UNIVERSITY OF
ARKANSAS

An Extended Report on 'Incorporating Two Weeks Open Source Software Lab Module in CFD and Fluids Courses'

Compiled by

Sumit Verma

Supervised By

Dr. R. Panneer Selvam

June 2021

Department of Civil Engineering, College of Engineering
University of Arkansas

2021 ASEE MIDWEST SECTION VIRTUAL CONFERENCE

UNIVERSITY OF
ARKANSAS

Organized By
American Society of Engineering Education

Incorporating Two Weeks Open Source Software Lab Module in CFD and Fluids Courses

Sumit Verma, Zahra Mansouri, R. Panneer Selvam
Department of Civil Engineering, BELL 4190 University of Arkansas, Fayetteville, AR 72701, USA

Abstract

To train future engineers and to equip them with necessary tools and skills for real-world problem solving, it is important to provide exposure to real-world problem solving by incorporating a software lab module while teaching engineering courses such as Computational Fluid Dynamics (CFD) and/or related Fluids courses. High cost of commercial software packages and limited number of licenses available for course instruction creates several challenges in incorporating commercial software packages in the instructional workflow. To circumvent such limitations, open-source software packages can be a good alternative as open-source software packages can be downloaded and used free of cost and thus provides a wider accessibility to students and practitioners. With the same motivation, in this contribution, an outline for implementing a two weeks course module by incorporating open source software in the instructional workflow is proposed and demonstrated by considering an example of wind flow around a building. The course module outlined in this work can also be extended to formulate a full fledge CFD course for instructional purposes. Besides the information provided in this paper, an extended report based on current work and the relevant case files are also shared via Github repository to assist readers in following and implementing the current work. With the help of information contained in this paper along with the extended report and uploaded case files, readers can install the open source software packages -'OpenFOAM' and 'ParaView', make their own simple case files, run simulations and visualize results, which is the ultimate motive of this work.

Keywords

Computational Fluid Dynamics, Building Aerodynamics, OpenFOAM, ParaView.

1. Introduction

1.1 Need for Open Source CFD Program for Class Instruction

When teaching engineering courses such as Computational Fluid Dynamics (CFD), Wind Engineering and related Fluids courses, instructors are challenged to create a course content, which not only provides a good understanding about the fundamental topics of CFD and Wind Engineering but also equips students with tools for solving real-world problems of fluid flows. To develop a reliable CFD model from scratch, a sound understanding of several topics such as Fluid Dynamics, Numerical Methods, Programming, etc. is required. Developing proficiency in students on these topics within a semester or over a couple of semesters requires an overwhelming amount of work in short time. Some engineering programs such as that at Cornell University offers a course on Engineering Simulation using commercial software package Ansys Fluent [1]. While commercial software package Ansys Fluent provides academic license for students free of cost but the functionalities available in academic license are restricted. The cost of license for different

commercial software packages can vary from one to another; however, in general, license cost for single seating roughly amounts to about $10,000 per year. Such high prices makes the commercial software packages almost unaffordable to purchase an individual license.

The goal of engineering programs in the nation is to train students with necessary knowledge and skills enabling them to apply those skills beyond classrooms (such as industry position or research position); however, training students using commercial software packages is too expensive as stated above. In addition to high costs, there are several other challenges such as availability of a limited number of license which may not be sufficient for the entire class. For instance, if only 8 licenses of commercial software are available but a class consists of 15 students, then, an alternative strategy of instruction such as dividing the class into groups and teaching the groups in different shifts has to be adopted. As the same teaching material has to be delivered to different groups at different times, the effective class hours gets comprised for learning new things. Besides, students may only be able to access the commercial software at some selected labs in the university. Due to limited available licenses, students may have to wait for their turn to run simulation jobs depending on the load on machines. Similarly, if students have to solve a fairly complex problem for their research, which requires a long simulation time (such as several days to weeks), then, the computers gets locked up for a long time as the already-running simulation might make most of the processor cores in the machine busy. As a result, no further jobs can be submitted on the same machine for a long time restricting its accessibility/usage until the simulation is complete. In addition to these challenges, there are further bottlenecks if the course needs to be delivered remotely such as the instructor has to ensure first that students can access and use commercial software installed within the university computer systems via remote access, which adds further challenges in course delivery. In order to tackle the challenges stated above, there is a need for an alternative to commercial software packages for wider accessibility of the class to a large group of students. In that regard, open-source software can be a good substitute in place of commercial software packages for teaching CFD, Wind Engineering and related Fluids courses. The following section provides an outline for implementing a course module by incorporating open source software packages-OpenFOAM and ParaView, in the instructional workflow.

1.2 Suggested Course Module for Instruction

In the department of Civil Engineering at University of Arkansas, a 3 weeks module on 'Introduction to Computational Fluid Dynamics' was introduced during the course offering of CVEG-5383 (Finite Element Methods in Civil Engineering), in which students were introduced to real-world problem solving of fluid flows using research codes adapted for classroom teaching. Based on similar modality, the entire course on CVEG 563V (CFD for Wind Engineering), was taught as a combination of theoretical lectures and real-world problem solving using adapted research codes. This modality was found to be useful in teaching fundamental concepts of the subject matter while also introducing the students to current research areas in the discipline using customized research codes adapted for teaching purposes.

With some changes applied to teaching modality described above, a new instruction modality for teaching CFD, Wind Engineering or related Fluids courses is proposed in this paper. The new instruction modality consists of two weeks open source software lab component using open source software packages - OpenFOAM and ParaView with real-world problem solving exercises. For demonstration purposes, an engineering problem of practical significance, i.e. wind flow around a

building is considered for this work. The work described in this paper can be assimilated as a 2 weeks module software lab component in course offerings on CFD or related Fluids courses. By expanding the scope of course outline described in this paper such as that by incorporating work described in [2-ZM paper] with additional relevant topics, a full fledge CFD course can be formulated. However, as this contribution is intended to serve only as an introductory outline, so, the details on installation procedure of OpenFOAM and ParaView, setting up simple flow problems, obtaining and visualizing the simulated results are discussed here.

1.3 Objectives of Current Work

(i) To provide a detailed outline of the installation process of 'OpenFOAM' and 'ParaView' in Windows 10 operating system.

(ii) To demonstrate the workflow of setting up the case files and running simulations including explanation of case file structure in OpenFOAM.

(iii) To demonstrate the procedure of obtaining data visualizations (such as contour plots, velocity vector plots, streamline plots, etc.) and extracting pressure profiles for further analysis of simulated results using an extended report based on this paper.

Before delving into detailed description of problem and solution procedure, in the following section, a brief introduction to OpenFOAM and ParaView (open source visualization software for solution obtained from OpenFOAM) and the key aspects of installation procedure are described briefly.

2. A Brief Introduction to 'OpenFOAM' and 'ParaView'

OpenFOAM stands for 'Open Source Field Operation and Manipulation', which is a C++ toolbox for solving problems of continuum mechanics such as that applied to fluid flow and heat transfer problems in computational fluid dynamics (CFD). A brief introduction and overview of setting up some problems including meshing capabilities, solvers and post-processing tools available in OpenFOAM is described in OpenFOAM user guide [3]. OpenFOAM not only provides a platform for solving a variety of problems encountered in fluid dynamics, heat transfer, electromagnetics and multiphase fluid flow problems, etc. but it also provides a platform to modify and develop custom solvers, applications and libraries as per the need of users. Further details about 'OpenFOAM' programming can be obtained from OpenFOAM programmer's guide [4]. However, to develop customized applications and libraries, user should have some familiarity with C++ semantics such as object-oriented programming, classes and objects, operator overloading, inheritance, etc. Further details about the organization of namespace, classes and file system available in OpenFOAM can be obtained from OpenFOAM source code guide [5]. However, for the scope of work described in this paper, the information in [3] is enough to set up the problem and obtain solutions. After obtaining solution from OpenFOAM, there is a need for some visualization utility for visually inspecting the obtained solution and for further post-processing of results. For that purpose, another open source software called 'ParaView' is considered for this work. Different techniques for visualizing and analyzing scientific data in 'ParaView' are covered in depth in the 'ParaView' documentation manual [6]. Besides, readers can also refer to section-B

of this report on 'ParaView Visualization' for a detailed step by step explanation along with illustrations to obtain visualizations reported in this work.

After a brief introduction to essential software components used in current work, installation procedure of those two software components are briefly described in the following text. In section 2.1 below, the important aspects of OpenFOAM and ParaView installation in Windows 10 operating system (OS) is explained and references are cited to help the readers in installation process. For a detailed step by step procedure along with illustrations, readers can refer to section-A of this report on 'OpenFOAM Installation' below.

2.1 Setting up OpenFOAM and ParaView in Windows 10

In the older versions of Windows OS, the feature 'Windows Subsystem for Linux' (WSL) was not available. However, in Windows 10, users can activate WSL by turning on the 'Developers Mode' feature and then installing 'Ubuntu' (a Linux distribution) in WSL platform. For a step by step process of activating 'Developers Mode' in Windows 10 including OpenFOAM installation, the Youtube video from [7] can be followed. Besides, readers can also refer to another Youtube video [8], which provides a good explanation for installing 'Ubuntu' in Windows 10. These resources are sufficient for the readers to download and install 'Ubuntu' in Windows 10 OS. Now, readers can proceed ahead with the installation process for OpenFOAM, which can be downloaded from OpenFOAM website cited in [9]. After downloading OpenFOAM in the PC, readers can follow along the Youtube video cited in [7] to proceed ahead with installation of OpenFOAM. Similarly, to install Paraview in PC, the download page on Paraview website can be referred [10]. These resources and the cited references should be enough to download and set up OpenFOAM as well as ParaView.

After installing and setting up the environment to run simulation jobs, an example problem (simulating wind flow around a cubical building) is considered that would fit very well for the two weeks open source software lab module proposed in this work. In the following section, the details of problem statement, procedure of modeling geometry for simulating wind flow around building as well as meshing is presented.

3. Description of Problem (Wind Flow around a Building)

In this work, wind flow around a cubical building model is studied using OpenFOAM. For that purpose, it is necessary to choose a suitable region in space around the building model (the chosen region in space around the building is also called computational domain) in which governing fluid flow equations are solved to analyze the wind flow pattern around a building.

For this work, a cube of dimension '1H' is considered for the building model, where 'H' is the height of building. Similarly, the computational domain is made up of a cuboid of dimension '12H' in X-direction, '7H' in Y-direction and '5H' in Z-direction respectively. As shown in Fig. 1, the direction of X-axis is the stream-wise direction and thus the flow enter computational domain through face "ACDB". The flow interacts with cubical building model located inside the computational domain and then exits the domain from face "EFHG". The interaction of fluid with the building model produces aerodynamic forces on the building, which will be computed and plotted in the later section. In Fig. 1, the face "ABFE" is the top face whereas the face "BDHF" is

the front face of computational domain. The top view and the front view of computational domain are shown in Fig. 2 (a) and (b) respectively.

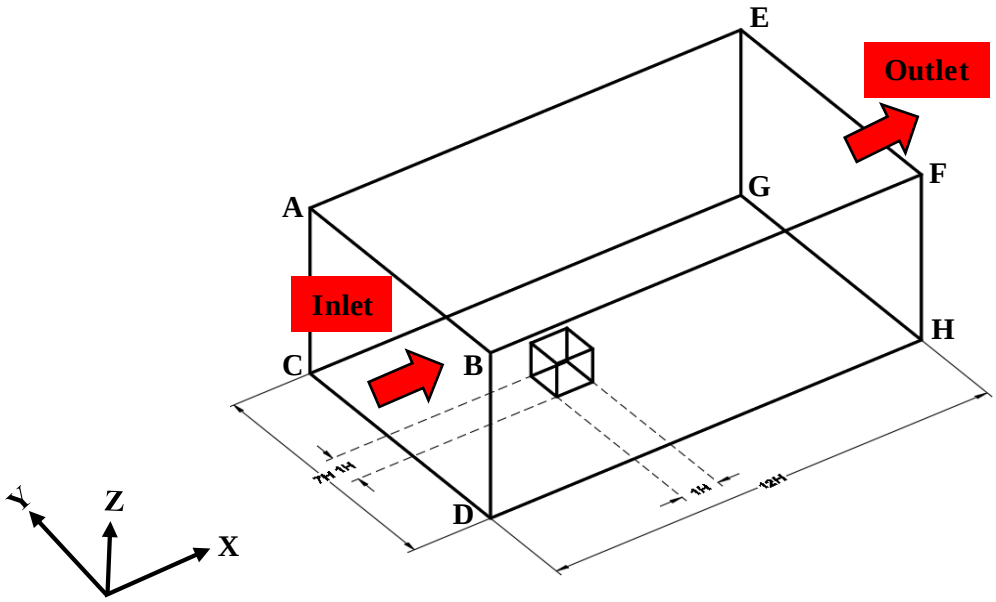
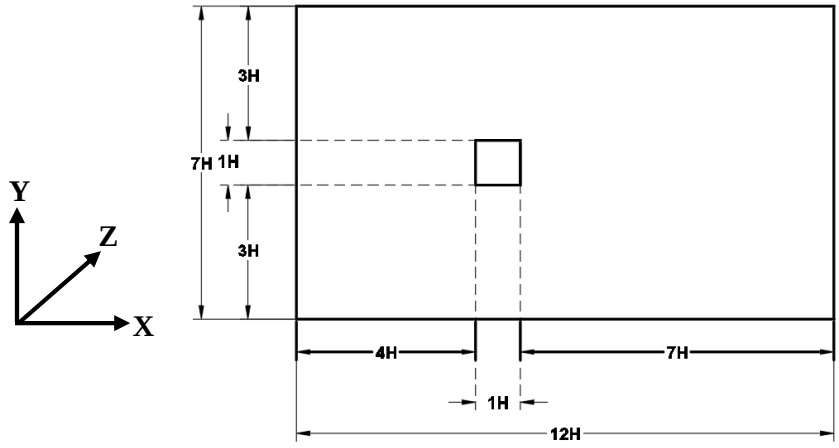
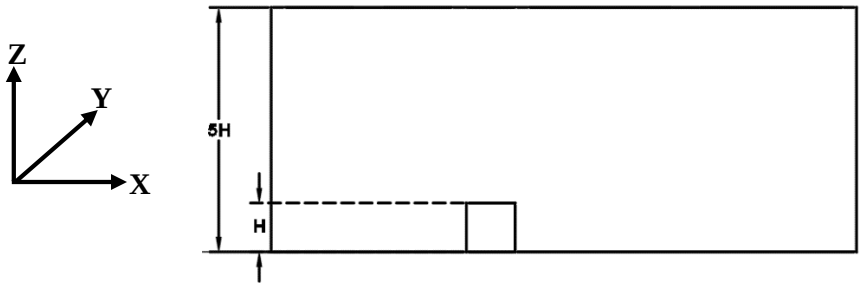


Fig. 1. Isometric view of computational domain with a cubical building model inside



(a) Top view



(b) Front View

Fig. 2. Orthographic projection of computational domain with building (a) Top view (b) Front view

4. Hierarchical Structure (File System) of a Case Directory in OpenFOAM

Before describing the details about meshing and solvers used in the current work, the hierarchy of file system in the case directory named "buildingUniform", used for current work is described briefly.

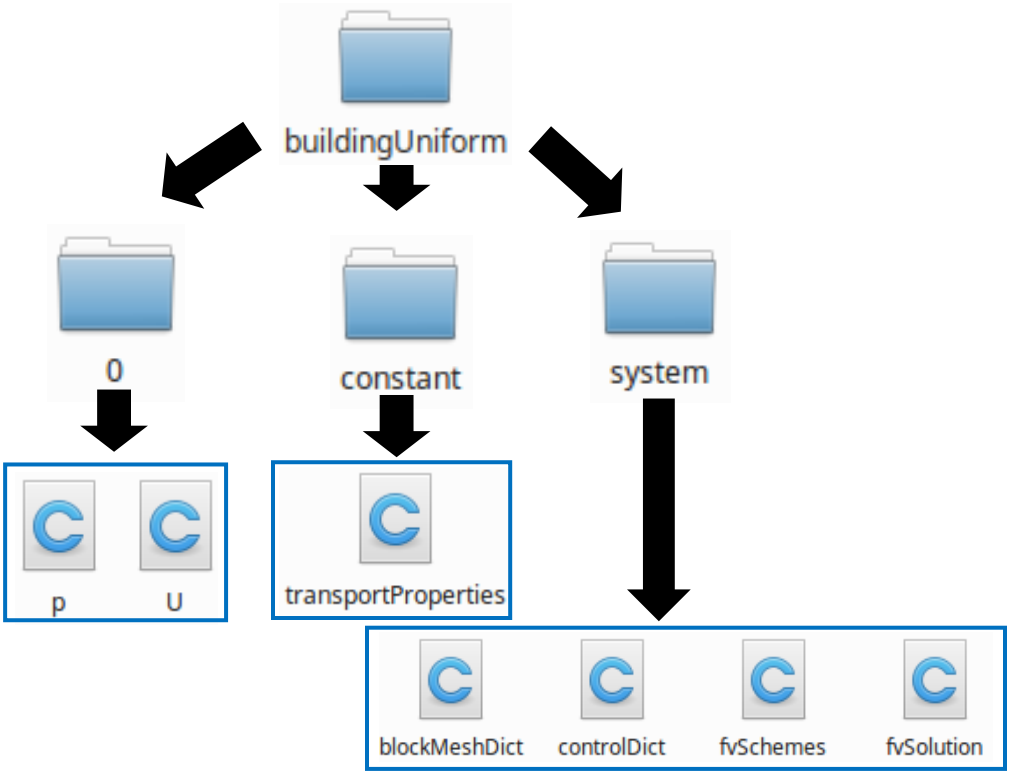


Fig. 3. Hierarchical structure of case directory considered for current problem in OpenFOAM

It should be noted that the hierarchical structure shown in Fig. 3 pertains to the problem described in current work. For some other work, the hierarchical structure may contain additional files or directories depending upon the problem; however, the basic hierarchical structure shown in Fig. 3 would stay the same even for other complicated problems. Finally, the files contained inside different directories (in hierarchical structure of Fig. 3) and the functions intended to be performed by each of those files during OpenFOAM simulation are described briefly in Table 1. For further details, readers are directed to OpenFOAM documentation guide [11].

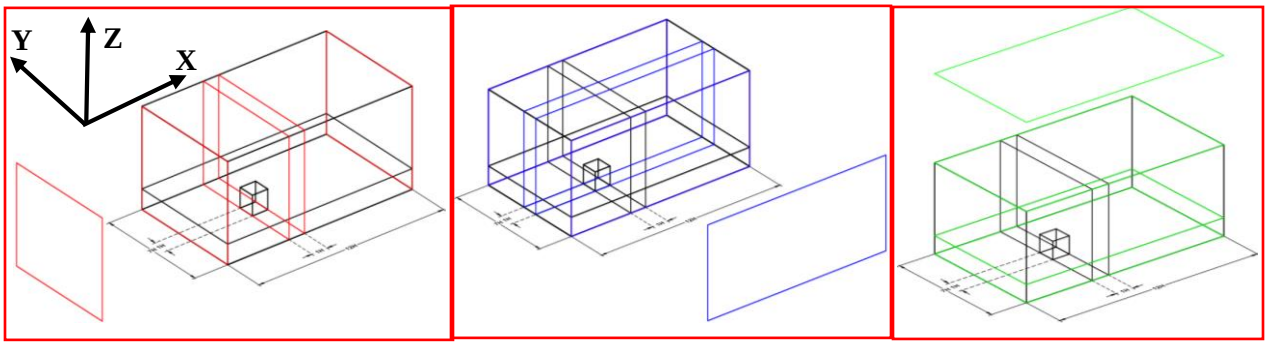
Table 1. Files contained inside different directories and intended function during OpenFOAM simulation

Directory Name	Files contained	Function intended for setting up and running simulation
0	p	Initial and boundary conditions for kinematic pressure
	U	Initial and boundary conditions for velocity
constant	transportProperties	kinematic viscosity of fluid under consideration
	blockMeshDict	Geometrical details for meshing of computational domain
	controlDict	Solver and solver settings for running simulation

system	fvSchemes	Numerical schemes used by the solver for different components of NS Equation such as time derivative, convection term, diffusion term, etc.
	fvSolution	Linear system solvers for 'p' and 'U', relaxation factors and velocity-pressure coupling algorithm, etc.

5. Meshing of Computational Domain (Pre-Processing)

In CFD, the governing equations of fluid flow, which comprises of a set of partial differential equations (pdes) are solved by discretizing the pdes into a system of linear algebraic equations. For that purpose, the entire computational domain is discretized into a number of small regions, also called computational stencils or cells. The computational domain used in current work is formed from different blocks, which in turn are produced due to intersecting planes parallel to X, Y and Z-axes.



(a) (b) (c)

Fig. 4. Demonstration of computational domain formed by intersection of different planes normal to (a) X-axis (b) Y-axis and (c) Z-axes

For instance, as shown in Fig. 4 (a), a plane normal to X-axis is placed at 4 different locations, i.e. $x^* = 0, 4, 5, 12$ (shown by red colored rectangles) whereas planes normal to Y-axis are placed at $y^* = 0, 3, 4, 7$ and planes normal to Z-axis are placed at $z^* = 0, 1$ and 5. When these planes intersect, several blocks are formed. The assembly of several blocks together makes up the computational domain for the current problem. For demonstration purposes, two such blocks (i.e. BLOCK-0 and BLOCK-2, which are color-coded red), formed due to intersecting planes at different locations are shown in Fig. 5 below.

5.1 Concept of Multi-block Grid

As the geometry of computational domain considered for current work is relatively simple consisting of hexahedrons, so, 'blockMesh' utility available in OpenFOAM is considered for the current work. Further details about 'blockMesh' utility can be obtained from [12].

To clarify the concept of multi-block grid, isometric view of two block units (i.e. BLOCK-0 and BLOCK-2) are shown in Fig. 5 (a) (in red color) while other blocks are not shown to avoid congestion of labels in the figure.

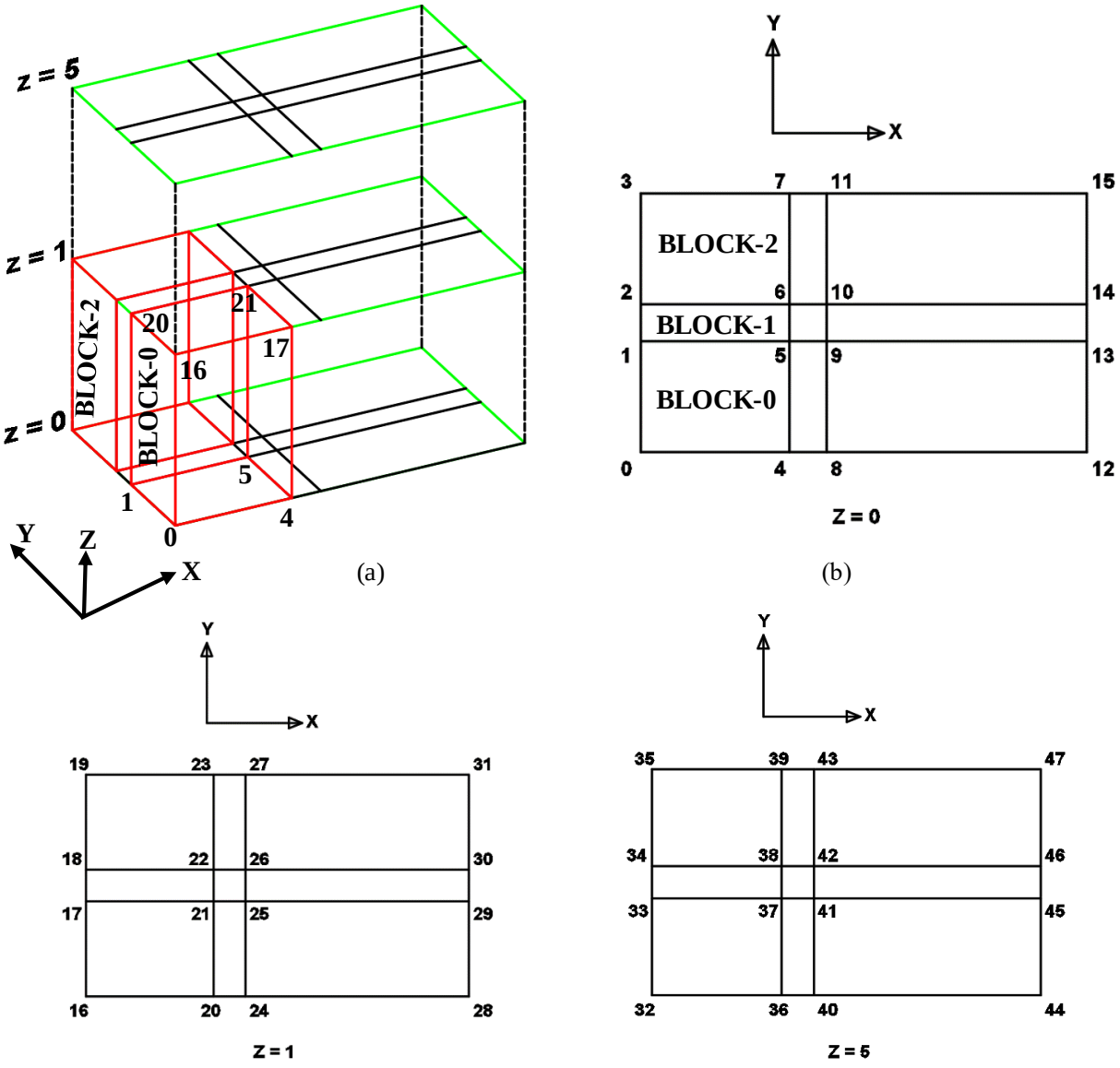


Fig. 5. (a) Isometric view of computational domain demonstrating some blocks used to make the computational domain and Numbering scheme for multi-block grid in (b) z = 0 plane (c) z = 1 plane (d) z = 5 plane (Sketches NTS)

Also, the regions (or faces) formed due to intersection of different planes are shown in Figs. 5(b)-(d). In Fig. 5(b), the regions (or faces) formed due to intersection of planes at an elevation of $z^* = 0$ is shown. Similarly, in Fig. 5(c), the regions (or faces) formed due to intersection of planes at elevation of $z^* = 1$ is shown whereas that in Fig. 5(d) includes the region (or faces) at elevation of $z^* = 5$.

To create a structured mesh consisting of hexahedral cells using 'blockMesh', certain set of points (or vertices) should be defined first. This is done in a file called 'blockMeshDict' inside the system directory of case file (Refer section 4). The numbering of vertices adopted for the current work is shown in Fig. 5 (b)-(d). For demonstration purposes, some vertices including their x, y and z-

coordinates are shown in Fig. 6(a). For instance, the coordinates of vertex '0' is x = 0, y = 0 and z = 0 and that of vertex '16' is x = 0, y = 0 and z = 1 and so on in Fig. 6(a).

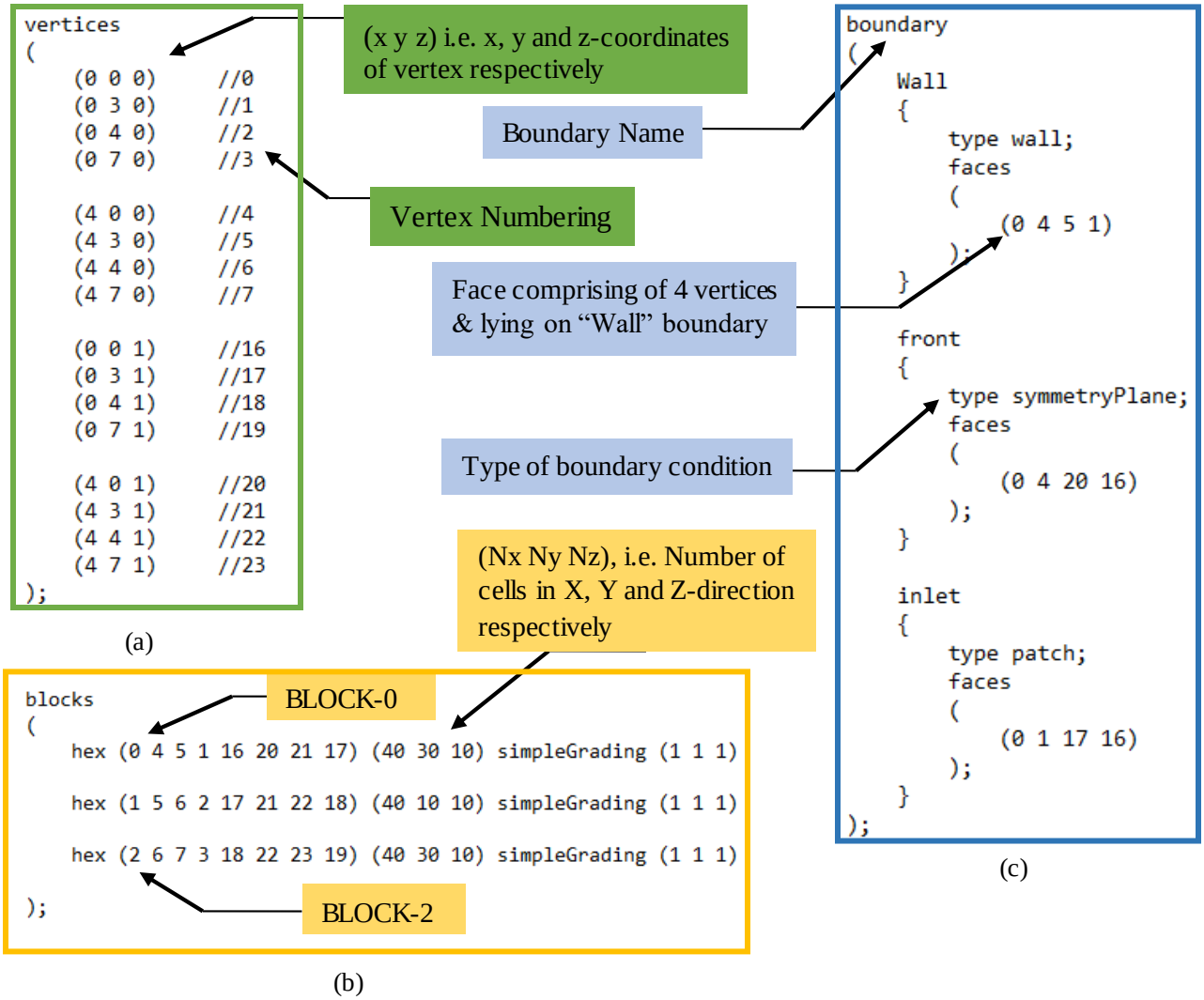
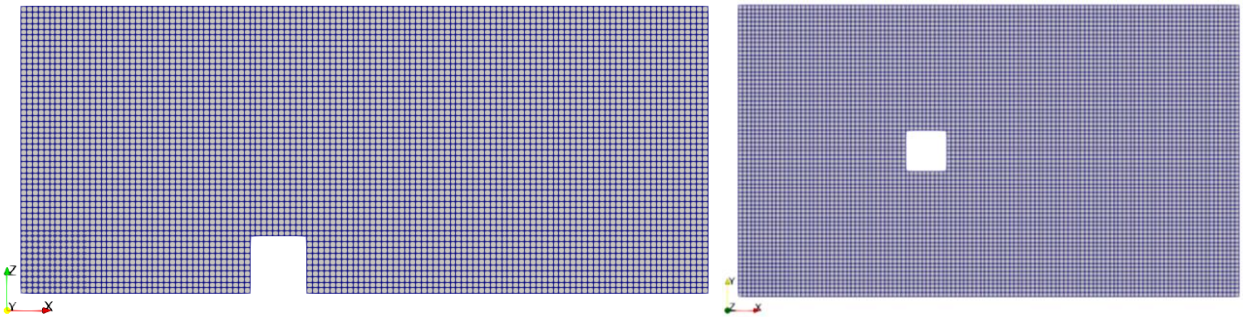


Fig. 6. Demo of (a) Vertex numbering including x, y and z-coordinates (b) vertices making hexahedral blocks for meshing of computational domain (c) boundary faces formed by combining different vertices, boundary name & type

Using the defined set of points (or vertices), a number of hexahedral blocks are created for the computational domain. Two blocks (i.e. BLOCK-0 & BLOCK-2) and the vertices that makes up those blocks are shown in Fig. 6(b) for demonstration purposes, i.e. 'BLOCK-0' is made up of vertices 0,4,5,1,16,20,21,17 whereas 'BLOCK-2' is made up of vertices 2,6,7,3,18,22,23,19 and so on (Refer section 5.3 of OpenFOAM user guide [3]). It should be noted that while specifying the vertices of block, the vertices should be listed in anticlockwise direction. For instance, the bottom plane of 'BLOCK-0' consists of vertices 0, 4, 5 and 1, which are listed in anticlockwise order (Refer Fig. 5 (b)) whereas the top plane of 'BLOCK-0' consists of vertices 16, 20, 21 and 17, which are again listed in anticlockwise order. The same idea is used while specifying the vertices of other blocks of the computational domain. In addition, the number of cells in X, Y and Z-direction are taken as 40, 30 and 10 respectively for 'BLOCK-0' (Refer. Fig. 6(b)). As the length in X, Y and Z-direction for 'BLOCK-0' are 4H, 3H and H respectively, 40 cells along X, 30 cells

along Y and 10 cells along Z-direction implies that a grid resolution of 0.1H is achieved in each of X, Y and Z-directions. The same grid resolution applies to the remaining blocks of computational domain in the current work.

As shown in Figs.5-6, the computational domain used for current work is composed of multiple individual blocks sharing some common vertices and faces. Although the multi-block arrangement consists of individual block units, the units are glued together through common vertices and faces and thus act as a combined whole while solving the governing equations. Similarly, some faces lying on respective boundaries such as the face consisting of vertices 0, 4, 5, 1 and lying on 'Wall' boundary and the face consisting of vertices 0, 1, 17, 16 and lying on 'inlet' boundary is shown in Fig. 6(c). It is to be noted that the vertices, blocks and boundary faces shown in Fig. 6 are for demonstration and explanation purpose only and do not represent the entire set of vertices, faces and blocks used in the current work. For complete details, readers are directed to uploaded case files available at [Github link].



(a) (b)

Fig. 7. Mesh of computational domain in (a) XZ-plane at y = 3.5 (b) XY-plane at z = 0.9

Using the multi-block grid concept, altogether 17 hexahedral blocks were finally used to make the computational domain in current work. Using grid resolution of 0.1H in each of X, Y and Z-direction, the total number of cells in the mesh was obtained as 419,000 (Refer Fig. 7 for mesh used in current work).

6. Governing Equations and Solver Details

Once the geometry of computational domain is created using multi-block grid and meshing is complete, the governing equations needs to be solved next. As the problem at hand deals with wind flow around a building and the Navier-Stokes (NS) equation when implemented with suitable boundary conditions can describe the wind flow behavior around a building, so, unsteady incompressible NS equation was used for the current problem. However, only laminar flow conditions are assumed for current work neglecting any turbulence effects in the wind as the Reynolds number (Re) of flow is low, i.e. Re = 100. Nevertheless, turbulent fluctuations are inherent property of wind flow in real-life scenarios and thus, modeling of wind flow around buildings including turbulence effects depicts a real-life flow scenario better and is described in detail in another follow-up paper [2].

6.1 Non-dimensional form of NS Equation

While seeking computational solution to problems, the governing equations are usually expressed in non-dimensional form. Large numerical values of variables expressed in physical units can lead to diverging solutions due to non-linearity of NS equation. So, for greater numerical stability, non-dimensional form of NS equations are preferred in CFD. For the current work, the governing equations are non-dimensionalized using two values (a) reference length taken as the building height (1H) and (b) reference velocity at inlet ($U_\infty^* = 1$). The governing equations for unsteady incompressible laminar flow in non-dimensional form expressed in vector notation are as follows:

$$\textbf{Continuity equation: } \nabla . \mathbf{U}^* = 0 \tag{1}$$

$$\textbf{Momentum Equation: } \frac{\partial \mathbf{U}^*}{\partial t^*} + \mathbf{U}^* \nabla . (\widetilde{\mathbf{U}^*}) = - \nabla \mathbf{p}^* + {}^{1}/_{\mathbf{Re}} \nabla . (\nabla \mathbf{U}^*) \tag{2}$$

The non-dimensional values are related to physical units by the following relations. Considering 'L', 'U', 'p' and 't' as the dimensional variables for length, velocity, pressure and time respectively in dimensional form of NS equation, the corresponding non-dimensional counterparts are given by 'L*', 'U*', 'p*' and 't*' and are related by the relation listed in (3).

$$L^* = {}^{L}/_{L_\infty}; U^* = {}^{U}/_{U_\infty}; p^* = {}^{p}/_{\rho U_\infty^2}; t^* = {}^{U_\infty t}/_{L_\infty} \tag{3}$$

Further details on NS equation can be obtained from [13] and [14] whereas about the non-dimensionalization of NS equations can be obtained from [15].

6.2 Initial and Boundary Conditions

The governing equations described by Eqns. (1) and (2) above is not enough to obtain wind flow around a building in computational domain. Besides, the governing equations, initial and boundary conditions are also necessary to solve and obtain wind flow around the building. It is stressed out that proper initial and boundary conditions are critically important to obtain a physically realistic solution or even to obtain a solution. As the momentum equation in (2) has non-linearity in the convection term, so, improper initial and boundary conditions provided for the problem may cause the solution to diverge very quickly.

In Fig. 8(a), the isometric view of computational domain is shown including different boundary faces such as inlet, outlet, etc. Further details about the name of boundary faces, their type, color coding for different faces from Fig. 8(a) and their mathematical form are described in Table 2 below. Similarly, the boundary faces of computational domain where wall/no-slip boundary condition (BC) is applied is shown in Fig. 8(b). The boundary faces where no-slip condition is applied are collectively categorized under the name "FaceGroup1" for convenience of naming in Table 2.

As shown in Fig. 8(b), the faces on which wall BC is implemented are as follows: DKLS, SLMU, UMNC, KOPL, MQRN, OHTP, PTVQ, QVGR, MLZW, LPYZ, PQXY and QMWX.

For the initial condition, both velocity and pressure are considered '0' throughout the computational domain, i.e. $\mathbf{U}^* = \mathbf{0}$ (or $u^* = v^* = w^* = 0$) and $p^* = 0$.

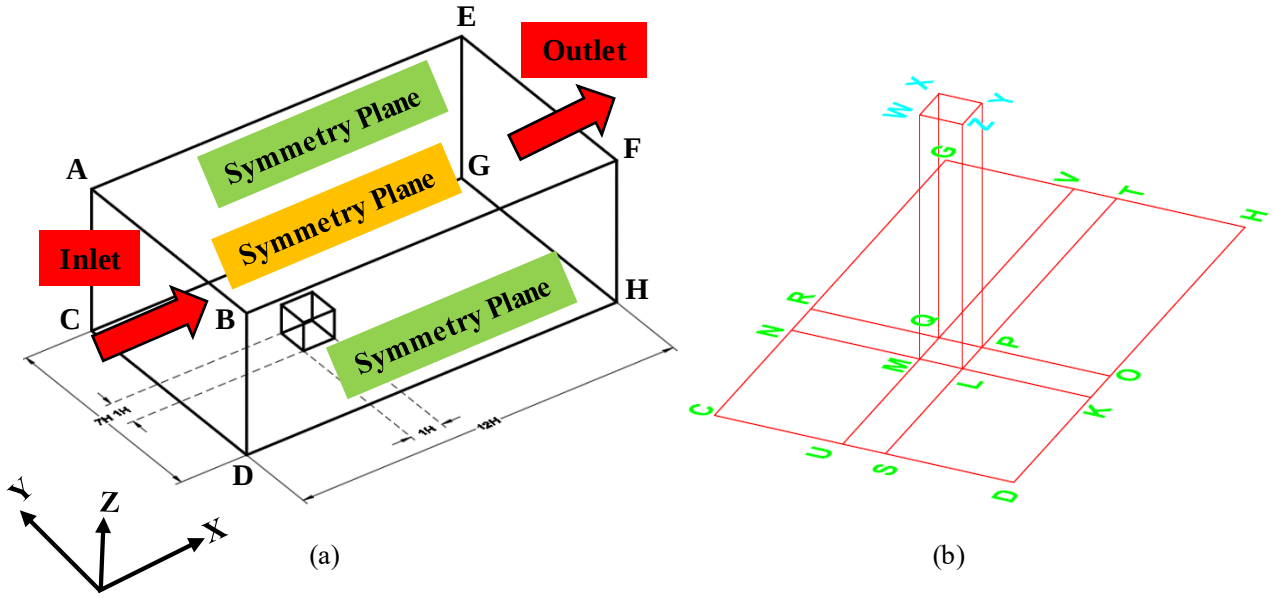


Fig. 8. (a) Isometric view of computational domain showing different boundary faces (b) Perspective view of computational domain showing 'wall' boundary faces only

Table 2. Description of boundary faces, types and boundary conditions for different faces in computational domain

Boundary name	BC type	Face	Color Coding	BC in mathematical form
Inlet	patch	ABDC	Red	$\frac{\partial p}{\partial n} = 0$; u = 1.0; v = w = 0
Outlet	patch	EFHG	Red	p = 0; $\frac{\partial u}{\partial n} = \frac{\partial v}{\partial n} = \frac{\partial w}{\partial n} = 0$
Wall	wall	FaceGroup1	-	u = v = w = 0; $\frac{\partial p}{\partial n} = 0$
Top	symmetryPlane	AEFB	Orange	w = 0; $\frac{\partial u}{\partial n} = \frac{\partial v}{\partial n} = 0$; $\frac{\partial p}{\partial n} = 0$
Front	symmetryPlane	BFHD	Green	v = 0; $\frac{\partial u}{\partial n} = \frac{\partial w}{\partial n} = 0$; $\frac{\partial p}{\partial n} = 0$
Back	symmetryPlane	AEGC	Green	v = 0; $\frac{\partial u}{\partial n} = \frac{\partial w}{\partial n} = 0$; $\frac{\partial p}{\partial n} = 0$

6.3 Solver selection for current problem in 'OpenFOAM'

Several solvers are available in OpenFOAM to solve incompressible fluid flow problems such as "**simpleFoam**", "**icoFoam**", "**pisoFoam**", etc. Further information about the different solvers available in OpenFOAM is available in [16]. As the current work deals with unsteady incompressible fluid flow condition without consideration for turbulent fluctuations, thus, the suitable solver for the current case is "**icoFoam**" and it is chosen for this work. After selecting the solver, it is necessary to specify the simulation time step for transient simulation and also the total time for which the simulation needs to be carried out. In the following section, the keywords that control simulation time step, total simulation time, interval for writing data files for visualization, etc. are described. In Table 3, as the value of 'writeInterval' is set at n = 500, it implies that the every 500 time-steps the data file will be stored for post-processing or visualization purpose. Also,

as the time-step size is set at $\triangle t = 0.0005$, thus, the data files will be written after every (n x $\triangle t$ = 500 x 0.0005) 0.25 time units. Futher details about the 'icoFoam" solver in OpenFOAM can be obtained from references [17] and [18].

Table 3. Important Keywords for controlling time-step, total simulation time with description for 'icoFoam' solver

Keyword	Values	Description
startTime (t_o)	0	Start time for simulation
endTime (t_e)	10	End time for simulation
deltaT ($\triangle t$)	5e-4	time step size for simulation
writeInterval (n)	500	write interval of data files for visualization

7. Results/Post-processing

Visualization plots are convenient means to inspect the flow field and to derive meaning flow information from the simulated results. In section 6.3, the manner in which data files are stored by setting up the write interval inside "controlDict" file was explained. For current work, simulation was run for a total time of 10 units and the data files were written by solver every 0.25 time units. Using the written data files, the following visualization plots are made. These visualization are from the final time-step of simulation, i.e. corresponding to $t^* = 10$ time units.

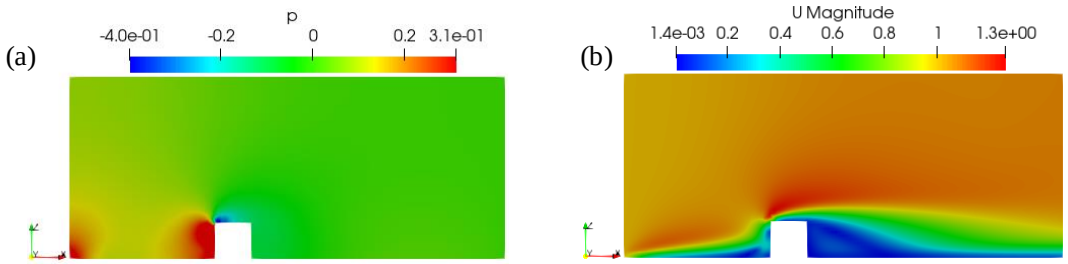


Fig. 9. Contour plots in computational domain through XZ-plane at $y^* = 3.5$ for (a) Pressure (b) Velocity

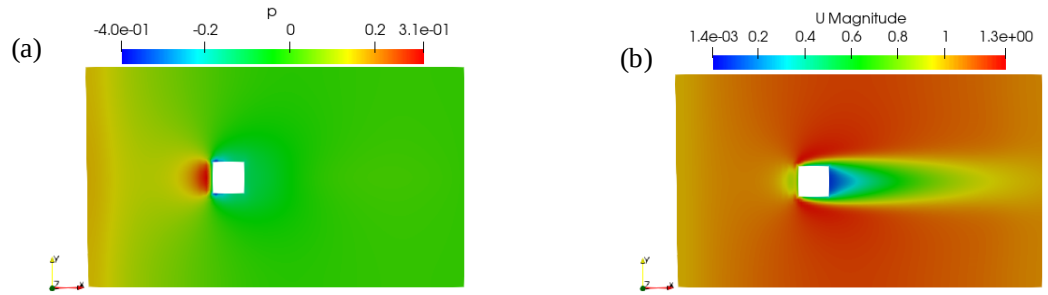


Fig. 10. Contour plots in computational domain through XY-plane at $z^* = 1.0$ for (a) Pressure (b) Velocity

In Fig. 9 (a) and (b), pressure contour and velocity contour plots for wind flow around a cubical building are shown respectively. The red colored patch right in front of building on the windward face in Fig. 9(a) represents the stagnation region formed when the fluid elements are brought to rest upon impacting the building. As the windward wall faces direct impact due to straight line

winds, a positive pressure is encountered on the windward face of the building. Similarly, in Fig. 9 (b), the blue colored patch (close to 0 value on contour scale) nearby the building faces indicates that the fluid elements are brought to rest in the vicinity of building, indicating that no-slip condition is implemented properly on the faces of building.

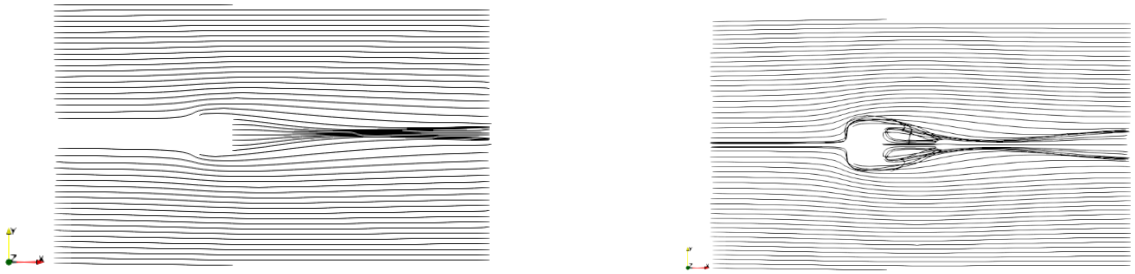


Fig. 11. Streamlines in computational domain through XY-plane at (a) z = 1.0 (b) z = 0.5

Similarly, pressure contour and velocity contour plots in the XY-plane at roof height of building (i.e. $z^* = 1$) are shown respectively in Fig. 10(a) and (b). The pressure contour plot in Fig. 10(a) also indicates the formation of stagnation region (indicated by red color patch) on windward face of the building with high pressure values. Similarly, the blue colored patch formed behind the building in Fig. 10(b) indicates that a low velocity region is created behind the building due to wind flow around building. The streamline plots taken at roof height and half of roof height are shown in Fig. 11(a) and (b) respectively, in which recirculation bubble formed behind the building can be observed from Fig. 11(b). At the leeward face of building, not only a low velocity region is formed but interesting flow phenomena such as wake formation, formation of backflow region, etc., can be noticed.

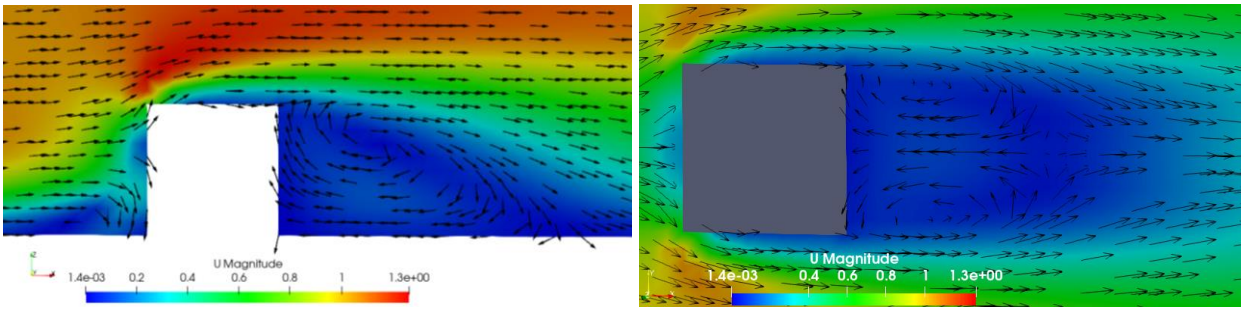


Fig. 12. Wake formation & backflow region formed behind building at (a) XZ-plane (y = 3.5) (b) XY-plane (z = 0.5)

Fig. 12(a) shows elevation view of computational domain when it is intercepted by an XZ section plane at $y^* = 3.5$ whereas that in Fig. 12(b) shows the plan view of computational when intercepted by an XY section plane at $z^* = 0.5$. In Fig. 12(a), it can be observed that the direction of wind flow is generally from left to right in computational domain; however, on the leeward side behind the building, a wake in formed and backflow is set up in the opposite direction, i.e. from right to left in the blue colored patch behind the building. Similarly, in Fig. 12(b), the backflow region created by wind flow is more conspicuously observed by the vectors pointing in leftward direction.

Finally, the profile of pressure coefficient along the centerline of building on the windward face, roof and leeward face of building is extracted and plotted in Fig. 13(b). In order to obtain pressure

coefficient profile along the centerline, the lines formed by joining vertices 0, 1, 2 and 3 (refer Fig. 13(a)) with a total length of 3 units (refer X-axis in Fig. 13(b)) was used and the cutting plane as shown in Fig. 13(a) was taken as XZ plane at $y^* = 3.5$. The pressure coefficient (Cp) was calculated from the solved pressure field using Eqn. (4).

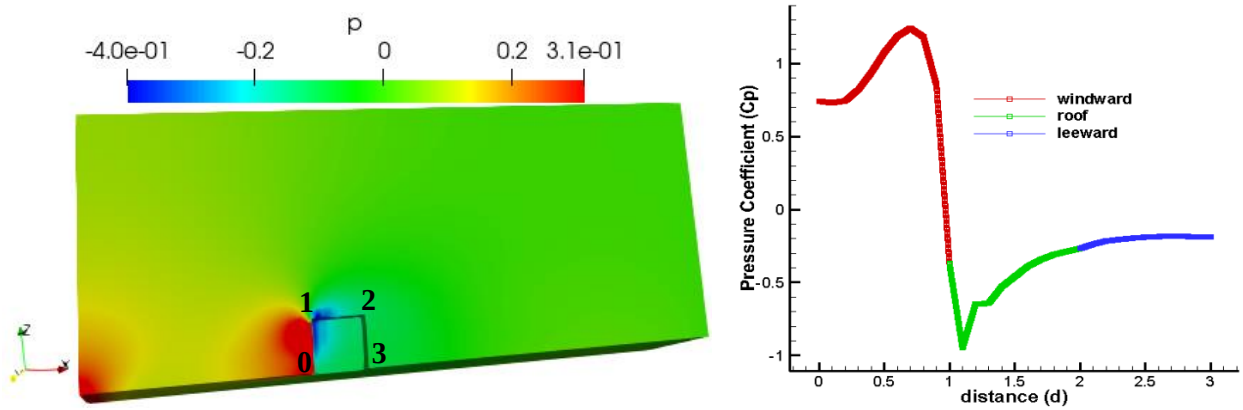


Fig. 13. (a) Clipped surface of the computational domain at XZ-plane ($y^* = 3.5$) for extraction of pressure profile along the centerline of building (b) Pressure coefficient profile along the centerline of building

In Eqn. (4), 'p' is obtained from the solved kinematic pressure field whereas 'p_∞' is the free-stream pressure value taken at the outlet for current problem. Similarly, 'ρ' is the density of fluid and 'V_∞' is the free-stream velocity of flow taken at inlet in the current case. The obtained pressure profile is important for estimation of wind loads due to straight line winds, which in turn is important for design of buildings against wind loads.

$$C_p = \frac{(p - p_\infty)}{{}^1/_2 \, \rho \, V_\infty^2} \tag{4}$$

7. Conclusion

In this contribution, an outline for implementing a two weeks course module by incorporating open source software in the instructional workflow is proposed and demonstrated by considering an example problem of wind flow around a building. With the help of this brief but concise introduction along with the uploaded case files available at [Github link], students and practicing engineers can easily install OpenFOAM and ParaView, make their own simple case files, run simulations and visualize results. Specifically, readers will be able to obtain several visualizations such as slicing of computational domain at different planes, obtaining contour plots (pressure and velocity contour), velocity vector plots and streamline plots for further analysis of the simulated results. Similarly, readers will also be able to obtain pressure profile along the centerline of building, which can be used further in design and analysis of building frames against wind loads.

Acknowledgements

The authors acknowledge the support received from National Science Foundation (NSF) under award number CMMI-1762999. Any opinions, findings, conclusions and/or recommendations from the current work are solely of the authors and do not necessarily reflect the views of NSF.

Biographical information

Sumit Verma received his bachelor's degree in Civil Engineering from Institute of Engineering, Pulchowk Campus, Kathmandu, Nepal in 2015. Then, he worked as a Civil Engineer for a consulting firm called 'G.I. Engineering' before starting his graduate studies from Fall 2018 at the University of Arkansas. Currently, he is in the 3rd year of his Ph.D. program working as graduate research assistant in the Department of Civil Engineering at University of Arkansas.

<< 2nd Author Biographical Data >>

<< 3rd Author Biographical Data >>

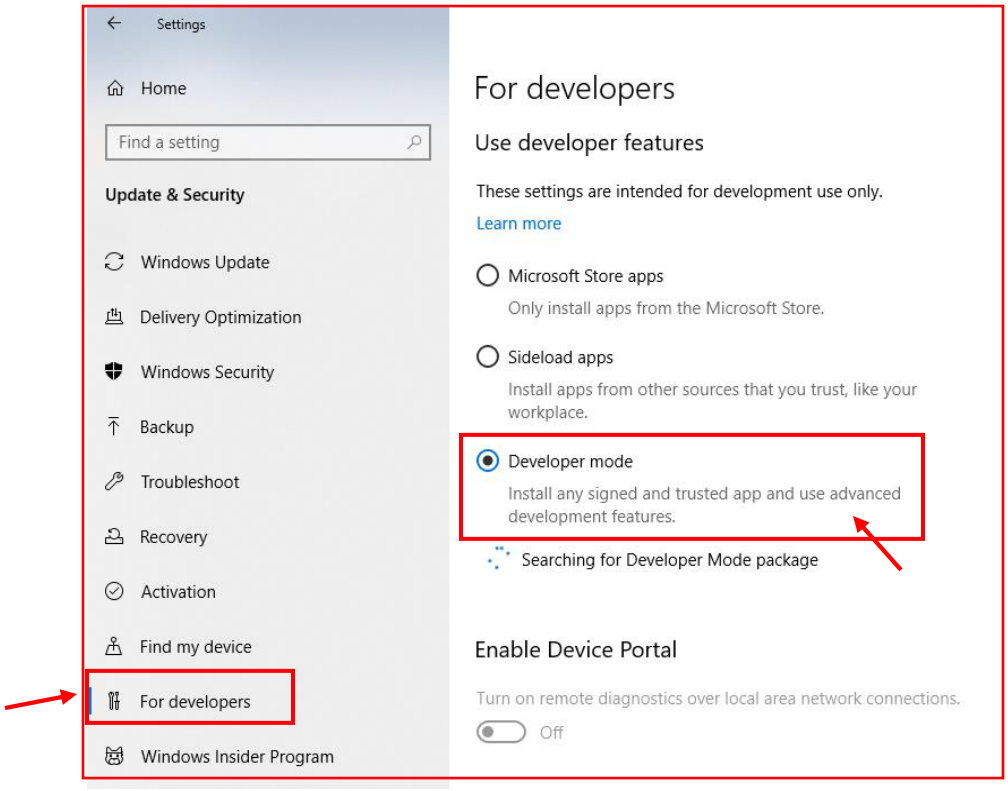
Section –A: OpenFOAM Installation

Installation of OpenFOAM in Windows 10

A detailed procedure of OpenFOAM installation is described in [1]. In the following text, the key steps in the installation process is described with illustrations and the keywords are highlighted in red.

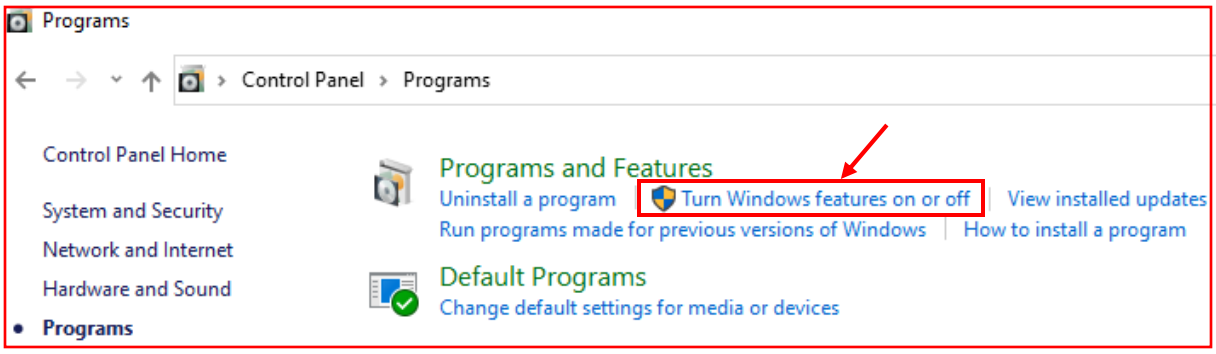
1. Activate "Developers Mode" in Windows 10. For that purpose, type "Settings" in the windows searchbar and press "Enter". A new window pops up, select "Update & Security" from the available options. Once selected, another window pops up from which select "For developers".

2. After selecting "For Developers", select "Developer mode". It will take a while to install developer mode packages.

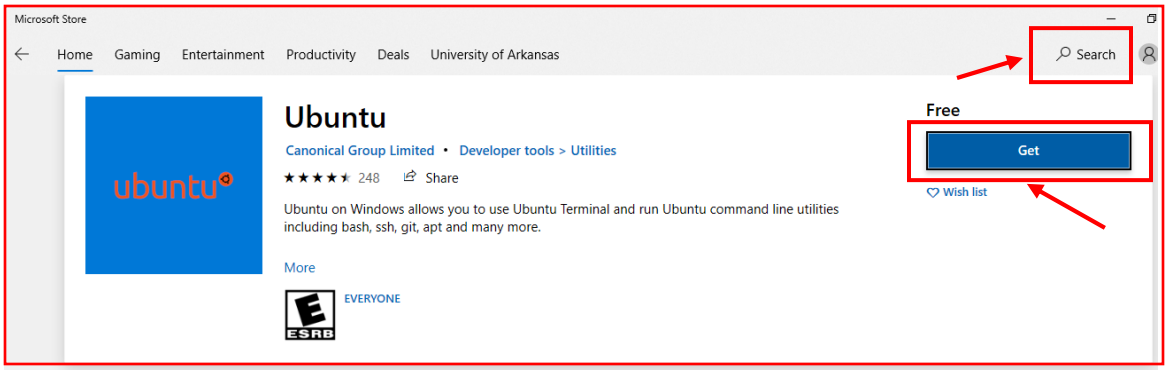


3. Then, go to search bar again and type "Control Panel". A window pops up from which select "Programs". Then, another window pops up, under "Programs and Features", select "Turn Windows Features on or off". A dialog box appear, scroll to the bottom and put a check mark next to "Windows Subsystem for Linux" and then finally select "OK". The PC needs to be restarted for the changes to come into effect, so, restart the PC.

Section –A: OpenFOAM Installation

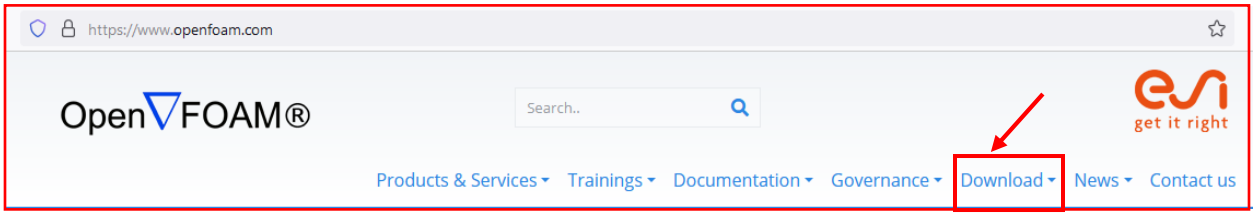


4. Next, we need to install Ubuntu in Windows. For that purpose, Ubuntu must be downloaded first from Microsoft Store. Click on the "Microsoft Store" icon located at the taskbar.



5. Then, type "Ubuntu" in the searchbox as shown above. Then, select "Get". This step will download and install Ubuntu in the PC. Now, open command prompt in windows and type "bash". Users will be prompted to provide "username" and "password" and accordingly, users can provide them as they please.

6. Now, OpenFOAM has to be downloaded in the PC. Go to https://www.openfoam.com/ and download "Windows 10 native (WSL)".



7. Follow the link for installation: https://www.openfoam.com/download/openfoam-installation-on-windows-10

8. Once, "OpenFOAM" is downloaded. Type "Bash on Ubuntu on Windows" in the search bar in Windows and a bash terminal will open up.

9. Type "cd" and press "enter". Now, a directory called "home" will be created where OpenFOAM will be installed. For that purpose, type "cd" and inside the "home" directory, another sub-directory with the username will be created where OpenFOAM will be copied and extracted for further installation.

Section –A: OpenFOAM Installation

10. Type "mkdir" home to create a home directory and then type "cd home". Once, you are inside home directory, create another directory with the username, i.e. mkdir "username". Now, navigate to the Downloads folder in Windows where OpenFOAM was downloaded by typing the following on bash terminal: cd /mnt/c/Users/username/Downloads/ OpenFOAM-v2012-windows10.tgz

11. Open another bash terminal and navigate to the home directory and then inside the directory whose directory name was kept as "username". Using following command, copy the downloaded OpenFOAM .tgz file into the current directory: cp – r /mnt/c/Users/username/Downloads/ OpenFOAM-v2012-windows10.tgz .

12. The step in 11 will copy the OpenFOAM .tgz file into the current working directory. Now type: tar –zxvf OpenFOAM-v2012-windows10.tgz and press enter. It will extract the OpenFOAM files in the current location.

13. A directory will now be created inside home/username directory which will be named as "tutorialsPractice". Then the tutorials folder from OpenFOAM installation will be copied to "tutorialsPractice" for running simulations and visualizing results.

14. Using the following command, copy the tutorials folder inside "tutorialsPractice":
    cd /home/username/tutorialsPractice
    cp ~/home/username/OpenFOAM/OpenFOAM-v2012/tutorials .

15. Using commands from step-14, the tutorials folder from OpenFOAM is copied to tutorialsPractice. Now go to "tutorialsPractice" and run a sample case file. Type:
    cd tutorials/incompressible/icoFoam/cavity

16. Type the following to do a test run: ./Allrun. This command will run the simulation and data files will be stored.

17. Before installing Paraview, copy the directory where simulation was carried out to Desktop for visualization purposes. For that purpose, open another terminal and use the following command: cp ~/home/username/tutorialsPractice/tutorials/incompressible/icoFoam/cavity /mnt/c/Users/username/Desktop

18. Now, Paraview will be installed in the PC. Go to https://www.paraview.org/download/
    And download 64 bit exe file for 64 bit system PC. Once, download run the exe file and with the default settings, proceed ahead with the installation process. Then, Paraview will be installed in the PC and the simulated case folder is also copied to Desktop. So, we can proceed ahead with visualization.

19. Open "Paraview" by typing it in the windows search bar. Paraview window opens up, click on "File" and then select "Open". Navigate to Desktop where the simulated case file exists and from there navigate up to "controlDict" file and select "OpenFOAM Reader". Then, the simulated geometry should be visible on the screen and further post-processing works can be done.

Section-B: PARAVIEW VISUALIZATION

(B) Procedure for obtaining contour plots

1. First, Open "Paraview" and then click on "File" tab at the top left corner of the screen and then select "Open". After, selecting "Open", a dialog box appears; at the bottom of dialog box, select "All files (*)" by clicking at the location pointed by arrow pointer "**1**" in Fig. 1.

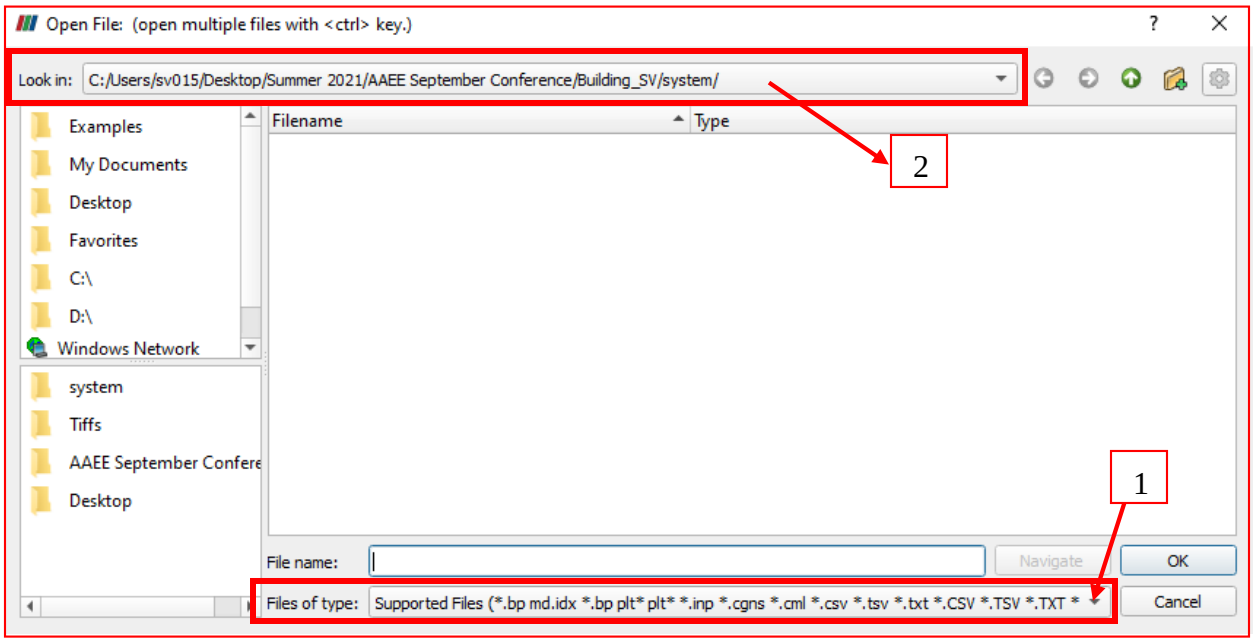


Fig. 1. Demonstrating the path where system folder resides and selecting proper file type for visualization

2. After completion of step-1, navigate to the location where the data file is stored in the PC (Refer arrow pointer "**2**" in Fig. 1). Once, you are inside the folder (where, folder means your case directory), navigate to "system" folder and then select "controlDict" as shown by arrow pointer "**3**". Finally, select "OK" as indicated by arrow pointer "**4**".

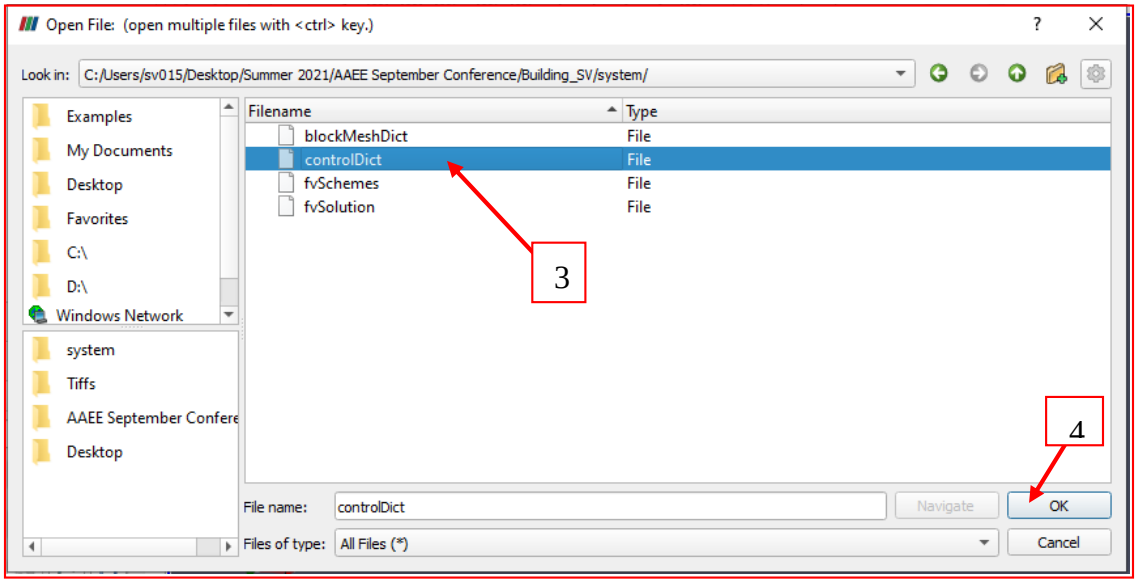


Fig. 2. Demonstrating 'controlDict' file to load for data visualization purposes

Section-B: PARAVIEW VISUALIZATION

3. After completion of step-2, another dialog box appears in which select "OpenFOAMReader" from the list of file reader options available as shown by arrow pointer "**5**". Finally, select "OK" as shown by arrow pointer "**6**". Then, select "Apply" as shown by arrow pointer "**7**" in Fig. 3(b) below.

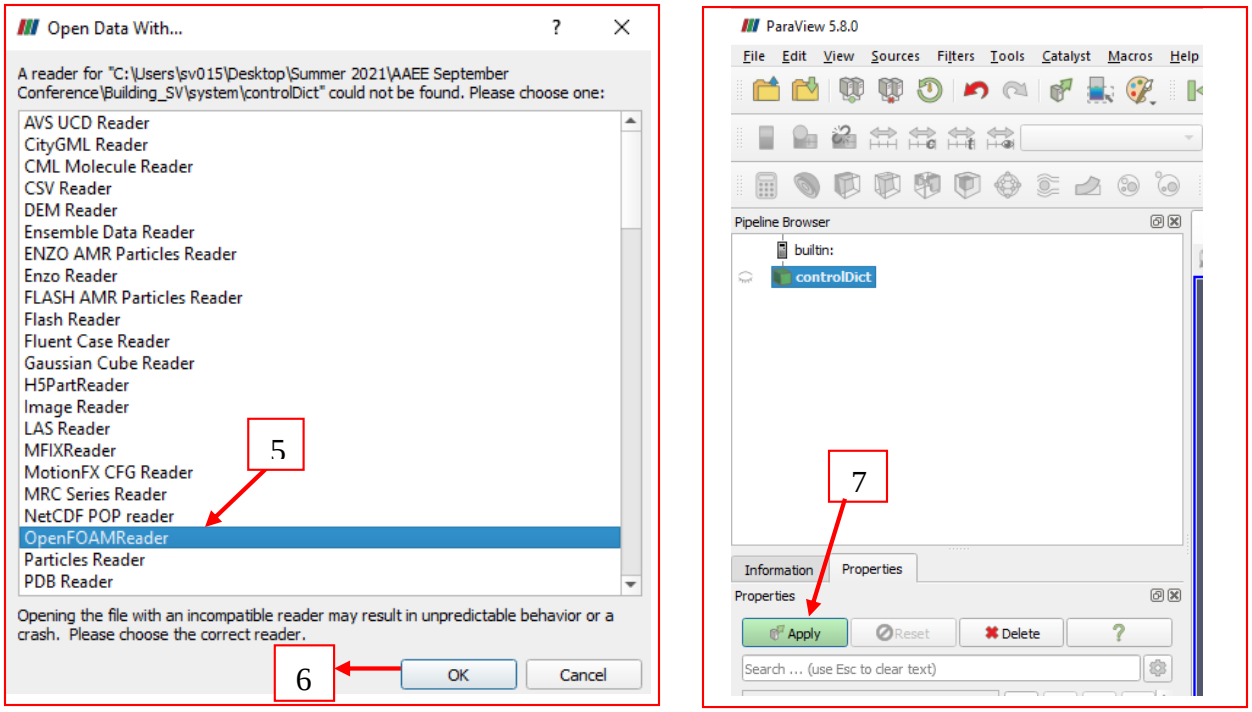


Fig. 3. Demonstrating (a) OpenFOAM data reader for visualization (b) 'Apply' tab to apply changes into effect

4. After completion of step-3, the following screen as shown in Fig. 4 will appear. Users can see the contour plot of 3D computational domain with default coloring scheme as shown in Fig.4. However, for the purposes of this work, the coloring scheme will be changed as follows. Scroll the slider down at the location shown by arrow pointer "**8**" (Refer Fig. 4).

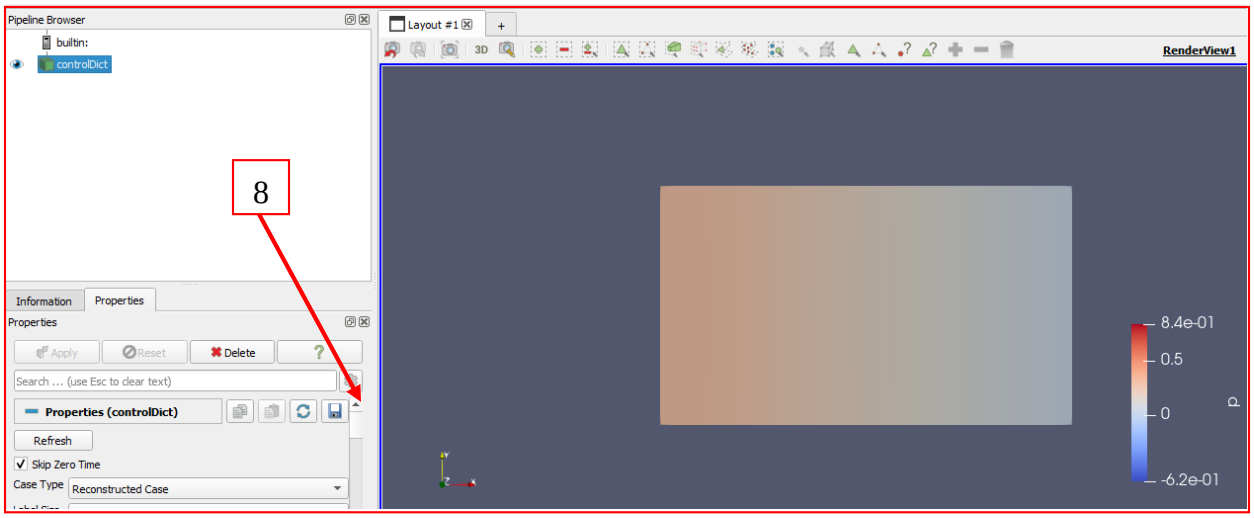


Fig.4. Demonstration of Slider bar in Paraview

Section-B: PARAVIEW VISUALIZATION

5. Once, you have reached the 'Coloring' section as shown by arrow pointer "**9**" in Fig. 5. Select the 'Choose Preset' icon as shown by arrow pointer "**10**" in Fig. 5. After clicking on "choose preset", another dialog box appears as shown in Fig. 6. Click on the field "default" as shown by arrow pointer "**11**" in Fig. 6 and then select "All". From the different available coloring schemes, select "Blue to Red Rainbow" (which was also chosen for the current work).

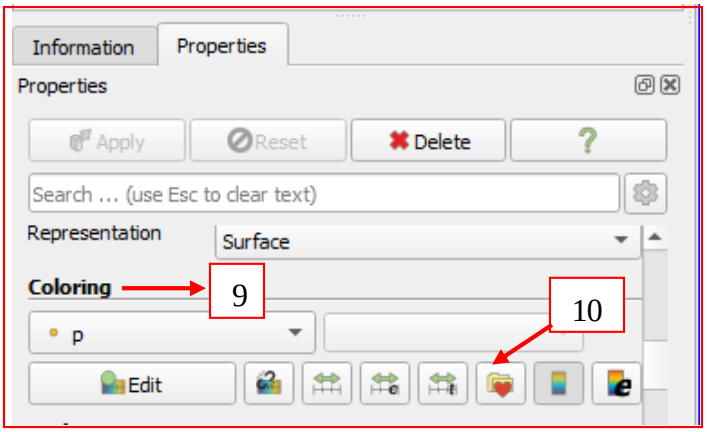


Fig. 5. Demonstration of icon for modifying the contour coloring scheme in 'Paraview'

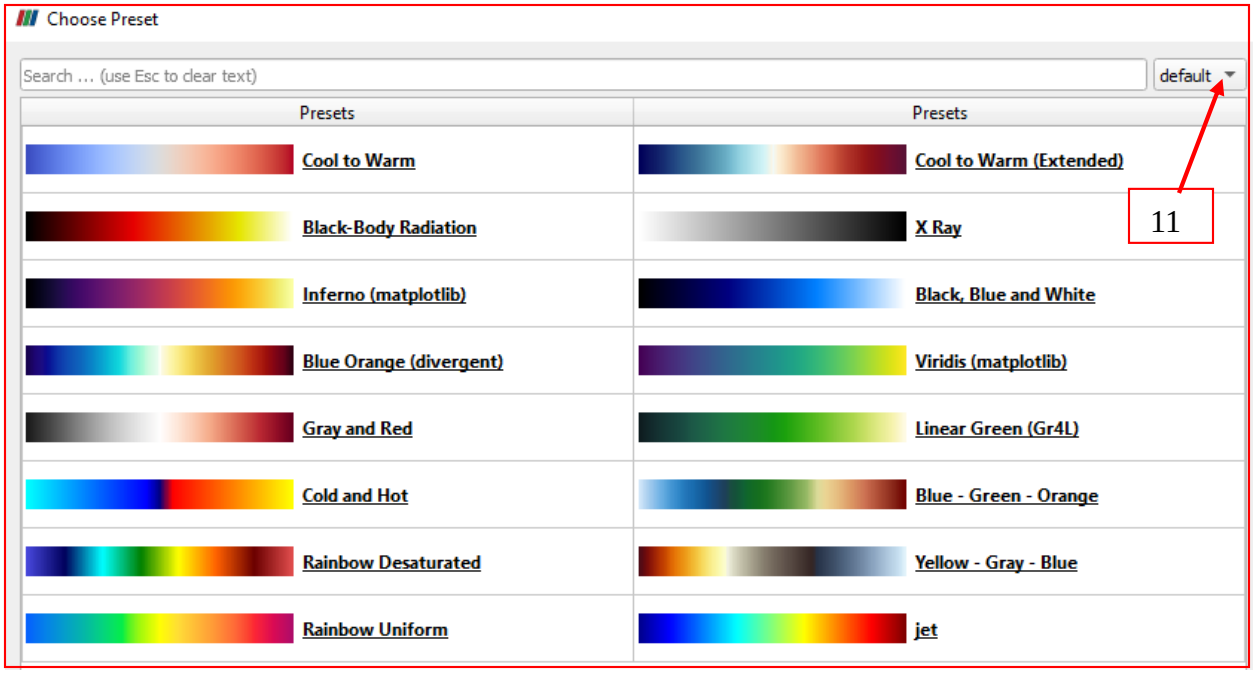


Fig. 6. Demonstration of different coloring schemes available in 'Paraview'

6. After completion of step-5, users can see the 3D computational domain with a changed coloring scheme as shown in Fig. 7.

Section-B: PARAVIEW VISUALIZATION

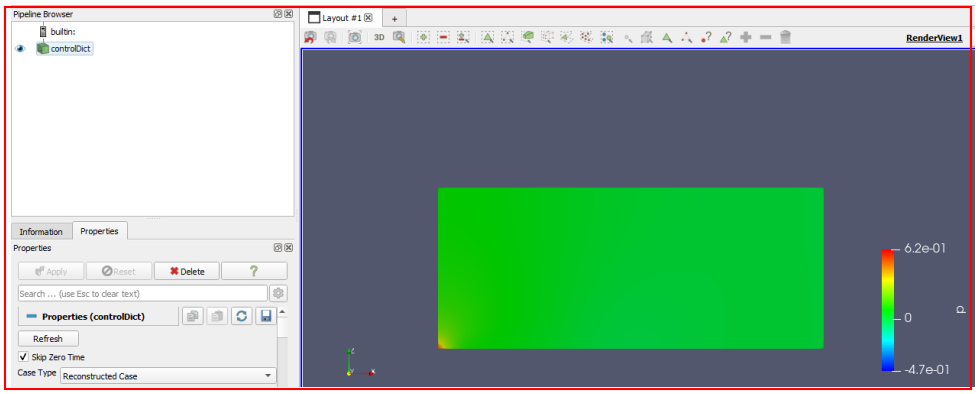


Fig. 7. Demonstration of 3D computational domain after loading 'controlDict' file in Paraview

7. Now, we will try to obtain a vertical section of 3D computational domain through the centerline of the building to show (XZ-plane at y = 3.5 as shown in Fig. 8). For that purpose, at first, the viewing plane will be switched to XZ-plane by clicking on the icon pointed by arrow pointer "**12**".

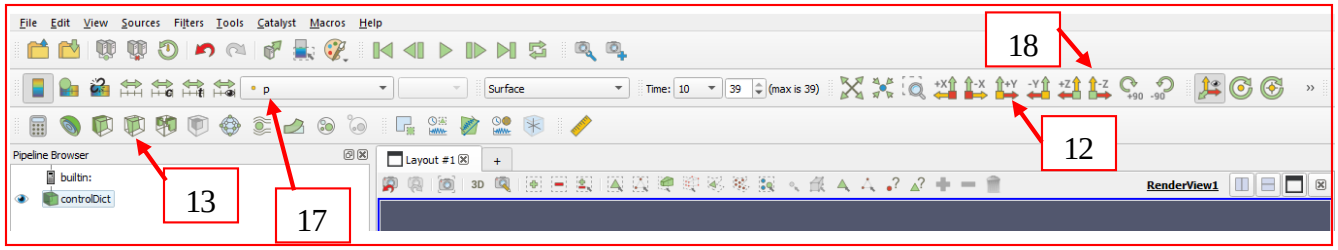
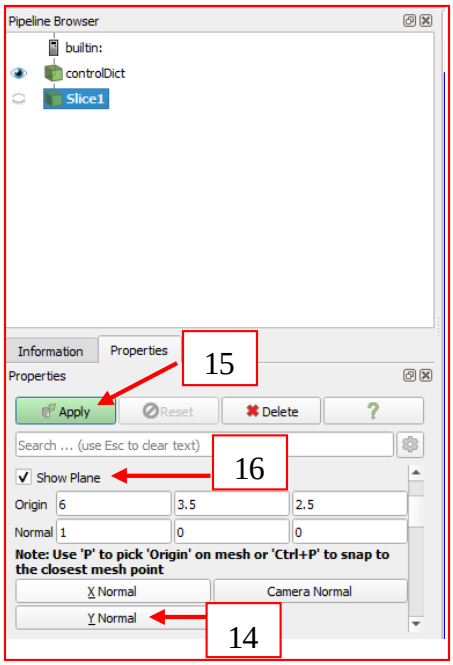
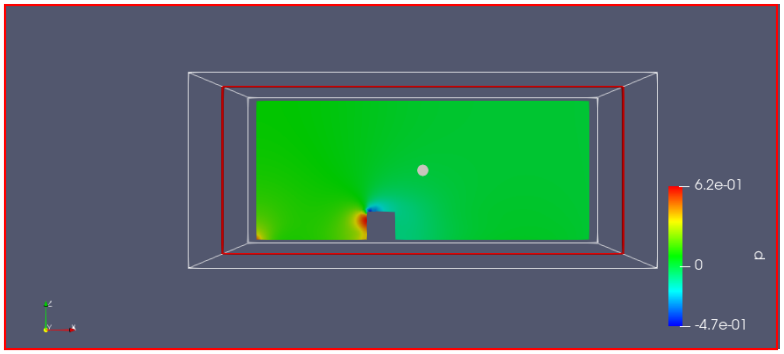


Fig. 8. Demonstration of icons for modifying view planes, obtaining slices and choosing different contours

8. After switching the view plane, click on the "Slice" icon as indicated by arrow pointer "**13**".





(b)

Fig. 9. (a) Demonstration of field for selecting cutting plane for slice (b) sectional view after applying 'slice' filter

Section-B: PARAVIEW VISUALIZATION

As we are targeting to look into XZ-plane at y = 3.5, so, "Y Normal" will be selected as indicated by arrow pointer "**14**" and then hit "Apply" as shown by arrow pointer "**15**". After this, users can see the the screen with a section plane through the centerline of computational domain including the cutting plane (shown by a red rectangle in Fig. 9(b)). If the users want to turn off the cutting plane, they can do so by unchecking "Show Plane" option as shown by arrow pointer "**16**".

9. After following the steps up to 9, users can obtain the contour plot as shown in Fig. 10 (a), which shows the pressure contour plot. If the users wish to view the velocity contour, users have to select "U" instead of "p" at the location of arrow pointer "**17**" in Fig. 8. Following this step, users can obtain a contour plot similar to that shown in Fig. 10 (b).

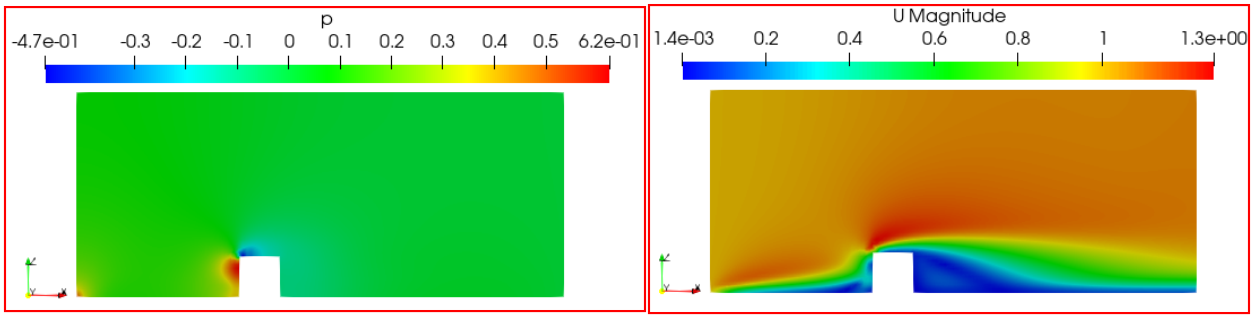


Fig. 10. Contour plots in the computational domain through the XZ-plane at y = 3.5 (a) Pressure contour (b) Velocity contour

10. If users wish to view the contours in XY-plane at z = β , (where β = some height say 1) then, the users should follow the steps from '**7-8**' but they should first switch to "**--Z**" view plane as indicated by arrow pointer "**18**" in Fig. 8 and then select "Z Normal" in Fig. 9(a), which is located just below "Y Normal" of Fig. 9(a) and can be obtained by scrolling the slider slightly down. Similarly, using step-9, the velocity contour plot can be obtained as shown in Fig. 11(b).

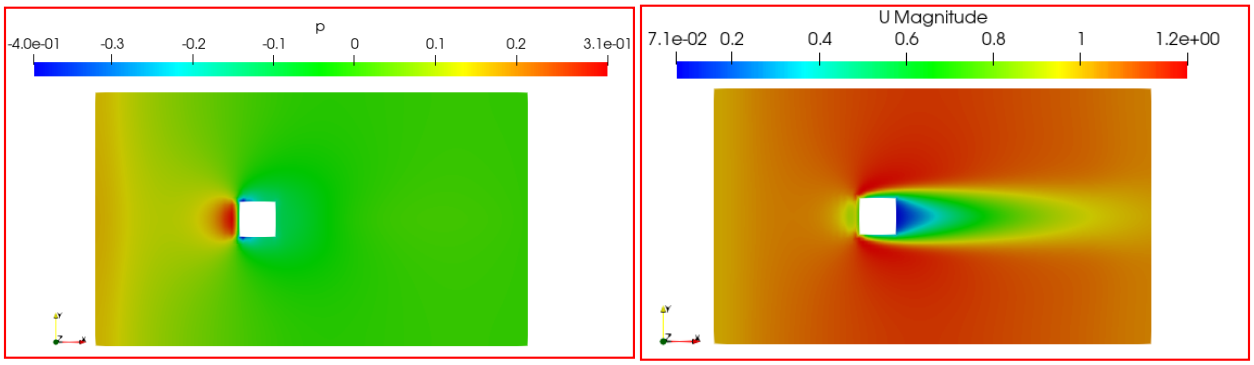


Fig. 11. Contour plots in the computational domain through the XY-plane at z= 1.0 (a) Pressure contour (b) Velocity contour

11. Now, for demonstration purposes, we will try to obtain velocity vector plot for Fig. 10(b) which can be applied to other cases as well such as Fig. 11(b), etc. For that purpose, it is asumed that users have already obtained the slice as shown in Fig. 10 (b) and we will contiue from there.

12. Click on the "**Glyph**" icon as shown by arrow pointer "**19**" in Fig. 12. Then, hit "Apply" as shown by arrow pointer "**20**" in Fig. 13.

Section-B: PARAVIEW VISUALIZATION

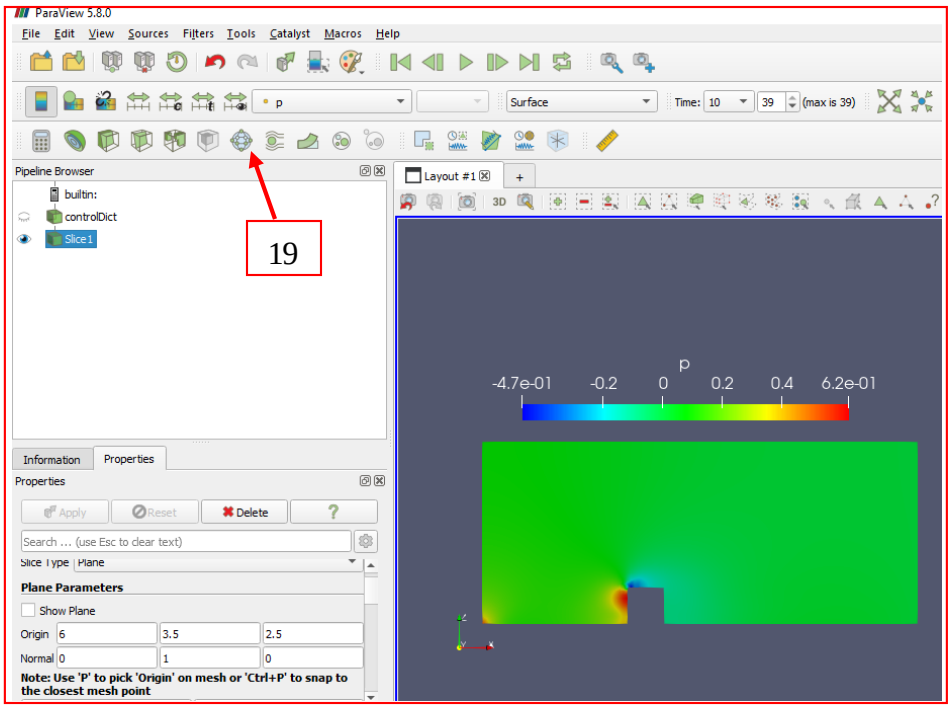


Fig. 12. Demonstration of Glyph icon

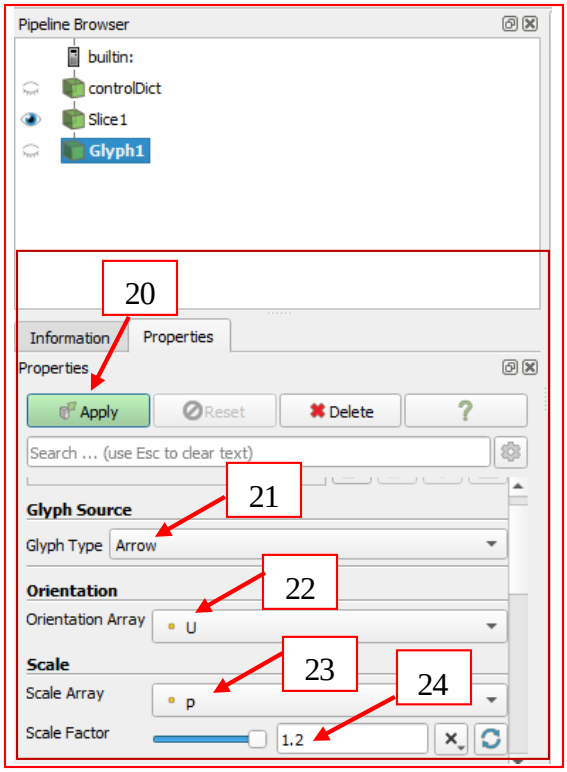


Fig. 13. Demonstration of customized fields for obtaining velocity vector plot

Now, some of the default field values pointed out by the arrow pointers "**21-24**" in Fig. 13 will be customized. "Glyph Type" indicated by arrow pointer "**21**" will be modified to "2D Glyph". The

orientation of velocity vector will be based on the orientation of velocity vector "U", so, "Orientation Array" will be kept as it is (Refer arrow pointer "**22**" in Fig. 13). However, for "Scale Array" (Refer arrow pointer "**23**" in Fig. 13), the field value will be changed to "No scale array" and finally, the value of scale factor (Refer arrow pointer "**24**" in Fig. 13) will be reduced to a smaller value such as "0.1" for the purposes of this work. Following these steps, users can obtain a velocity vector plot as shown in Fig. 14 below.

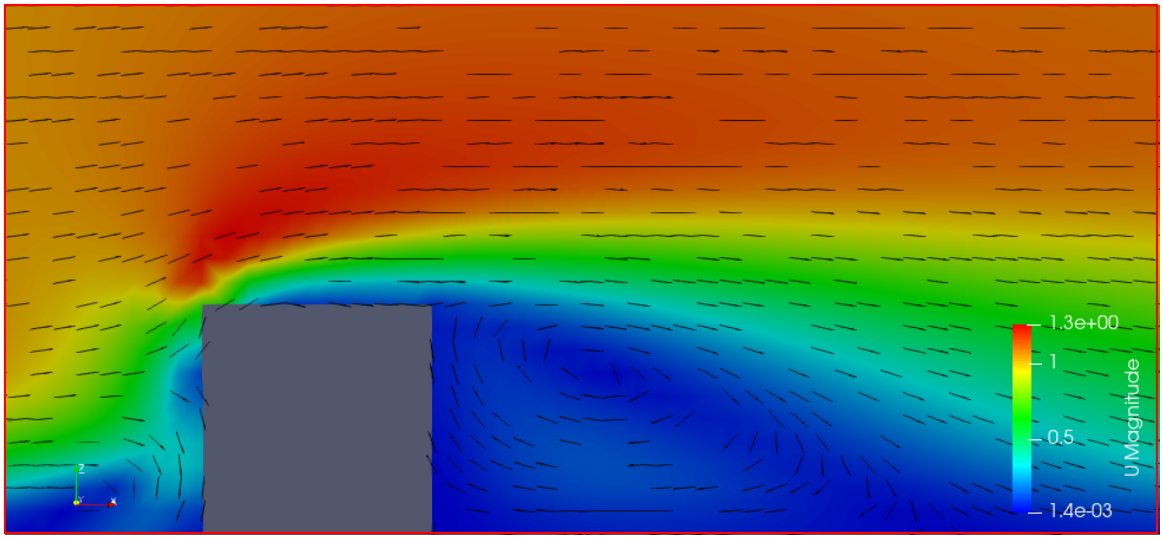


Fig. 14. Velocity vector plot showing wake formation behind the building at XZ-plane (y = 3.5)

13. Now, we will try to obtain streamlines for the plot shown in Fig. 10 (b). For that purpose, select the "controlDict" file shown by arrow pointer "**25**" in Fig. 15. After that, click on the "Stream tracer" icon pointed out by arrow pointer "**26**". Then, hit "Apply" as shown by arrow pointer "**27**" in Fig. 16.

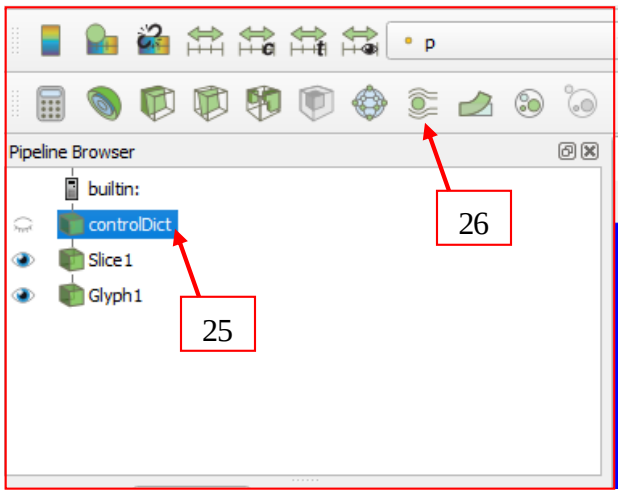


Fig. 15. Demonstration of Streamtracer icon

14. After completing step-13, once again, we will modify some of the default field values pointed out by arrow pointers "**28-30**".

Section-B: PARAVIEW VISUALIZATION

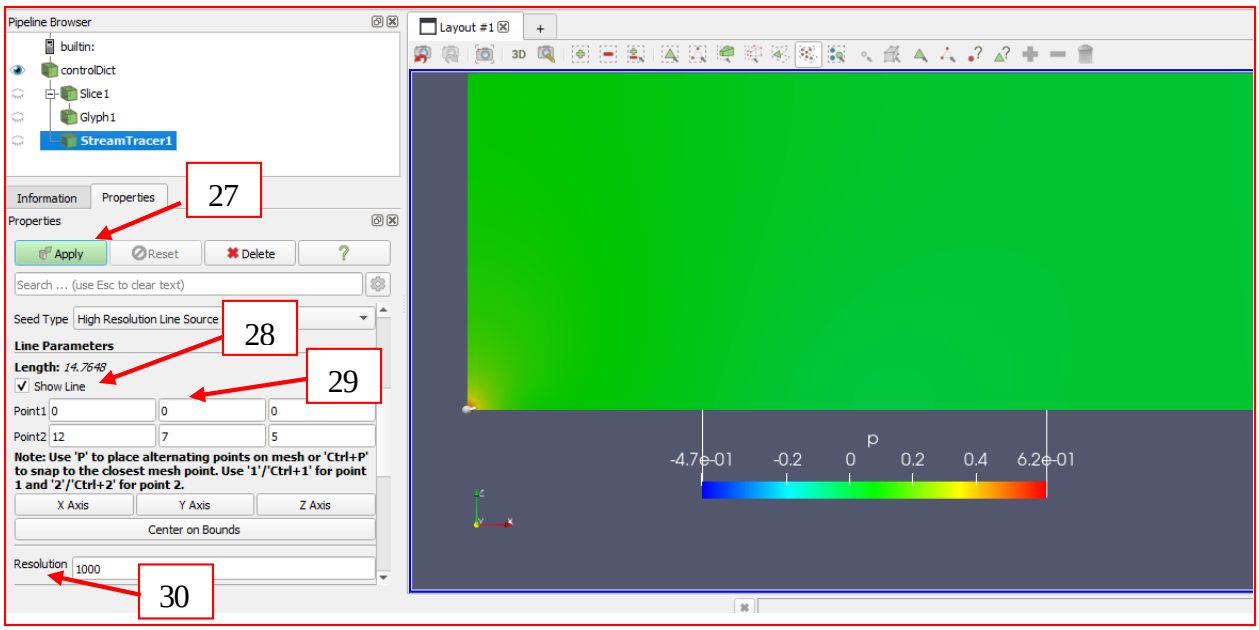


Fig. 16. Demonstration of customized fields for obtaining streamlines

First, we will start from the field pointed out by arrow pointer "**29**", in which we will specify the x, y and z coordinates of a line source for injecting the seed particles (or fluid particles). In our case, we want to obtain streamlines through a section plane that passes through the center line of the building in the computational domain at y = 3.5 and we want t place the line source just touching the windward face of the building. So, the coordinates for Point 1 would be Point1 (x, y, z) = Point1 (4, 3.5, 0) and for Point 2 would be Point2 (x, y, z) = Point2 (4, 3.5, 5). After modifying the coordinates for Point1 and Point2, hit "**Apply**" as indicated by arrow pointer "**27**" in Fig. 16.

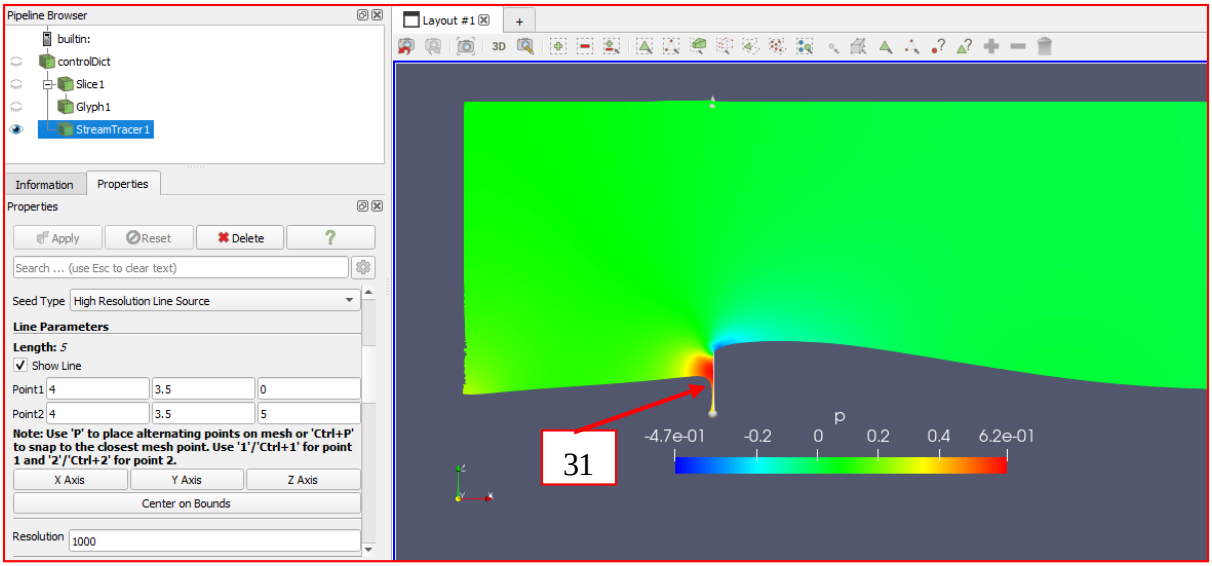


Fig. 17. Streamlines (1000 in number) in the computational domain passing through the XZ-plane at y = 3.5

In Fig.17, arrow pointer "**31**" is used to show the line source used to obtain the streamlines and it can be noticed that the line source falls on the wind ward face of the building. If the user does not

want the line source visible in the Paraview window, they can turn it off by unchecking the "**Show Line**" option pointed out by arrow pointer "**28**".

15. Finally, as we can see in Fig. 17 that we have got a very dense distribution of streamlines, which consists of 1000 streamlines distributed along the length of line source. For our purpose, such a dense distribution is unnecessary, so, we will reduce it to a low value such as "50" for our work. This is done by modifying the value of "Resolution" to "50" at the location pointed out by arrow pointer "**30**". Hence, the final plot for streamlines obtained is shown in Fig. 18.

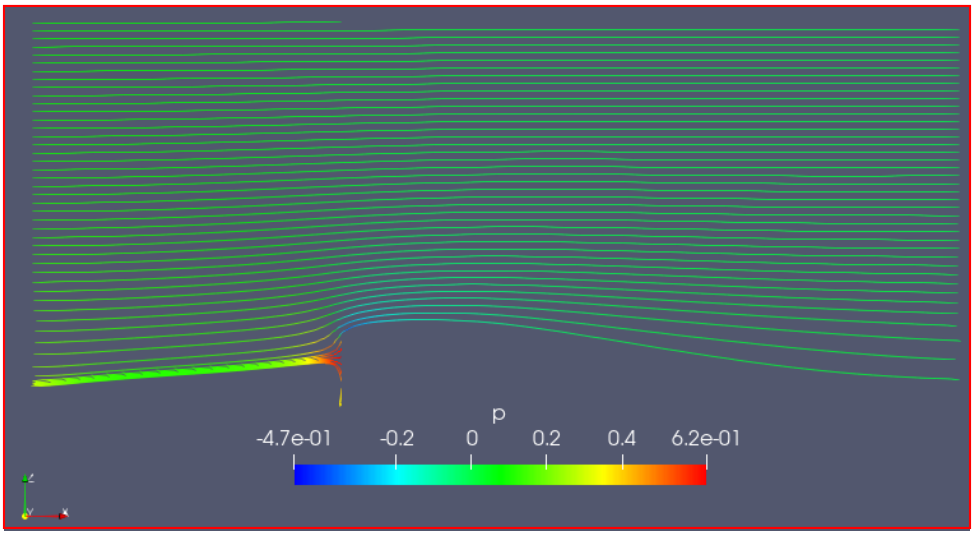


Fig. 18. Streamlines (50 in number) in the computational domain passing through the XZ-plane at y = 3.5

Procedure to obtain Pressure profile along the centerline of building

1. Open "Paraview" and load the 'controlDict' file using steps-1 to 6 from Page 2-4.

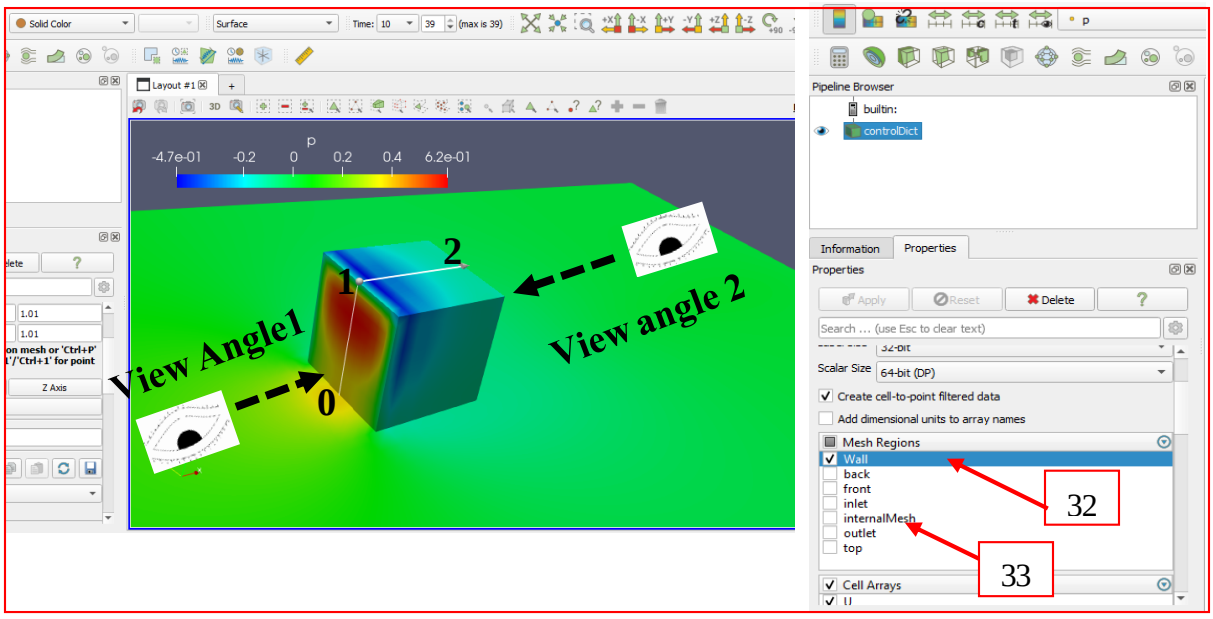


Fig. 19. (a) Different view angles for the building in domain (b) Demonstration for choosing select mesh regions

Section-B: PARAVIEW VISUALIZATION

For the purpose of current work, the "Wall" region of mesh will be turned on while the "internalMesh" will be unchecked as indicated by arrow pointers "**32**" and "**33**". Then, within the Paraview window, the viewing angle for the object (building and bottom wall) can be adjusted using mouse. Following the steps, perspective view of the building model is obtained and in Fig. 19 (a), two different viewing angles are shown. "View Angle 1" covers the windward face and roof of the building model whereas "View Angle 2" covers the roof and the leeward face of the building. As stated in [X], the line 0-1 represents the centerline of building along windward face, line 1-2 represents the centerline of building on roof whereas the line 2-3 represents the centerline on leeward face of the building and is shown in Fig. 20.

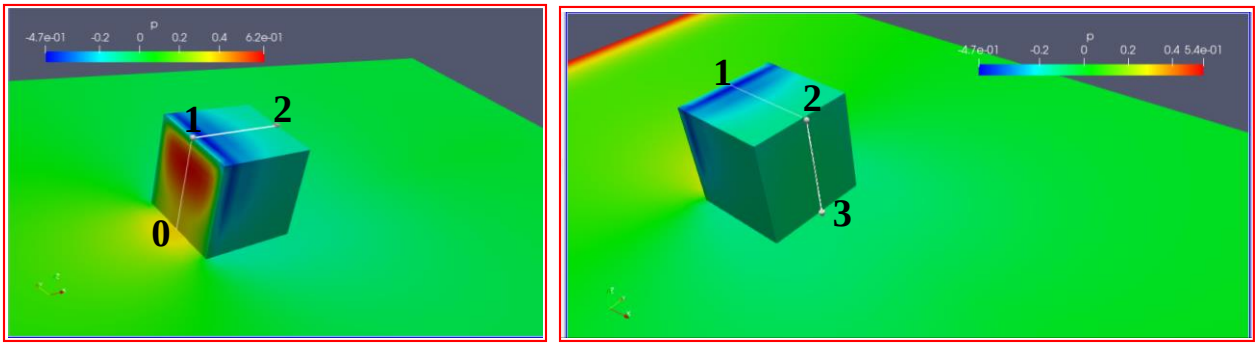


Fig. 20. Building model seen from (a) view angle 1 (b) view angle 2 (Refer Fig. 19 for view angles)

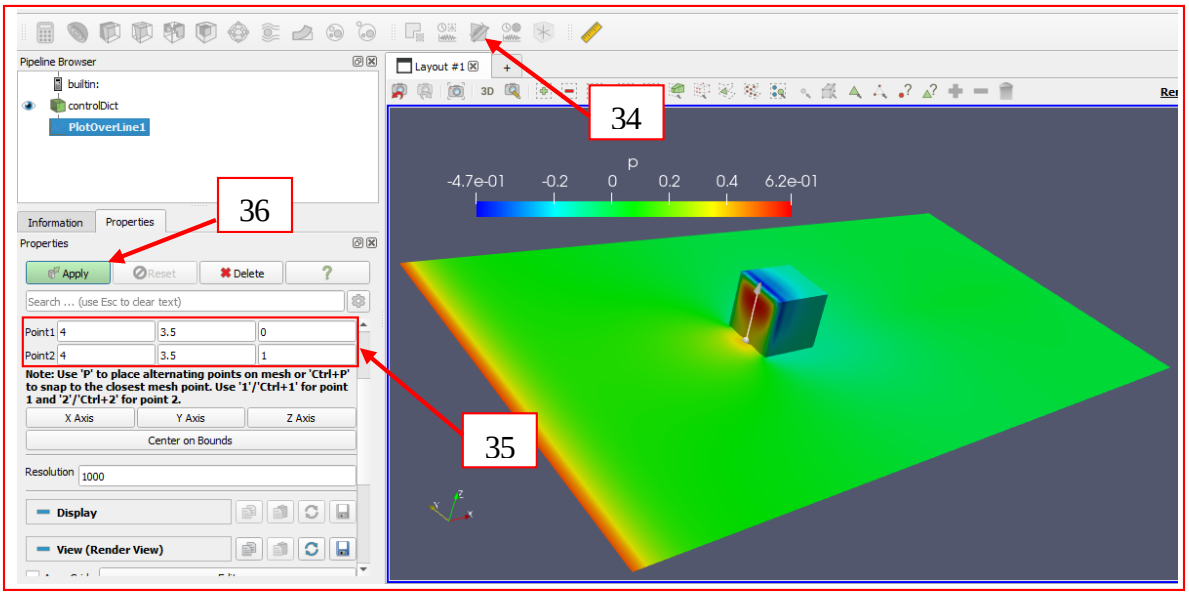


Fig. 21. Demonstration of customized fields for extracting pressure profile data on windward face of building

2. Click on "Plot over Line" icon as pointed out by arrow pointer "**34**" in Fig. 21. Then, specify the coordinates of the end points of the line along which data is to be extracted at the location pointed out by arrow pointer "**35**". For demonstration purposes, in the current case **line 0-1** is taken (**Refer Fig. 20 and 21**). Once, the coordinates are specified, hit "Apply" as pointed by arrow pointer "**36**".

Section-B: PARAVIEW VISUALIZATION

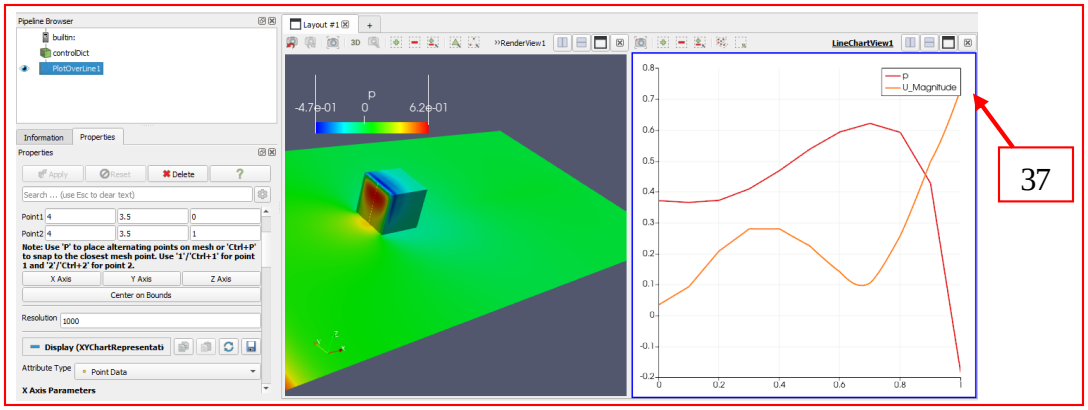


Fig. 22. Demonstrating viewport to save the extracted pressure profile data

3. After completion of step-2, users can obtain the screen shown in Fig. 22. Users should now place their mouse cursor within the viewport as pointed out by arrow pointer "**37**". Users can also notice a blue colored outline around the viewport pointed out by arrow pointer "**37**". This is an indication that the viewport is active now.

4. Now, navigate to the "File" tab (refer arrow pointer '**38**') and then select "Save Data". Then, a new dialog box appears as indicated by arrow pointer "**39**' in Fig. 23.

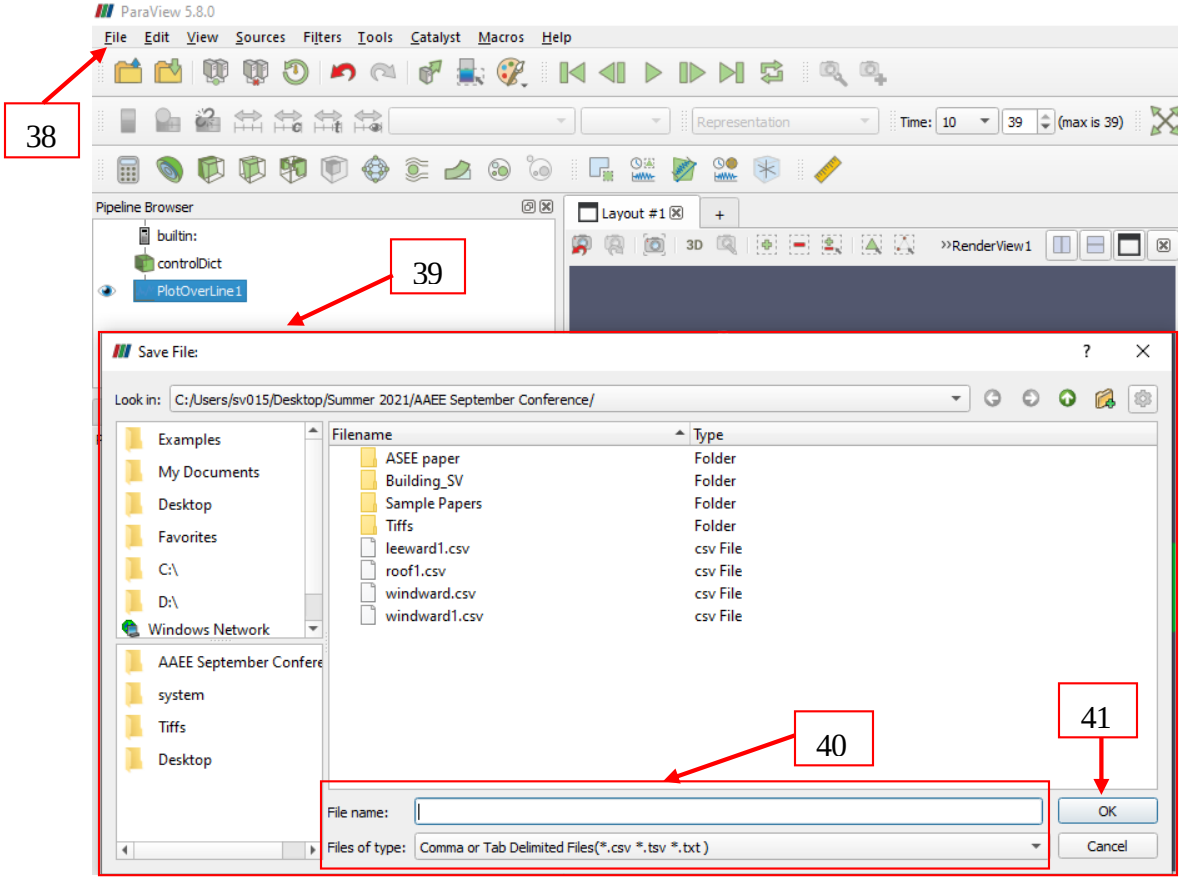
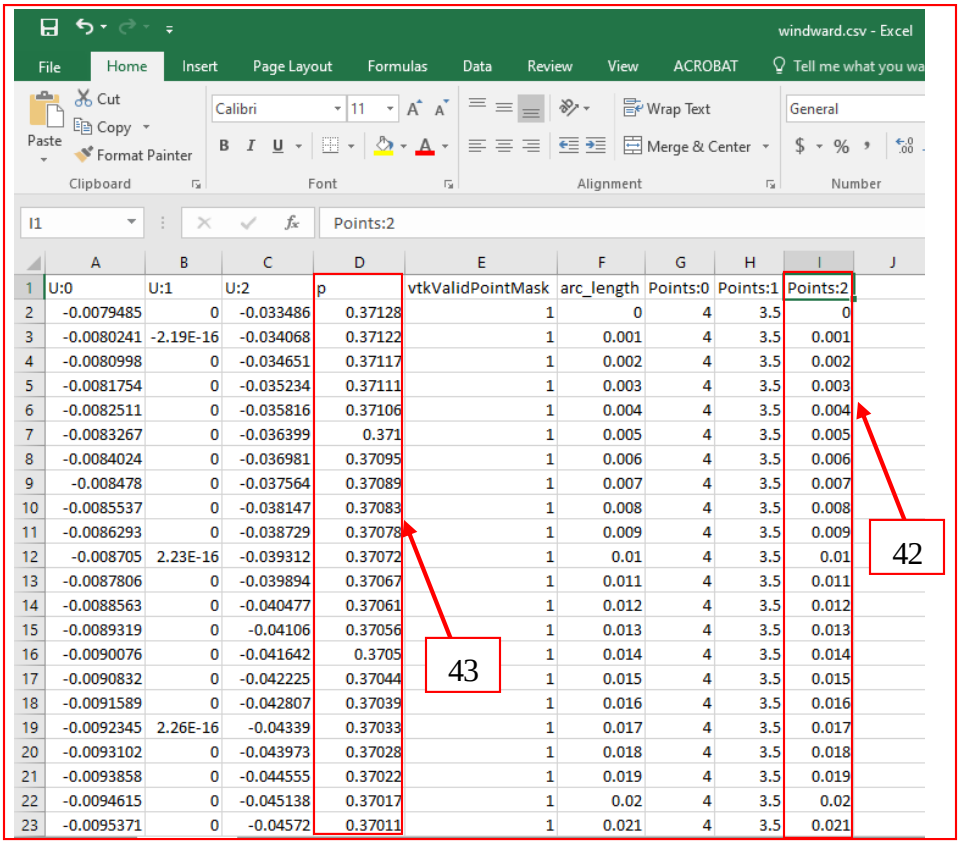


Fig. 23. Demonstrating fields that should be chosen/filled to save extracted pressure profile data

Specify the name of file as you please and then the file format at the location pointed out by arrow pointer "**40**" and then finally hit "OK" as pointed out by arrow pointer "**41**". Also, keep

in mind the desired path where the file is going to be stored and make adjustments for path as you please. Another dialog box appears after selecting "OK" with two options "Cancel" and "OK", simply select "OK". After this, the dialog box closes and the data file is written.

5. Now, the extracted data file is inspected to collect required data fields. As shown in Fig.24, the data fields of interest for us are the ones pointed out by arrow pointers "**42**" and "**43**" (as we are looking forward to obtaining a pressure coefficient profile plot along the centerline of building). The column pointed out by arrow pointer "**42**" contains the distance along the windward face of the building whereas that pointed out by arrow "**43**" contains the corresponding kinematic pressure values. The process described in 1-4 is to obtain pressure profile on the windward face of the building. Users can follow the same process for pressure profile on the roof and on the leeward side as well. Then, users can continue working with excel to merge the data extracted from the windward, roof and leeward face of the building or alternatively use a Matlab script listed in this document.



	A	B	C	D	E	F	G	H	I	J
1	U:0	U:1	U:2	p	vtkValidPointMask	arc_length	Points:0	Points:1	Points:2	
2	-0.0079485	0	-0.033486	0.37128	1	0	4	3.5	0	
3	-0.0080241	-2.19E-16	-0.034068	0.37122	1	0.001	4	3.5	0.001	
4	-0.0080998	0	-0.034651	0.37117	1	0.002	4	3.5	0.002	
5	-0.0081754	0	-0.035234	0.37111	1	0.003	4	3.5	0.003	
6	-0.0082511	0	-0.035816	0.37106	1	0.004	4	3.5	0.004	
7	-0.0083267	0	-0.036399	0.371	1	0.005	4	3.5	0.005	
8	-0.0084024	0	-0.036981	0.37095	1	0.006	4	3.5	0.006	
9	-0.008478	0	-0.037564	0.37089	1	0.007	4	3.5	0.007	
10	-0.0085537	0	-0.038147	0.37083	1	0.008	4	3.5	0.008	
11	-0.0086293	0	-0.038729	0.37078	1	0.009	4	3.5	0.009	
12	-0.008705	2.23E-16	-0.039312	0.37072	1	0.01	4	3.5	0.01	
13	-0.0087806	0	-0.039894	0.37067	1	0.011	4	3.5	0.011	
14	-0.0088563	0	-0.040477	0.37061	1	0.012	4	3.5	0.012	
15	-0.0089319	0	-0.04106	0.37056	1	0.013	4	3.5	0.013	
16	-0.0090076	0	-0.041642	0.3705	1	0.014	4	3.5	0.014	
17	-0.0090832	0	-0.042225	0.37044	1	0.015	4	3.5	0.015	
18	-0.0091589	0	-0.042807	0.37039	1	0.016	4	3.5	0.016	
19	-0.0092345	2.26E-16	-0.04339	0.37033	1	0.017	4	3.5	0.017	
20	-0.0093102	0	-0.043973	0.37028	1	0.018	4	3.5	0.018	
21	-0.0093858	0	-0.044555	0.37022	1	0.019	4	3.5	0.019	
22	-0.0094615	0	-0.045138	0.37017	1	0.02	4	3.5	0.02	
23	-0.0095371	0	-0.04572	0.37011	1	0.021	4	3.5	0.021	

Fig. 24. Demonstrating column data of interest (distance and pressure) to obtain the final pressure profile

For current work, merging of data on windward, roof and leeward side of the building was done using the Matlab script, which produces a Tecplot compatible data file for visualization in Tecplot (another scientific data visualization software).

To obtain the merged dataset using the Matlab script, the filenames should be named "windward_data.csv" for windward dataset, "roof_data.csv" for roof dataset and similarly for leeward dataset and these files should be contained inside the same folder where Matlab script

resides or else users should specify proper paths for the datasets at the location indicated by arrow pointer "**44**". After following the entire procedure, the final pressure coefficient profile along the centerline of the building is shown in Fig.25.

MATLAB Script:

```
%% Matlab script to combine pressure profile plot on building faces

format LONG;

%% Data import

data_w = importdata('windward_data.csv');
data_r = importdata('roof_data.csv');
data_l = importdata('leeward_data.csv');

%% Determine size of array
N_w = length (data_w.data);
N_r = length (data_r.data);
N_l = length (data_l.data);

%% pre-allocating space for the arrays
d = zeros(N_w+N_r+N_l,1);
p = zeros(N_w+N_r+N_l,1);

for i = 1:1:N_w
    d(i,1) =  data_w.data (i,6);
    p(i,1) =  data_w.data (i,4);
end

index_counter_w = i;

for j = 1:1:N_r
    d(index_counter_w+j,1) =  d(index_counter_w,1) + data_r.data (j,6);
    p(index_counter_w+j,1) =  data_r.data (j,4);
end

index_counter_r = index_counter_w + j;

for k = 1:1:N_l
    d(index_counter_r+k,1) =  d(index_counter_r,1) + data_l.data (k,6);
    p(index_counter_r+k,1) =  data_l.data (k,4);
end

plot(d,p);

fid = fopen('combined_pressure_profile.plt','w');

fprintf(fid,'Variables = "d","p"\n');

fprintf(fid,'Zone T = "windward"\n');

for a =1:1:N_w
```

```
    fprintf(fid,'%5.16f\t%5.16f\n',d(a),p(a));
end

fprintf(fid,'Zone T = "roof"\n');

for a =N_w+1:1:N_w+N_r
    fprintf(fid,'%5.16f\t%5.16f\n',d(a),p(a));
end

fprintf(fid,'Zone T = "leeward"\n');

for a = N_w+N_r:1:N_w+N_r+N_l
    fprintf(fid,'%5.16f\t%5.16f\n',d(a),p(a));
end

fclose(fid);

Cp = zeros (length(p),1);
p0 = 0;
v0 = 1.0;

for i = 1:1:length(Cp)
    Cp(i,1) = (2.*(p(i,1)-p0))/(v0^2);
end

fid2 = fopen('combined_Cp_profile.plt','w');

fprintf(fid2,'Variables = "d","Cp"\n');

a = 0;

fprintf(fid2,'Zone T = "windward"\n');

for a =1:1:N_w
    fprintf(fid2,'%5.16f\t%5.16f\n',d(a),Cp(a));
end

fprintf(fid2,'Zone T = "roof"\n');

for a =N_w+1:1:N_w+N_r
    fprintf(fid2,'%5.16f\t%5.16f\n',d(a),Cp(a));
end

fprintf(fid2,'Zone T = "leeward"\n');

for a = N_w+N_r:1:N_w+N_r+N_l
    fprintf(fid2,'%5.16f\t%5.16f\n',d(a),Cp(a));
end

fclose(fid2);
```

Section-B: PARAVIEW VISUALIZATION

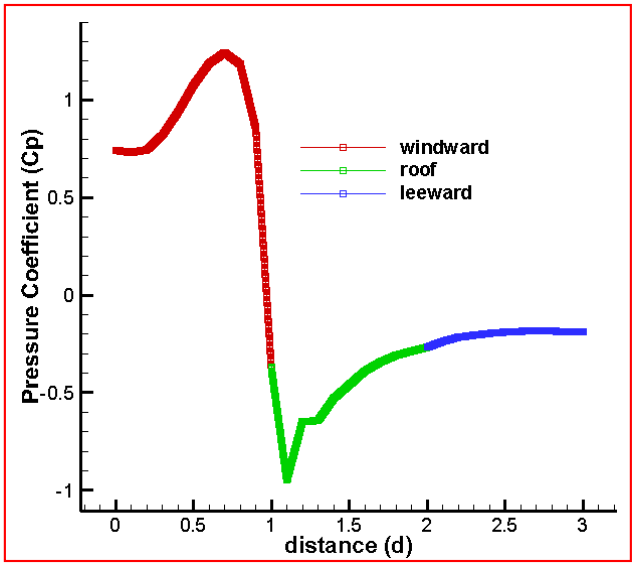


Fig. 25. Pressure coefficient profile along the centerline of the building

References

[1] FREE Cornell University Course Teaching Engineering Simulations, ansys.com https://www.ansys.com/blog/engineering-simulations-course

[2] Z. Mansouri, S. Verma and R.P. Selvam, "Teaching modeling turbulent flow around building using LES turbulence method and open-source software OpenFOAM" in Proc. 2021 ASEE Midwest Section Virtual Conference, Sep. 13-15, 2021.

[3] C.J. Greenshields, "OpenFOAM User Guide version 8, The OpenFOAM Foundation". foam.sourceForge.net. http://foam.sourceforge.net/docs/Guides-a4/OpenFOAMUserGuide-A4.pdf

[4] C.J. Greenshields, "OpenFOAM Programmer's Guide, The Open Source CFD Toolbox". foam.sourceForge.net. http://foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf

[5] OpenFOAM v8 The OpenFOAM Foundation C++ Source Code Guide, cpp.openfoam.org https://cpp.openfoam.org/v8/

[6] Welcome to ParaView Documentation!, paraview.org https://docs.paraview.org/en/latest/

[7] Jozsef Nagy. How to install OpenFOAM and run a simulation in Windows 10-tutorial. (Aug. 5, 2017). Accessed: Jun. 19, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=xj0dB_PsElg

[8] ProgrammingKnowledge2. How to Install Ubuntu on Windows 10 (WSL). (Jun 8, 2020). Accessed: Jun. 19, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=X-DHaQLrBi8

[9] OpenFOAM Installation on Windows 10. openfoam.com https://www.openfoam.com/download/openfoam-installation-on-windows-10

[10] Get the Software. paraview.org https://www.paraview.org/download/

[11] File structure of OpenFOAM cases. openfoam.com https://www.openfoam.com/documentation/user-guide/2-openfoam-cases/2.1-file-structure-of-openfoam-cases

[12] OpenFOAM v6 User Guide: 5.3 Mesh generation with blockMesh, cfd,direct https://cfd.direct/openfoam/user-guide/v6-blockmesh/

[13] H.K. Versteeg and W. Malalasekera, *The finite volume method for unsteady flows* in An Introduction to Computational Fluid Dynamics – the finite volume method, 2nd ed. Essex CM20 2JE, England: Pearson Education Limited, 2007.

[14] J.H. Fergizer and M. Peric, *Finite Volume Methods* in Computational Methods for Fluid Dynamics, 3rd ed. New York, USA: Springer-Verlag Berlin Heidelberg, 2002.

[15] Y.A. Cengel and J.M. Cimbala, *Approximate solutions of the Navier-Stokes equation* in Fluid Mechanics Fundamentals and Applications, 3rd ed. New York, NY 10020, USA: McGraw-Hill, 2014.

[16] Standard solvers. openfoam.com https://www.openfoam.com/documentation/user-guide/a-reference/a.1-standard-solvers

[17] Incompressible flow solver: IcoFoam, openfoamwiki.net https://openfoamwiki.net/index.php/IcoFoam

[18] OpenFOAM: User Guide v2012, openfoam.com https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-incompressible-icoFoam.html