

    # Apply boundary conditions for pressure
    p[:, 0] = p[:, 1]  # dp/dx = 0 at inlet
    p[:, -1] = p[:, -2]  # dp/dx = 0 at outlet
    p[0, :] = p[1, :]  # dp/dy = 0 at bottom
    p[-1, :] = p[-2, :]  # dp/dy = 0 at top

# Plot results
x = np.linspace(0, Lx, nx)
y = np.linspace(0, Ly, ny)
X, Y = np.meshgrid(x, y)
plt.contourf(X, Y, np.sqrt(u**2 + v**2))
plt.colorbar()
plt.streamplot(X, Y, u, v, density=2, color='k')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Velocity magnitude and streamlines')
plt.show()
```

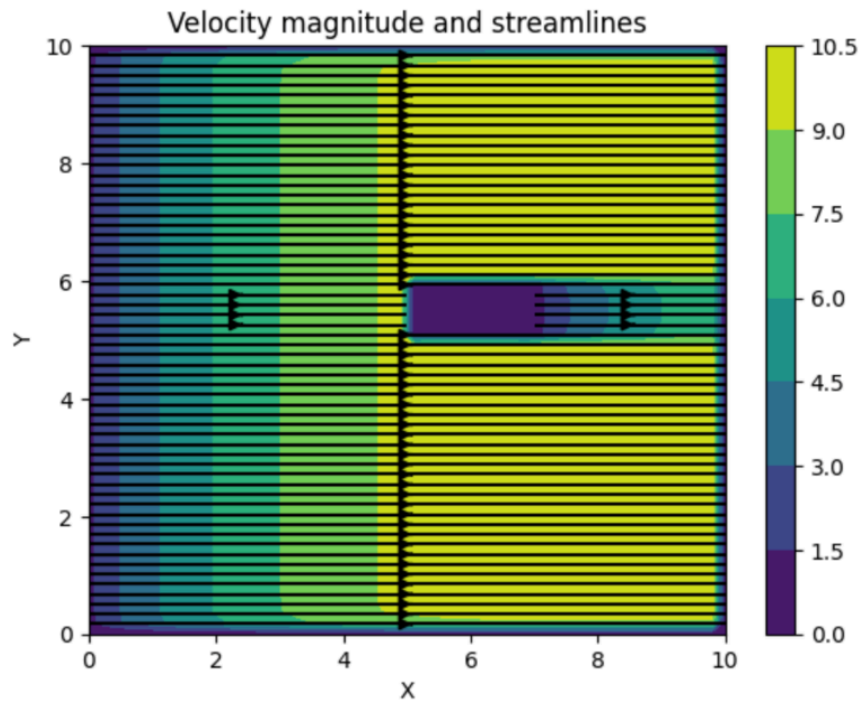
Using Brattice sail as a parameter:

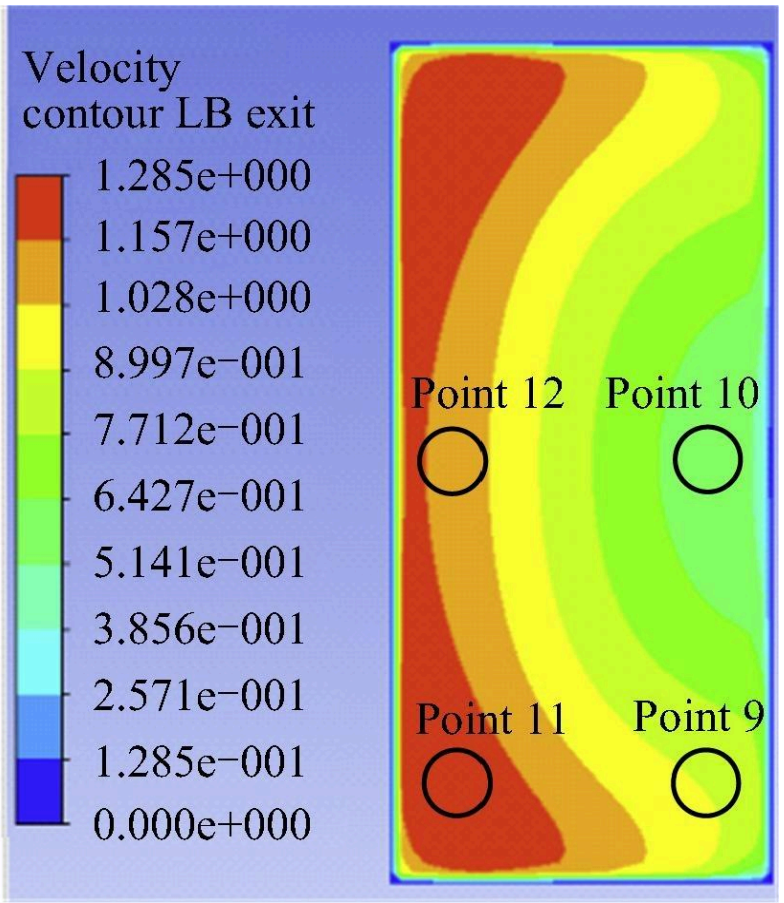


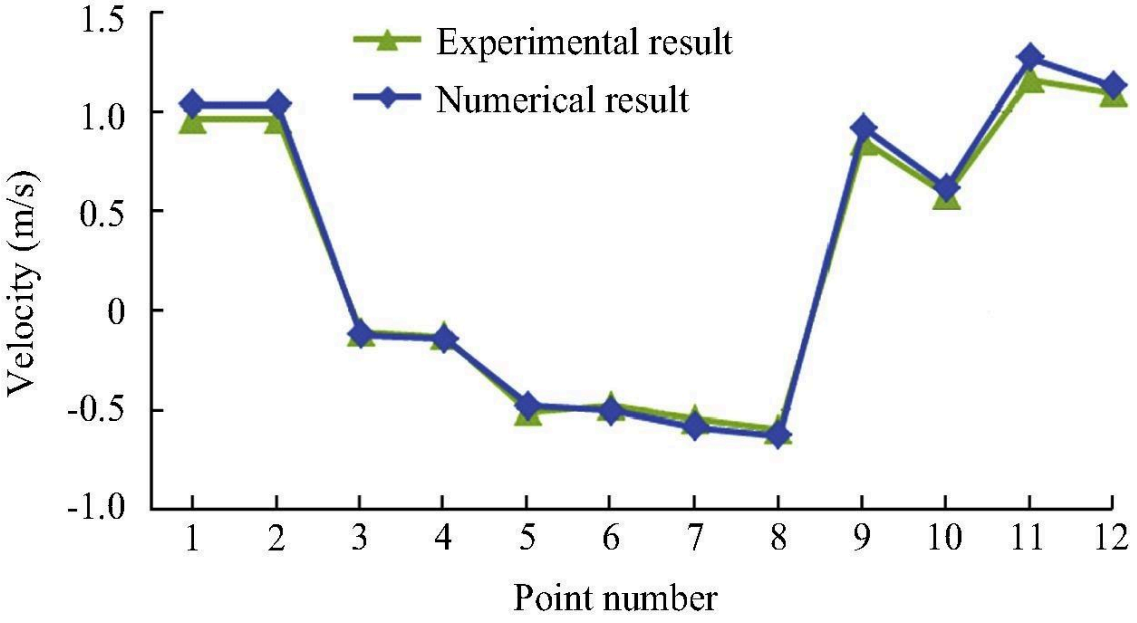
```python
import numpy as np
import matplotlib.pyplot as plt

# Define parameters
nx = 50  # Number of grid points in x-direction
ny = 50  # Number of grid points in y-direction
Lx = 10.0  # Length of the domain in x-direction
Ly = 10.0  # Length of the domain in y-direction
dx = Lx / (nx - 1)  # Grid spacing in x-direction
dy = Ly / (ny - 1)  # Grid spacing in y-direction
nu = 0.01  # Viscosity
rho = 1.0  # Density
dt = 0.001  # Time step size
nt = 1000  # Number of time steps
u = np.zeros((ny, nx))  # Velocity in x-direction
v = np.zeros((ny, nx))  # Velocity in y-direction
p = np.ones((ny, nx))  # Pressure

# Brattice sail parameters
sail_length = 2.0  # Length of brattice sail
sail_height = 1.0  # Height of brattice sail
sail_position_x = 5.0  # x-position of brattice sail
sail_position_y = 5.0  # y-position of brattice sail

# Define boundary conditions
u[:, 0] = 1.0  # Inlet velocity
u[:, -1] = 0.0  # No-slip boundary condition at outlet
v[0, :] = 0.0  # No-slip boundary condition at bottom
v[-1, :] = 0.0  # No-slip boundary condition at top

# Main loop
for n in range(nt):
    un = u.copy()
    vn = v.copy()
    pn = p.copy()

    # Solve momentum equations
    for i in range(1, nx - 1):
        for j in range(1, ny - 1):
            # Check if the current grid point is within the influence of the brattice sail
            if (i*dx > sail_position_x and i*dx < sail_position_x + sail_length and
                j*dy > sail_position_y and j*dy < sail_position_y + sail_height):
                # Set velocity to zero within the region influenced by the brattice sail
                u[j, i] = 0.0
                v[j, i] = 0.0
            else:
                # Use the original velocity calculation
                u[j, i] = (un[j, i] -
                           un[j, i] * dt / dx * (un[j, i] - un[j, i - 1]) -
                           vn[j, i] * dt / dy * (un[j, i] - un[j - 1, i]) +
                           nu * dt / dx**2 * (un[j, i + 1] - 2 * un[j, i] + un[j, i - 1]) +
                           nu * dt / dy**2 * (un[j + 1, i] - 2 * un[j, i] + un[j - 1, i]) -
                           dt / rho * (pn[j, i + 1] - pn[j, i - 1]) / (2 * dx) +
                           9.8 * dt)  # Add gravity term

                v[j, i] = (vn[j, i] -
                           un[j, i] * dt / dx * (vn[j, i] - vn[j, i - 1]) -
                           vn[j, i] * dt / dy * (vn[j, i] - vn[j - 1, i]) +
                           nu * dt / dx**2 * (vn[j, i + 1] - 2 * vn[j, i] + vn[j, i - 1]) +
                           nu * dt / dy**2 * (vn[j + 1, i] - 2 * vn[j, i] + vn[j - 1, i]))

    # Solve pressure equation
    for i in range(1, nx - 1):
        for j in range(1, ny - 1):
            p[j, i] = ((pn[j, i + 1] + pn[j, i - 1]) * dy**2 +
                       (pn[j + 1, i] + pn[j - 1, i]) * dx**2) / (2 * (dx**2 + dy**2))

    # Apply boundary conditions for pressure
    p[:, 0] = p[:, 1]  # dp/dx = 0 at inlet
    p[:, -1] = p[:, -2]  # dp/dx = 0 at outlet
    p[0, :] = p[1, :]  # dp/dy = 0 at bottom
    p[-1, :] = p[-2, :]  # dp/dy = 0 at top

# Plot results
x = np.linspace(0, Lx, nx)
y = np.linspace(0, Ly, ny)
X, Y = np.meshgrid(x, y)
plt.contourf(X, Y, np.sqrt(u**2 + v**2))
plt.colorbar()
plt.streamplot(X, Y, u, v, density=2, color='k')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Velocity magnitude and streamlines')
plt.show()
```

Link of Google Colab Notebook: Topic-02.ipynb

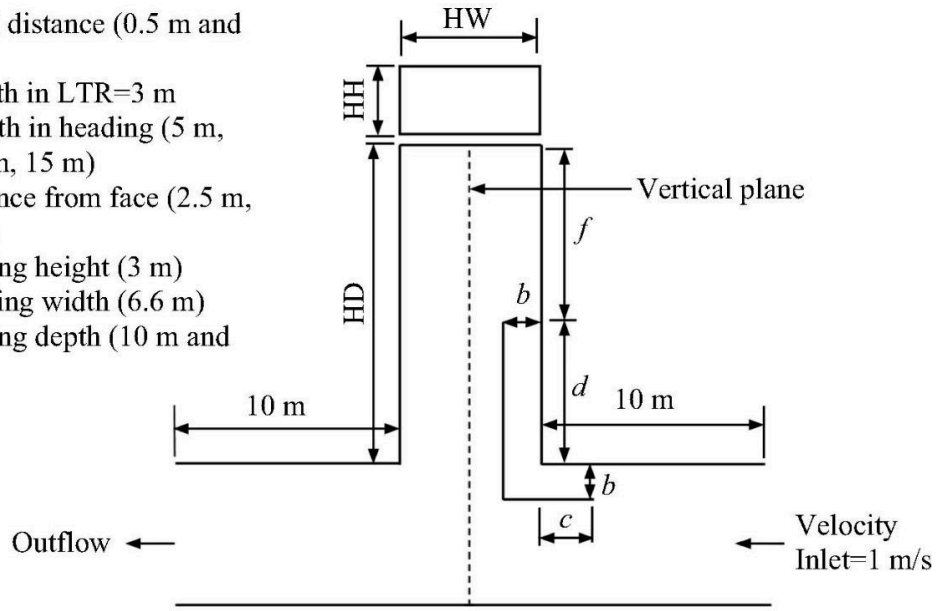
RESULTS:

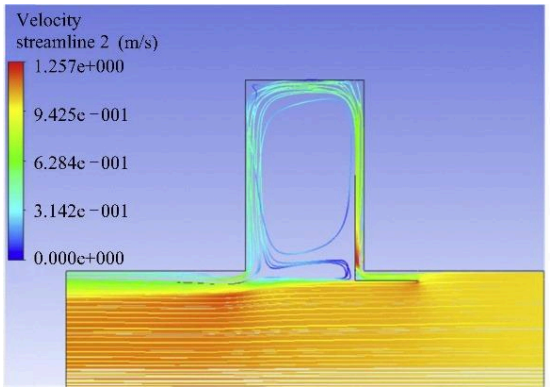

Velocity magnitude and streamlines
Y
X
0
2
4
6
8
10
10.5
9.0
7.5
6.0
4.5
3.0
1.5
0.0

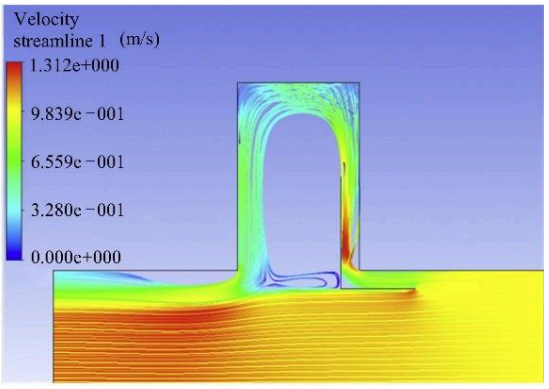

Velocity
contour LB exit
1.285e+000
1.157e+000
1.028e+000
8.997e-001
7.712e-001
6.427e-001
5.141e-001
3.856e-001
2.571e-001
1.285e-001
0.000e+000
Point 12
Point 10
Point 11
Point 9

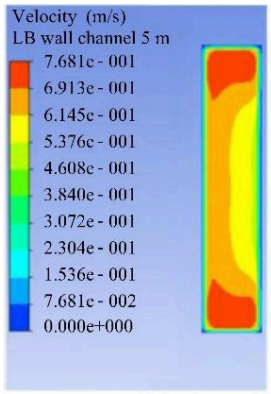


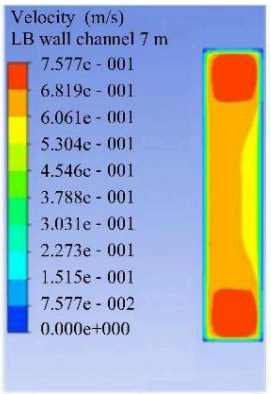
b=LB-wall distance (0.5 m and 1 m)
c=LB length in LTR=3 m
d=LB length in heading (5 m, 7.5 m, 10 m, 15 m)
f=LB distance from face (2.5 m, 5 m, 10 m)
HH=Heading height (3 m)
HW=Heading width (6.6 m)
HD=Heading depth (10 m and 20 m)

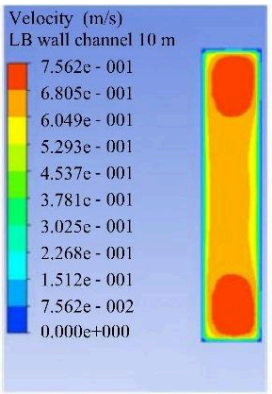


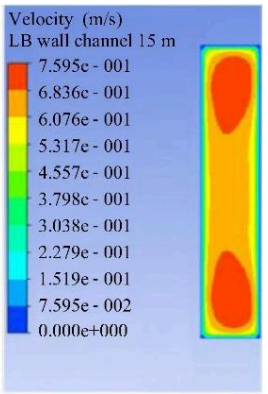

Velocity
streamline 2 (m/s)
1.257e+000
9.425e−001
6.284e−001
3.142e−001
0.000e+000

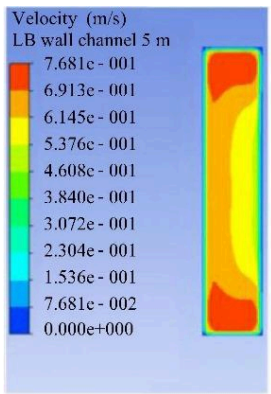

Velocity
streamline 1 (m/s)
1.312e+000
9.839e−001
6.559e−001
3.280e−001
0.000e+000

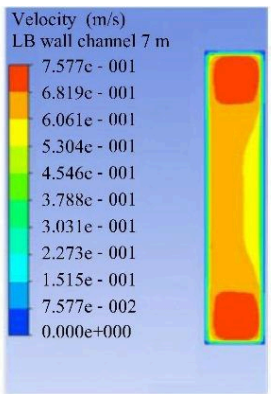

Velocity (m/s)
LB wall channel 5 m
7.681e-001
6.913e-001
6.145e-001
5.376e-001
4.608e-001
3.840e-001
3.072e-001
2.304e-001
1.536e-001
7.681e-002
0.000e+000

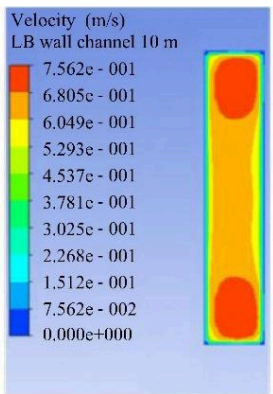

Velocity (m/s)
LB wall channel 7 m
7.577e-001
6.819e-001
6.061e-001
5.304e-001
4.546e-001
3.788e-001
3.031e-001
2.273e-001
1.515e-001
7.577e-002
0.000e+000

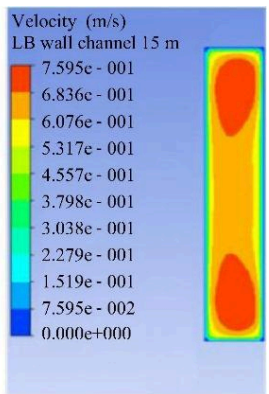

Velocity (m/s)
LB wall channel 10 m
7.562e-001
6.805e-001
6.049e-001
5.293e-001
4.537e-001
3.781e-001
3.025e-001
2.268e-001
1.512e-001
7.562e-002
0.000e+000

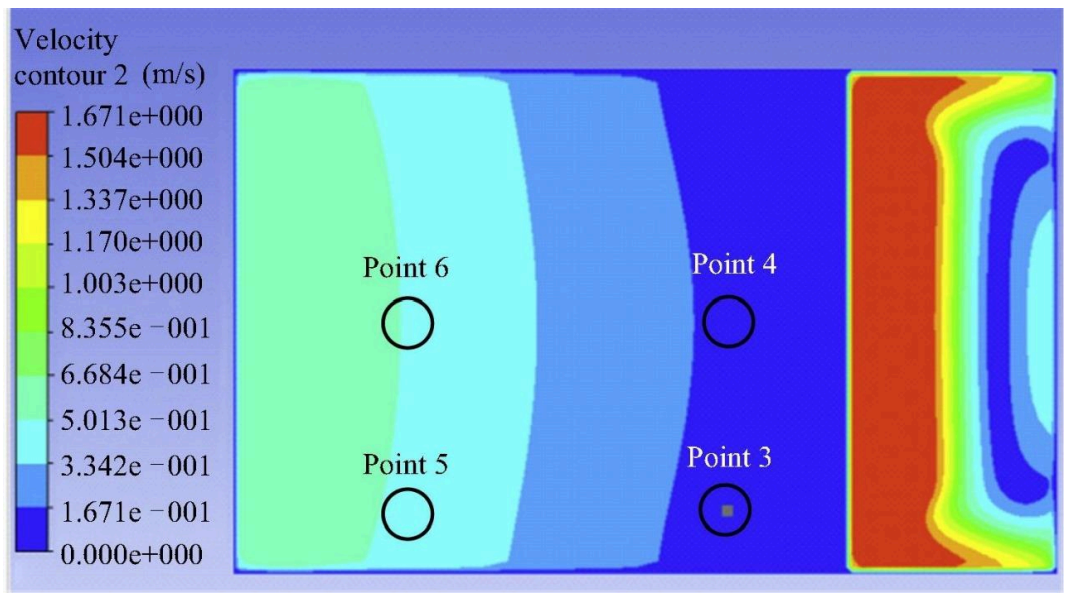

Velocity (m/s)
LB wall channel 15 m
7.595e-001
6.836e-001
6.076e-001
5.317e-001
4.557e-001
3.798e-001
3.038e-001
2.279e-001
1.519e-001
7.595e-002
0.000e+000

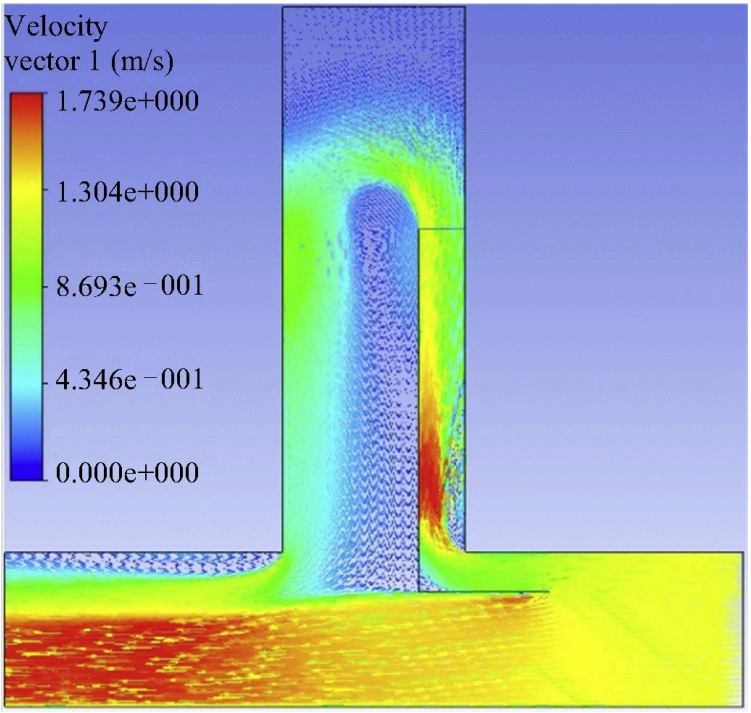

Velocity (m/s)
LB wall channel 5 m
7.681e-001
6.913e-001
6.145e-001
5.376e-001
4.608e-001
3.840e-001
3.072e-001
2.304e-001
1.536e-001
7.681e-002
0.000e+000

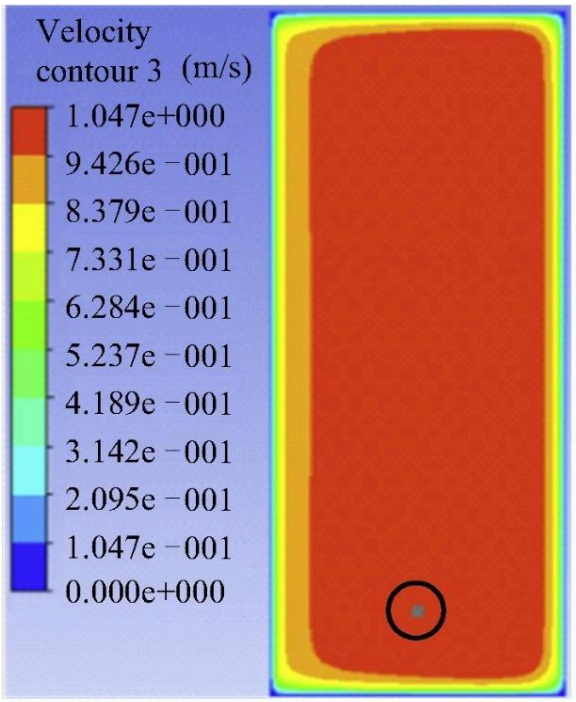

Velocity (m/s)
LB wall channel 7 m
7.577e-001
6.819e-001
6.061e-001
5.304e-001
4.546e-001
3.788e-001
3.031e-001
2.273e-001
1.515e-001
7.577e-002
0.000e+000


Velocity (m/s)
LB wall channel 10 m
7.562e-001
6.805e-001
6.049e-001
5.293e-001
4.537e-001
3.781e-001
3.025e-001
2.268e-001
1.512e-001
7.562e-002
0.000e+000

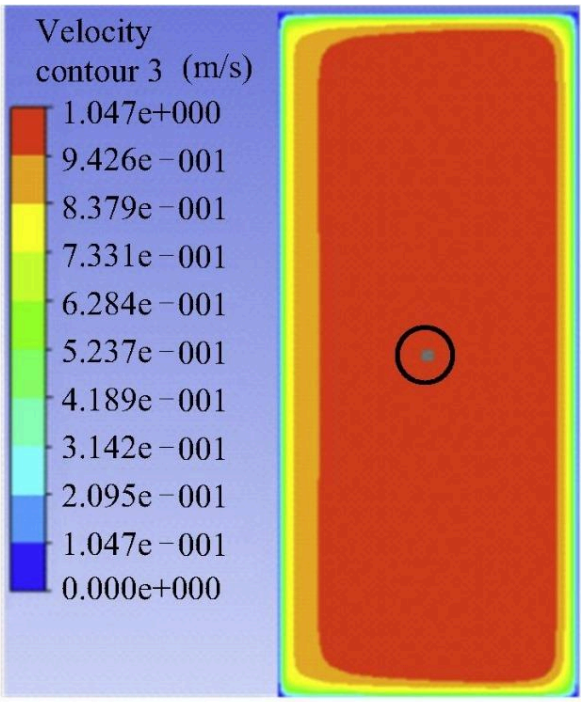

Velocity (m/s)
LB wall channel 15 m
7.595e-001
6.836e-001
6.076e-001
5.317e-001
4.557e-001
3.798e-001
3.038e-001
2.279e-001
1.519e-001
7.595e-002
0.000e+000


Velocity
contour 2 (m/s)
1.671e+000
1.504e+000
1.337e+000
1.170e+000
1.003e+000
8.355e−001
6.684e−001
5.013e−001
3.342e−001
1.671e−001
0.000e+000
Point 6
Point 4
Point 5
Point 3





(a) Velocity point 1



(b) Velocity point 2

INVESTIGATIVE INTERPRETATIONS:

Brattice sails are crucial components in underground mines for safety purposes due to their role in ventilation control. Here's why they are important:

- **Ventilation Regulation**: Underground mines require proper ventilation to dilute and remove hazardous gases, such as methane and carbon monoxide, and to provide fresh air for miners to breathe. Brattice sails help regulate airflow within the mine workings, ensuring that ventilation reaches all areas effectively.
- **Prevention of Gas Buildup**: By directing airflow, brattice sails help prevent the buildup of explosive or toxic gases in confined spaces within the mine. This reduces the risk of gas explosions or asphyxiation, safeguarding the lives of miners.
- **Temperature Control**: Brattice sails can also help in regulating temperature within the mine. Proper ventilation helps to dissipate heat generated by machinery and the natural geothermal heat from underground, preventing overheating and maintaining a comfortable working environment for miners.
- **Smoke and Dust Control**: In the event of a fire or other emergencies causing smoke or dust, brattice sails can be used to control the spread of these harmful substances throughout the mine. This is crucial for maintaining visibility and ensuring the safety of miners during evacuation or firefighting efforts.
- **Preventing Contamination**: Brattice sails can also help prevent contamination of air in different sections of the mine. This is particularly important in mines where different areas may have different air quality requirements, such as areas where blasting occurs or where harmful substances are being handled.

Overall, brattice sails play a vital role in maintaining a safe and healthy working environment in underground mines by facilitating proper ventilation, temperature control, and containment of hazardous substances. Their proper installation and maintenance are essential for ensuring the safety and well-being of miners.

CONCLUSION:

It shows a minor dependency of dimensionless streamlines on the magnitude of the main airstream velocity. Without a brattice, streamlines appeared weaker compared to when a brattice was present, showcasing the increased airflow into the cross-cut region with brattice usage. The study observed that the distance of the brattice from the wall influenced airflow enhancement.

The y-velocity in the cross-cut region exhibited different behaviours with varying brattice lengths. Longer brattice lengths demonstrated higher maximum velocities near the right wall of the cross-cut, suggesting improved ventilation mechanisms.

It provides valuable insights into predicting flow behaviours in mine sections employing brattices to redirect airflow into cross-cuts. It underscores the importance of brattice design and placement for optimising ventilation efficiency and contaminant removal within underground mines. Moving forward, the code can be improved to further develop the current model to explore additional brattice configurations and different locations within mine workings. This expansion could yield deeper insights into the optimal design and placement of brattices for enhancing airflow dynamics and maintaining safe working conditions in underground mining environments. Such advancements could significantly contribute to improving ventilation strategies and overall safety standards in underground mining operations.

REFERENCES:

- Van Heerden, J., and Sullivan, P., 1993. Application of CFD for evaluation of dust suppression and auxiliary ventilating systems used with continuous mines. in *Proceedings of the US Mine Ventilation Symposium*, Salt Lake City, UT, USA, pp 293-297.
- https://github.com/barbagroup/CFDPython#
- Numerical Simulation of Ventilation Air Flow in Underground Mine Workings (S.M. Aminossadati: The University of Queensland, CRCMining, Brisbane, Australia, K. Hooman: The University of Queensland, Brisbane, Australia)
- Kissell, F. N. and Matta, J. E., 1979. Face ventilation system for coal mines.

GROUP MEMBERS:

1. Archana Satapathy- 21MI10012
2. Ayush Sikarwal- 21MI33007
3. Bhoumik Mhatre- 21MI33008
4. Krish Alok Jain- 21MI10031
5. Manish Kumar- 21MI31017
6. Pranjal Goyal- 21MI31019
7. Rohan Kumar- 21MI10043
8. Siddharth Asthana- 21MI3EP28