

AEROSOL TRANSMISSION OF COVID-19 AND OTHER AIRBORNE  
DISEASES IN OFFICE ENVIRONMENTS USING COMPUTATIONAL FLUID  
DYNAMIC MODELING AND MACHINE LEARNING

BY

KISHON WEBB

A THESIS SUBMITTED TO

THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE

GRADUATE PROGRAM IN THE DEPARTMENT OF MECHANICAL  
ENGINEERING

YORK UNIVERSITY

TORONTO, ONTARIO

August 2023

© Kishon Webb 2023

---

## Abstract

The COVID-19 pandemic has shown the world how quickly airborne diseases can spread and the lasting impact they can have. Computational fluid dynamic (CFD) models and simulations and machine learning (ML) are powerful tools that allow engineers to create models to predict and advance tools to fight these airborne diseases. The research in this thesis studied the effects of heating, air conditioning and ventilation (HVAC) strategies in small office spaces. A novel methodology was developed to utilize ML, CFD and parallel computing by utilizing the user defined function (UDF) tool of ANSYS Fluent. It was shown that the resulting risk models were quick and effective at predicting high risk areas using spatial data or predicting regions of high risk over time. Future research will refine this method by creating higher fidelity ML models and investigating a wider range of input and output parameters.

## Acknowledgements

I would like to thank my supervisor Prof. Marina Freire-Gormaly, for giving me this unique opportunity and introducing me to the world of Academia. I never thought I would do my master's and many times I was very close to giving up and not finishing. She has encouraged me and been patient, understanding and has vouched for me many times and for that I am grateful.

I also want to thank all the people from the MFG lab, both past and present, who helped make the MFG lab the place it is and will continue to be. Specifically, I would like to thank my brother and RAY student Keandre Webb and my friend and colleague Harman Nagra for their help and support during my research. Additionally, I'd like to thank, Brandon Truong, Liam Horrigan, Belen Barrios, Abu Raihan Ibna Ali, Thippathong Piluk, and Amanda Capacchione for their companionship during my master's journey.

A special thanks to Arma Khan, which without whose guidance would make this thesis not possible. She was always there to help provide resources and explain concepts relating to CFD and aerosol transmission. Her knowledge was fundamental to the writing of this thesis.

The financial support from the Natural Sciences and Engineering Research Council (NSERC) of Canada for the NSERC CGS-M Scholarship was invaluable.

Of course, I want to thank the ones who raised me, my loving parents and siblings who instilled the love of God and hard work as principles that I live by today.

Thank you all!

# Table of Contents

Abstract .....	ii
Acknowledgements .....	iii
Table of Contents .....	iv
Table of Figures .....	viii
Table of Tables .....	x
List of Parameters .....	xi
List of Abbreviations .....	xii
Chapter 1. Introduction .....	1
1.1. Background .....	1
1.2. Motivation .....	3
1.3. Research Objectives .....	4
1.4. Thesis Outline .....	4
Chapter 2. Literature Review .....	5
2.1. Transmission of COVID-19 .....	5
2.2. HVAC and CFD .....	7
2.2.1. CFD Study by Khan .....	8
2.3. Machine Learning .....	10
2.3.1. Introduction to ML .....	10
2.3.2. ML and Neural Networks .....	10
2.3.3. Running and Evaluating ML Models .....	17
2.4. Machine Learning and Computational Fluid Dynamics .....	18
2.4.1. Study by Mirzaei et al. ....	18
2.4.2. Study by Mesgarpour et al. ....	21
2.4.3. Summary .....	23

Chapter 3. Further Utilization of CFD to Investigate Effects of Ventilation Strategies on Spread of Airborne Diseases in Small Office Spaces .....	24
3.1. Introduction .....	24
3.2. Methodology .....	25
3.2.1. Model Geometry .....	25
3.2.2. Meshing Design .....	26
3.3. Numerical Model.....	32
3.3.1. Numerical Solver .....	32
3.3.2. Boundary and Initial Conditions.....	33
3.3.3. Injection Conditions .....	33
3.3.4. Performance Indicators.....	34
3.4. Sensitivity Studies .....	36
3.4.1. Mesh Sensitivity Study .....	37
3.4.2. Numerical Study .....	38
3.5. Results .....	41
3.5.1. Mixing - Velocity and Temperature .....	41
3.5.2. Particle Dispersion.....	48
3.5.3. Performance Metrics.....	50
3.6. Discussion .....	51
3.6.1. Particle Dispersion.....	51
3.6.2. Performance Indicators.....	53
3.7. Conclusions .....	54
3.7.1. Summary.....	54
3.7.2. Final Thoughts.....	55
3.8. Future Work .....	56

Chapter 4. Development of Novel Method to Aid in Particle Dispersion Analysis using Parallel Processing and HPC.....	57
4.1. Introduction .....	57
4.2. Architecture Methodology .....	58
4.3. Getting Started with Parallel Computing .....	59
4.4. Setup.....	60
4.4.1. HPC Initialization and Utilization .....	60
4.4.2. Loading ANSYS and UDF .....	61
4.5. Data Processing .....	62
4.5.1. Initialization.....	63
4.5.2. Data Input/Output .....	63
4.5.3. Node Data Transfer .....	63
4.5.4. File Saving.....	64
4.5.5. Data Storage .....	65
4.6. Results and Discussion.....	66
4.7. Conclusions .....	69
4.7.1. Summary.....	69
4.7.2. Future Work.....	70
Chapter 5. Development of Novel Methods for Predicting Temporal-Spatial Particle Dispersion using CFD and ML .....	71
5.1. Introduction .....	71
5.2. Methodology .....	72
5.2.1. Data Extraction .....	72
5.2.2. Data Processing .....	74
5.2.3. Risk Models.....	76
5.3. Results and Discussion.....	79

5.3.1. Spatial-Temporal Risk Model .....	79
5.3.2. Regional-Temporal Risk Model .....	82
5.3.3. Summary of the ML Results.....	84
5.4. Conclusions .....	85
5.4.1. Summary.....	85
5.4.2. Future Work.....	85
Chapter 6. Conclusions .....	87
6.1. Summary .....	87
6.2. Contributions.....	87
6.2.1. Comparative Study of Ventilation Strategies .....	87
6.2.2. Advanced Combined Use of UDF,CFD and ML for Airborne Transmission.....	87
6.2.3. Resources for Researchers to use HPC and CFD .....	88
6.2.4. Development of Risk and Predictive Models .....	88
6.3. Recommendations for Future Work.....	89
6.3.1. Comparative Study of Ventilation Strategies .....	89
6.3.2. Advanced use of UDF and CFD.....	89
6.3.3. Resources for Researchers to use HPC and CFD .....	89
6.3.4. Expanding Input and Parameters of ML.....	90
References.....	91
Appendix A – Bash code to open Fluent Cas File .....	105
Appendix B – Bash Code to Open Workbench Project File.....	107
Appendix C – UDF Code.....	108

## Table of Figures

Figure 2-1: Comparison of Viral Sedimentation for 60% and 40% RH [62] .....	6
Figure 2-2: ML Pipelines for Application in Engineering [83,84] .....	11
Figure 2-3: Machine Learning Techniques [83] .....	11
Figure 2-4: General Flow Diagram for Neuron in a Neural Network [86].....	12
Figure 2-5: Process of Creating a Reduced Order Model [86] .....	12
Figure 2-6: 2D Concentration plots over time for Mirzaei et al. Study [99] .....	19
Figure 2-7: Graphic of Methodology for Mirzaei et al. Study [99] .....	19
Figure 2-8: Dimensions Results from Mirzaei et al.'s study [99] .....	20
Figure 2-9: ANN Vertical Test Results from Mirzaei et al.'s study [99] .....	20
Figure 2-10: ANN Validation Results for Mirzaei et al. Study [99] .....	21
Figure 2-11: Bus Geometry from Mesgarpour et al.'s study [67] .....	22
Figure 3-1: Previous Airborne Disease CFD Study by Khan [11] .....	25
Figure 3-2: Domain and Boundaries of Berg Office Model .....	25
Figure 3-3: Element Quality for Original Mesh– Human Model .....	28
Figure 3-4: Cell Orthogonal Quality for Original Meshing .....	29
Figure 3-5: Mesh with Refinement Features Replaced Face Sizing .....	30
Figure 3-6: Location of Temperature Probes .....	37
Figure 3-7: Steady State Temperature vs Total Cell Number for Temperature Probes .....	38
Figure 3-8: Plot of Residual Values for Fine Mesh (400,000 cells) .....	39
Figure 3-9: Plot of Temperature Probes for First Order Turbulent Term Approximations .....	39
Figure 3-10: : Plot of Velocity Probes for Second Order Turbulent Term Approximations .....	40
Figure 3-11: Mixing – Right-side View .....	42
Figure 3-12: Mixing - Isometric View .....	42
Figure 3-13: Mixing – Top View .....	43
Figure 3-14: Mixing – Temperature Right view .....	43
Figure 3-15: Displacement Ventilation – Right View .....	44
Figure 3-16: Displacement Ventilation – Top View .....	45
Figure 3-17: Displacement Ventilation – Isometric View .....	45
Figure 3-18: Displacement Ventilation – Temperature Profile .....	46
Figure 3-19: Stratum, Isometric View .....	46

Figure 3-20: Stratum – Top View.....	47
Figure 3-21: Stratum Ventilation Strategy, Right View .....	47
Figure 3-22: Stratum – Temperature Profile .....	48
Figure 3-23: Displacement Ventilation - Particle Advancement over 10s particle time .....	49
Figure 3-24: Stratum - Displacement - Particle Advancement over 10 s particle time .....	50
Figure 4-1: CBPT Overall Process Diagram .....	59
Figure 4-2: Node Architecture for ANSYS Parallel Computations [104] .....	60
Figure 4-3: Setup - Flow Diagram .....	62
Figure 4-4: Partitioned Mesh Distributed Between Two Compute Nodes [104].....	63
Figure 4-5: 2D to 1D Array Encoding .....	64
Figure 4-6: GRAHAM Cluster Showcase – Folder Overview .....	66
Figure 4-7: GRAHAM Cluster Showcase – UDM DATA Folder .....	66
Figure 4-8: GRAHAM Cluster Showcase – DATA Files .....	67
Figure 4-9: Progression of Domain WPC at Particle Time at: a) 1 second, b) 2 seconds, c) 3 seconds, d) 5 seconds, e) 8 seconds, f) 10 seconds.....	67
Figure 4-10: Spatial-Temporal Risk Model for Particle Time at: a) 0.1s, b) 0.2s, c) 0.5s d) 1s... 68	
Figure 4-11: Spatial-Temporal Risk Model for Particle Time at: a) 2s, b) 3, c) 5s d) 10s e) 15s. 69	
Figure 5-1: ML Model Development Overview .....	71
Figure 5-2: RAM Model Development Overview.....	72
Figure 5-3: Cells with no particles.....	73
Figure 5-4: Particles reach cell elements and get counted.....	73
Figure 5-5: Particle Diameter Weighting Function.....	74
Figure 5-6: Particle Risk Factor as a function of time-summed particle concentrations.....	75
Figure 5-7: The results of the DNN model’s XZ Risk Gradient at y=0.95 m and 0.5 s .....	80
Figure 5-8: The results of the DNN model’s XZ Risk Gradient at y=0.95m and 2.5s .....	81
Figure 5-9: XZ Risk Gradient at y=0.95m and 5.0s .....	81
Figure 5-10: Example of the bounding box Created by Regional Risk Model .....	84

## Table of Tables

Table 2-1: Summary on Common Machine Learning Architectures	13
Table 2-2: Results from Mesgarpour et al.'s study [67]	22
Table 3-1: Domain Dimensions for Bergeron Office Space	26
Table 3-2: Original Values for Office Space Mesh Element Sizing	27
Table 3-3: New Values for Office Space Mesh Element Sizing	30
Table 3-4: Mesh Metric Comparison (Initial vs New Mesh)	31
Table 3-5: Boundary Conditions for Office Space CFD Simulation	33
Table 3-6: DPM Conditions for Office Space Simulation	34
Table 3-7: Injection Conditions for Office Space Simulation	34
Table 3-8: Results of Numerical Sensitivity Analysis for Turbulence	41
Table 3-9: Noted System Values for Ventilation Strategies after 80 s Flow Time	50
Table 3-10: Performance Indicators for each Ventilation Strategy after 80s Flow Time	51
Table 3-11: Quantities of Particles of different after 10s Particle Time	51
Table 5-1: Validation Results of STR Machine Learning Models	80
Table 5-2: Validation ML Results for Regional-Temporal Model	83

## List of Parameters

Subscript $k$	corresponding parameters of turbulent kinetic energy
Subscript $\epsilon$	corresponding parameters of turbulent dissipation rate
Subscript $_{eff}$	corresponding to effective quantity
Subscript $p$	corresponding parameters of particles
$k$	Turbulent Kinetic Energy, $m^2/s^2$
$\epsilon$	Turbulent Dissipation Rate, $m^2/s^3$
$\rho$	Continuous Phase Density of Fluid, $kg/m^3$
$\alpha$	Inverse Prandtl Number
$\mu$	Molecular Viscosity
$G_k$	Generated Turbulent Kinetic Energy, $m^2/s^2$
$G_b$	Generated Turbulent Dissipation Rate, $m^2/s^3$
$C$	Turbulent Dissipation Constants
$Y_M$	Turbulent Transport Term
$S_k$	Turbulent Inertial Term
$S_\epsilon$	Turbulent Skewness Term
$R$	Turbulent Response Term
$F_D$	Drag Force, $kg \cdot m/s^2$
$g$	Force of Gravity,
$u$	Velocity, $m/s$
$d$	Diameter, $m$
$C_c$	Cunningham Correction Factor
$\lambda$	Molecular Mean-Free Path, $m$
$T_e$	Exhaust/Outlet Temperature, $K$
$T_s$	Supply/Inlet Temperature, $K$
$V$	Fluid Inlet Velocity, $m/s$
$P_v$	Particle Value
$P_d$	Particle Diameter Weighting
$P_w$	Cell Weighted Particle Value

## List of Abbreviations

ADPI	Air Diffusion Performance Index
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Program Interface
BC	Boundary Condition
CBPT	Cell Based Particle Tracker
CFD	Computational Fluid Dynamics
CRE	Contaminant Removal Efficiency
DNN	Deep Neural Network
DPM	Discrete Phase Model
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HPC	High Performance Computing
HRE	Heat Removal Efficiency
HVAC	Heating, Ventilation, and Air Conditioning
IAJS	Intermittent Air Jet Strategy
IC	Initial Condition
IOM	Index of Mixing
LTS	Local Thermal Sensation
MAE	Mean Average Error
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MPI	Message Passing Interface
MSE	Mean Squared Error
MV/DV/SV	Mixing, Displacement and Stratum Ventilation
OS	Operating System
OTS	Overall Thermal Sensation
PAE	Percentage Average Error
PC	Personal Computer

PCV	Particle Count Value
PDW	Particle Diameter Weighting
PI	Performance Indicator
PRESTO!	Parallel Reynolds-Averaged Navier-Stokes Toolkit for Optimization
PRF	Particle Risk Factor
RAM	Risk Assessment Model
RANS	Reynolds-Averaged Navier-Stokes
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RH	Relative Humidity, %
RMSE	Root Mean Squared Error
RNG	Re-Normalisation Group
ROM	Reduced Order Model
RTR	Regional-Temporal Risk
SSH	Secure Shell
STR	Spatial-Temporal Risk
SVR	Support Vector Regression
TUI	Text User Interface
UDF	User Defined Function
UDM	User Defined Memory
VNC	Virtual Network Computing
WB	Work Bench
WPC	Weighted Particle Count

# Chapter 1. Introduction

## 1.1. Background

The last five years have seen a myriad of historical events and innovations that has forever shaped the way our world operates. One such event, is the devastation of the Covid-19 pandemic and subsequent lockdowns. SARS-CoV-2, commonly referred to as Covid-19, was first discovered in December of 2019 in Wuhan China [1,2] with the first case reported on the 31<sup>st</sup> with symptoms typical of pneumonia. Action prompted by the Chinese Center for Disease Control and Prevention (China CDC), World Health Organization (WHO) and other governmental bodies in China occurred to try and mitigate the spread in January 2020 [1]. However, the virus had already become international; in mid-January 2020, the first case outside of China was confirmed in Thailand [3]. By the end of January, cases had been confirmed in Japan, the Republic of Korea and in the USA, among other countries. As the months continued to go on, global cases and deaths related to the virus continued to rise rapidly. By July of 2021, total cases and number of deaths had reached 200 million and 2 million respectively [1,3]. Currently, these numbers are predicted to be over 700 million and 6 million [4]. As an airborne virus, COVID-19 was highly infectious having multiple modes of transmission. Generally, the virus can spread through three modes: direct contact, airborne or fomite (indirect) [5]. Many infected persons can be infected but asymptomatic, meaning they could spread the virus without knowing they are sick and evading detection [6,7].

With how devastating the effects of the pandemic were, global collaborative efforts began. The general scientific community as well as private and public organizations, began to utilize various modeling techniques to aid in predicting the spread of infection, in hopes of finding best practices or other preventive measures. Besides more broad methods utilized during the Covid-19 pandemic, (limiting indoor capacities, lockdowns, etc.) investigations into more long-term solutions involved investigating the role of ventilation in the spread of airborne diseases [8–12]. Generally, it has been found that ventilation plays a significant role in movement of airborne diseases [8,9,11,13]. However, it is not practical to extract generalized recommendations from purely experimental studies. This is where various mathematical models can, and have been, utilized [14]. Of these models, two are more widely well known: disease (or epidemiological) modeling, which is a type of statistical modeling, and computational fluid dynamics (CFD) simulations. CFD is a type of numerical modeling that can incorporate temporal-spatial aspects

---

that many mathematical models lack. Many well-known studies [8,15–22] utilized CFD to extract information on experimental results or to simulate alternative implementations without losing fidelity to experimental results.

In the same vein of mathematical modeling, Machine Learning (ML) is another significant tool that has wide reaching applications in many fields. Machine Learning encapsulates groups of algorithms that can automate the processes of extracting information from data [23]. ML can quickly and efficiently apply algorithms previously used manually by humans (optimization, regression) to large data sets to create models [24]. ML is quickly gaining in popularity across many fields such as manufacturing, finance, 3D printing and more [25–28]. Because of these unique properties, it is an invaluable tool that is readily available for use in CFD, since it often deals with large data sets. Various applications such as optimizing airplane wings for optimal lift to drag or estimating flow fields from limited properties are all possible. While these things have and will continue to be done by humans, ML can aid immensely in the design [29,30] and optimization [31,32] process.

Finally, high performance computing (HPC) is becoming more available when combined with cloud computing service. HPC enables research access to accelerated computation resources, which often use computers with high amounts of memory, central processing units (CPUs) and graphics processing units (GPUs), all of which are useful in running high speed computations. Many researchers are already using this in a variety of domains to accelerate simulation speeds [33,34]. In addition, many utilize parallel processing (MPI, OpenMP) that can further enhance speeds by allowing the use of multiple computers (nodes), CPUs and GPUs in parallel [34]. Splitting processes in this way vastly increases speeds by allowing multiple processing units (PUs) to operate in tandem. ML models, optimization algorithms and other numerical/mathematical models also benefit from these computational resources, which can be readily available online in many cases.

CFD as a model building tool is already an invaluable resource. However, it is limited in computation time among other factors [35]. Trying to reduce the resources required to run high fidelity simulations will often significantly reduce the reliability of simulation results [35]. By utilizing ML and HPC as tools, novel methods and workflows can be developed to help current future researchers perform high fidelity simulations quickly. These technologies can be further enhanced with customized scripts to help take advantage of the benefits of all of them.

## 1.2. Motivation

The damage caused by the pandemic and subsequent lockdowns are astronomical in both economically and regarding the loss of human life and well-being. Since the start of the global COVID-19 (SARS-CoV-2) pandemic, significant strides have been made into researching aspects surrounding the virus. The SARS-CoV-2 virus is known to spread through four primary routes: direct contact, indirect contact, droplet transmission and through airborne routes [5]. Airborne transmission occurs when the virus spreads as an aerosol, allowing it to stay suspended in air for long periods of time [36]. A considerable amount of research has focused on creating numerical models to predict the spread of the virus in indoor locations [37–39]. CFD simulations are widely used globally and can simulate the airflow patterns within a room considering the Heating, Ventilation and Air Conditioning (HVAC) systems. However, there lacks a clear linkage between the mathematical models and the CFD models. Recently, CFD has begun to utilize discrete risk assessment methodology (RAM) to bridge this gap. Given the large datasets that CFD produces, incorporation of newer technologies like machine learning to further evaluate airborne transmission routes is far behind due to the computational cost of running large amounts of CFD simulations. This is where machine learning can be used to enhance the abilities of conventional computational dynamics research [24,40–42].

Many researchers have begun utilizing ML in a variety of fields on this trend is likely to continue to grow, especially with advent of natural language processing (NLP) which is becoming mainstream in the form of chatbots like OpenAI’s ChatGPT [43–45]. Previous work done by colleagues in the Freire-Gormaly Lab [11,13] show indications that CFD can become even more useful if supplemented with techniques from other domains such as statistical modeling, disease modeling mathematical modeling. This thesis aims to aid in the growing number of novel developments that use AI to develop more efficient tools for better utilization of CFD. Many fields already use, and are beginning to use ML, in the field of fluid dynamics and solid mechanics[46], including many mainstream industries such as: aerospace [47], medicine [48], manufacturing [49], mining [50], microscopy [51,52], climatology [53] and others [25,27,34,41,42,54,55].

### 1.3. Research Objectives

The main objective of this research is to further investigate the role ventilation plays in spread of airborne diseases using CFD and develop predictive models to accomplish the same. To aid in this endeavour, a useful methodology to utilize ML, CFD and HPC to predict how these diseases spread in indoor environments will also be developed. By combining ML and HPC with the long-standing field of CFD, future research will progress much more rapidly. This also provides an easy to access resource for future researchers to aid in the development processes of new CFD simulation methodologies, with less limitations than previous generations of researchers. Thus, the specific goals of this research are:

- Continue previous research to use CFD to analysis the effects of ventilation of the spread airborne diseases.
- Develop a novel process to utilize HPC in enhancing CFD simulations.
- Develop a RAM to transform particle data into a spatial-temporal risk mapping.
- Create a novel methodology to predict the spread of airborne diseases using risk assessment modeling and machine learning.
- Analyze the effectiveness of various ML models and archetypes for use in predicting particle dispersion.

### 1.4. Thesis Outline

This thesis is organized into six chapters. Chapter one deals with introduction to the research, including background, motivation, and research objective. Chapter two provides a literature review on current CFD practices and usage as it relates to Covid-19 and performs a background and literature review on ML and modern applications to CFD. Chapter three details the investigation into varying ventilation strategies in small office spaces. Chapter four describes the development of a tool to enhance the extraction of particle data, specifically for use with CFD software. Chapter five outlines a novel method to integrate CFD and machine learning to predict temporal-spatial particle spread using a parallelized risk assessment model. Chapter six summarizes the conclusions of the three major studies and outlines avenues for future work.

## Chapter 2. Literature Review

### 2.1. Transmission of COVID-19

The spread COVID-19 has caused significant damage with its rapid ability to infect large swaths of people. This, along with the many vectors of travel make it difficult to predict infection rates. Currently, epidemiology uses mathematical and statistical methods to create models. Mathematical models use mechanistic understanding of the problems to develop models, whilst statistical models are mainly data driven [56–58]. Both these approaches are useful for modeling the general progression of the virus at large time scales and with large numbers of samples. On the other end of the spectrum, physical models and numerical models are much more suited for situations involving much fewer individuals, where time and distance scales are much smaller. This use of experimentation and numerical modeling is a useful way to look at more specific cases of the propagation of airborne diseases.

COVID-19 is a respiratory infectious disease caused by the family of coronavirus [4,22,59]. Transmission of the viral load occurs directly or indirectly by way of contact transmission, airborne transmission, or fomite transmission [5]. Airborne transmission occurs via fluid particles [60,61], namely droplets and aerosols. These particles are heavily affected by complex flow physics and can be difficult to characterize. One method, proposed by Rosti et al. [62], delineates the transmission in terms of three modes. Ballistic, inertial and tracer transmissions. Ballistic transmissions are short range and occur in larger and heavier droplets where the forces due to drag and gravity are large. Inertial transmissions are short medium ranged and dominated by the initial particle velocities as well as the ambient conditions. Finally, tracers are long range that are dominated by external air flow. Figure 2-1 below outlines these three transmission modes.

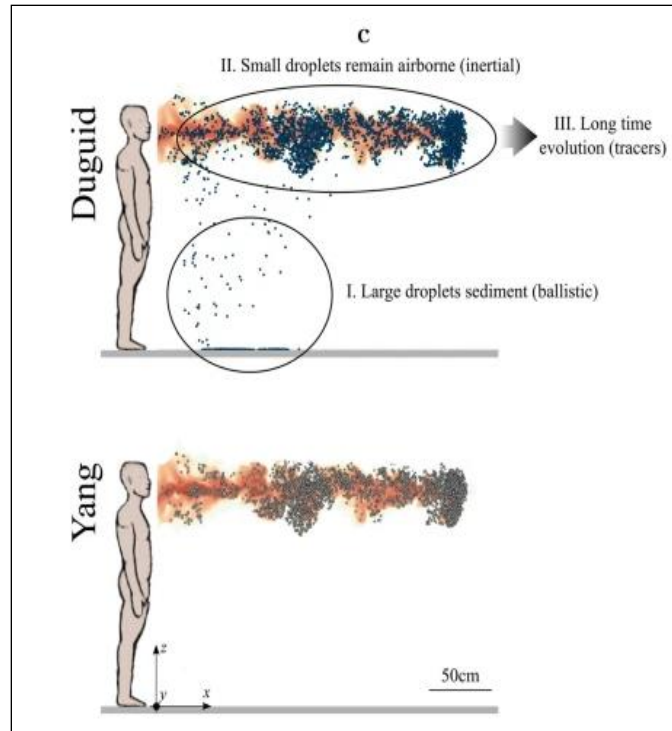


Figure 2-1: Comparison of Viral Sedimentation for 60% and 40% RH [62]

This study will focus on inertial and ballistic transmission, pertaining to short to medium range transmission. This is because larger droplets are much more dangerous and pose a higher risk of infection [63]. As a viral cloud expands it also thins, reducing risk of infection at further distances. Only specific indoor conditions will be considered since they have consistent ambient factors, include dynamic of HVAC and transmissions typically happen at small distance and time scales.

Airborne transmission occurs via a mixture of different sized droplets and aerosols, which in turn affect how they will travel. While the exact delineation between what constitutes one versus the other is not defined clearly in literature, many sources have made attempts to classify the two [64,65]. After a biological injection (cough, sneeze, etc.), a mixture of aerosols and droplets are released. Smaller droplets can quickly evaporate and turn into further aerosols. Droplets tend to drop quickly while aerosols, being much smaller and lighter, tend to linger in the air for longer periods [66,67]. An experimental study by Duguid [68] investigated droplets carrying bacteria/nuclei for droplet diameters from  $0.25\text{ }\mu\text{m}$  to  $2000\text{ }\mu\text{m}$ . They found that droplets in the  $100\text{-}2000\text{ }\mu\text{m}$  range fell quickly to the ground while  $1\text{-}100\text{ }\mu\text{m}$  would become airborne.

Airborne transmission is heavily affected by the airflow. Thus, many factors can contribute to its spread. Many sources find that relative humidity (RH) and ambient temperature contribute significantly to higher risk of transmission [69,70]. Warmer and wet climates are found conclusively in literature to be areas with higher risk of infection [69]. The following environmental conditions all contribute to the physical flow phenomena including, but not limited to gravity, inertia, buoyancy, drag/lift, evaporation and condensation and others. However, when looking at indoor environments, another more dominant factor takes focus: ventilation.

## 2.2. HVAC and CFD

HVAC stands for heating, ventilation, and air cooling. It refers to the industry that deals with heating, cooling, and otherwise ventilating indoor spaces. There are various possible ventilation configurations, however the common general configurations can be classified into ventilation strategies [9,12]. Due to the broad implementation of ventilation no single experimental or mathematical study can have much agreement on what ventilation is preferred. Typically, the most common two investigated are displacement ventilation (DV) and mixing ventilation (MV). Additional ventilation systems commonly explored are stratum and underfloor ventilation. As outlined by Khan in her literature review on ventilation systems, literature has found that any one ventilation system can have various benefits over others [11]. Things like thermal comfort, energy efficiency, ease of implementation, air dilution and others make it difficult to select any one strategy over the other.

This does not mean that careful consideration of ventilation is not important. Ventilation is one of the most important factors that can affect the dispersion of airborne diseases [8,66,70]. To that end, CFD is a valuable tool to predict spread without costly experimental studies. Many studies have investigated the effects of ventilation on airborne transmission using CFD, but there is no consensus on the settings to use across all simulations. Notable studies include those by Shao et al. [6], Yang et al. [71], Talaat et al. [21] and Liu et al [20]. Although an older study, an airplane cabin study by Yan et al. is also notable [72]. A previous literature review by Khan [11] summarized the main results from their simulations and compiled a CFD methodology that extracted the major settings of each CFD simulation. It was found that although the earlier simulation studies [20,21,71,73] had various methods used, there were a series of common or reoccurring settings that provided accurate and high-fidelity simulations. Khan [11] also concluded

that the location of various ventilation elements (inlets, outlets etc.) can have dramatic impacts on the spatial and temporal distribution of airborne diseases.

### 2.2.1. CFD Study by Khan

The methodology used by Khan [11] for her CFD simulations were taken as a good example of high-fidelity study since the experimental study by Yin et al. [8] was used as the basis for a validation study. Khan [11] found her CFD settings provided sufficient convergence with experimental data from Yin et al. [8] at selected locations in the domain to validate the fidelity of her CFD simulation.

Khan's study [11] utilized a Reynolds Averaged Navier-Stokes (RANS) model with a pressure-based solver. Her software of choice was ANSYS 2020 R2. Turbulence was modeled with an RNG  $k - \epsilon$  model with standard wall functions. This model is useful for modeling indoor environments since it can simulate both high and low Reynolds numbers [11]. The kinetic energy and dissipation terms are outlined in Equations [2-1] and [2-2] below as outlined in [74]:

Kinetic energy,  $k$ :

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j} \left( \alpha_k \mu_{eff} \frac{\partial k}{\partial x_j} \right) + G_k + G_b - \rho \epsilon - Y_M + S_k \quad [2-1]$$

Dissipation rate,  $\epsilon$

$$\frac{\partial}{\partial t}(\rho \epsilon) + \frac{\partial}{\partial x_i}(\rho \epsilon u_i) = \frac{\partial}{\partial x_j} \left( \alpha_\epsilon \mu_{eff} \frac{\partial \epsilon}{\partial x_j} \right) + C_{1\epsilon} \frac{\epsilon}{k} (G_k + C_{3\epsilon} G_b) - C_{2\epsilon} \rho \frac{\epsilon^2}{k} - R_\epsilon + S_\epsilon \quad [2-2]$$

In addition, the energy function was activated to simulate the thermal effects on the flow. Buoyancy was modeled using the Boussinesq approximation model with the appropriate density and thermal expansion terms. Pressure-velocity coupling was solved with the coupling method and the pressure was solved with the PRESTO! Pressure interpolation method. Least squares cell based gradient discretization was used and other parameters used second order upwind. Particle injections were solved using a couple Eulerian-Lagrangian Discrete Phase Model (DPM). Flow was modeled using a steady state flow while particles coupled with the flow, used unsteady (transient) particle tracking. Relevant ANSYS Fluent equations for Lagrangian reference frames are shown below:

Particle Velocity,  $u_p$

$$\frac{du_p}{dt} = F_D(u - u_p) + \frac{g(\rho_p - \rho)}{\rho_p} + F_a \quad [2-3]$$

Drag Force,  $F_D$

$$F_D = \frac{18\mu}{\rho_p d_p^2 C_c} \quad [2-4]$$

Cunningham correction,  $C_c$

$$C_c = 1 + \frac{2\lambda}{d_p} \left( 1.257 + 0.4e^{-\left(\frac{1.1d_p}{2\lambda}\right)} \right) \quad [2-5]$$

The Thermophoretic force and the Saffman lift force were included in the DPM model, to simulate forces due to temperature gradients and near wall shear forces. The particle used were modeled using the Rosin-Rammler diameter distribution with droplet diameters based on Duguid's experimental results [68]. Khan used the Rosin-Rammler/Weibull to fit the particle data to a distribution. From that, a probability density function (PDF) and cumulative density function were made, the specifics of which are explained in Khan's thesis [11].

Khan [11] performed CFD simulations on a small office space, based off an office in York University [75]. She analyzed three cases: layout with no outlets, a layout with a far-off outlet and one with an overhead outlet. Her goal was to study if ventilation configuration played a significant role in airborne transmissions in small spaces. She found that adding new exhausts did not significantly change the risk of particle exposure, and in fact showed that it can lead to worse situations due to pressure-based mixing within the domain. She suggested that a change in ventilation design might be a possible solution to this issue. She specifically suggested that displacement would likely perform the best given its configuration to move air upward. In other studies, with a larger domain (lecture hall), Khan utilized mathematical disease modeling to enhance her simulations and extract more meaningful data.

Khan's study [11] and others [20,21,71–73], showed that ventilation is important. She also showed that CFD can be effectively utilized to predict dispersion of airborne particulates and validated her methodology to experimental results. She utilized mathematical modeling to enhance her CFD model and create a risk assessment model (RAM) based on temporal-spatial data. This study will attempt to do something similar, but instead of using disease modeling techniques, machine learning will be used.

## 2.3. Machine Learning

### 2.3.1. Introduction to ML

Machine learning (ML) is a new and emerging field that at its essence, is building on the many years of knowledge in optimization [76–78], computer science [79,80] and statistics [81,82]. ML is a branch of Artificial Intelligence (AI) that refers to the process of creating or training algorithms to learn from data to make decisions or predictions [23]. ML has become explosively more popular due to the emergence of Artificial Neural Networks (ANN), algorithms that have become extremely common, and the advancement of computational capabilities. ML mainly operates on a black box. This means that the understating of why the model works, how it makes decisions, and sometimes how or what it makes certain correlations with, are hidden from observation. While this means that ML cannot be used to deepen the understanding of fundamental theatrical knowledge about the physical world [83]. Instead, it can be used in combination with classical mechanistic engineering models. Mackay and Nowell dub these meshing of approaches as ML “gray box” models [83]. These models are physics-guided using ML as a mediator between real world quantities as inputs and outputs. ML can be configured to reflect the physical constraints (ex. differential relations), probabilistic relations or simulation results. With some understanding of the underlying physical mechanisms, ML can be used more confidently to make models around real-world phenomena.

### 2.3.2. ML and Neural Networks

On basic principle, ML can be thought of as simple black box models where inputs and outputs are fed and used to create a comprehensive algorithm that accurately models the physical process in question (in the context of engineering problems). Figure 2-2 outlines the general process for creating ML models in engineering applications.

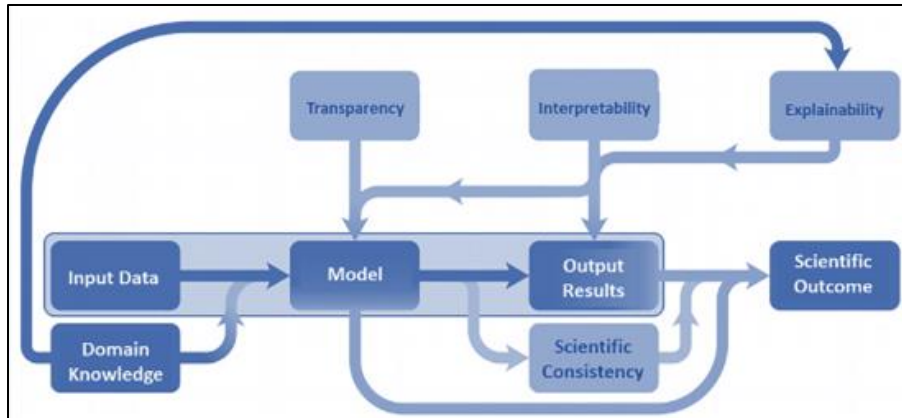


Figure 2-2: ML Pipelines for Application in Engineering [83,84]

Input data is used to train the model which can then be used to predict outputs for other data sets. Machine learning models come in many forms, but they can generally be classified into three major classes: Reinforcement Learning (RL), Semi-Supervised Learning (SSL), and Supervised Learning (SL) [85]. RL works by optimizing an algorithm in a live environment. The ML model will use reward or penalty functions (like loss and fitness functions referenced in optimization spaces), to train the model to encourage optimal behaviour. SL is the opposite. It works in a closed system environment. All the required data is given to the model in a “supervised” context, meaning that the user training the model will tell the model what is or isn’t ok to take in as data. In the middle of these, SSL works by running a pre-built model in an open environment where it can still take in information and change. It operates by using both principles of SL and RL [24,83]. An overview of these learning techniques is shown in Figure 2-3, below.

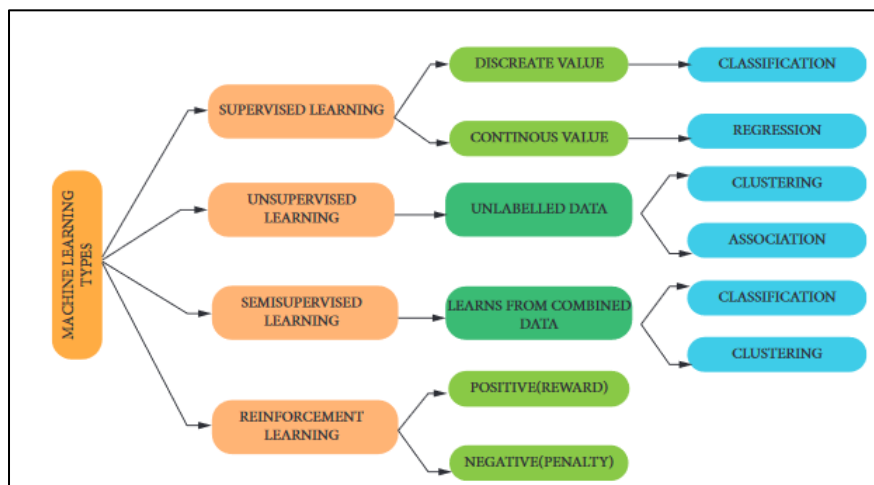


Figure 2-3: Machine Learning Techniques [83]

An Artificial Neural Network (referred to as ANN) is a type of model based on the biological nervous system, where an individual neuron can send and receive signals from other neurons. A neuron makes decisions by summing the weighted signals of inputs into the neuron. The neuron has a bias applied in conjunction with the sum of the weighted sum of signals. This value is then sent to an Activation function, which calculates the value of the neuron [83,85]. The figure below shows this process in graphical form.

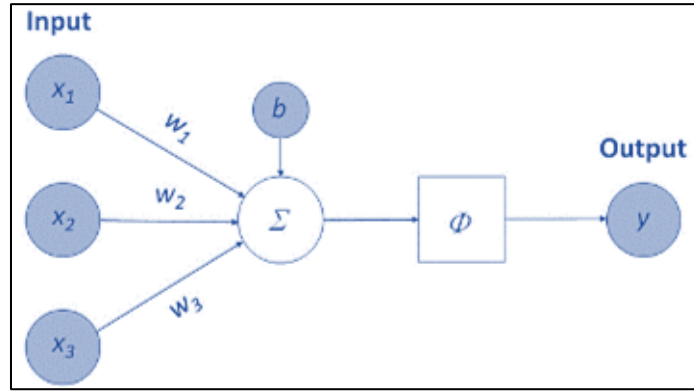


Figure 2-4: General Flow Diagram for Neuron in a Neural Network [86]

Neural Networks are useful for doing what is commonly referred to as model reduction. They can take in input data and find a lower dimensional manifold representation of that data, commonly referred to as a Reduced Order Model (ROM). They can then use that manifold to then predict other parameters [86]. An example of this processes was outlined by Brunton et. al. [86] and it is shown in Figure 2-5, below.

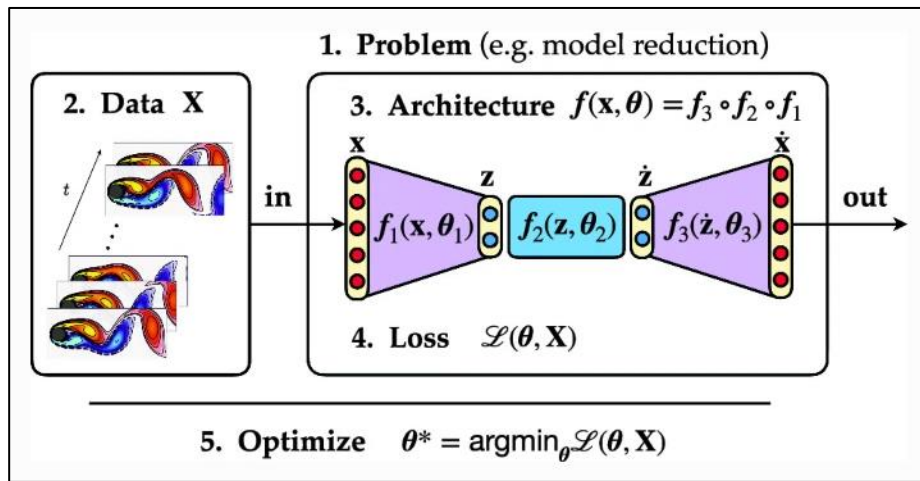


Figure 2-5: Process of Creating a Reduced Order Model [86]

The function  $y = f(x; \theta)$  represents the architecture of the ML model. There are various ML architectures that can be used, and associated parameters, hyper parameters, activation functions and other such associated values that go into building any such model. For brevity, the common ML models and related info is displayed in the table below. Table 2-1 provides a brief description of the ML models, pros and cons, the topology (algorithmic structure).

Table 2-1: Summary on Common Machine Learning Architectures

ML Model / Architecture		Description
Linear Regression (LR) [87]	Simple	Statistical method involving finding the linear relationship between two variables. Simplicity and ease of use make it very popular.
	Multiple	Multiple Inputs into a LR model. Can also have MIMO (multiple outputs) LR models.
Decision Tree [88]		A non-parametric learning algorithm used for regression and classification activities. Its structure includes branches, leaf nodes, internal nodes, and root nodes.
Random Forests [89]		An algorithm that combines the result of multiple decision trees to find a single output. It can handle both classification and regression problems.
Support Vector Machines (SVM) [90,91]		Known to be a robust classification and regression technique which maximizes prediction accuracy without overfitting issues. Commonly used to analyze large data values of predictor fields. Applications include CRM for businesses, facial recognition, text mining concept extraction, etc.
Deep Neural Network (DNN) [92,93]		Essentially, this is a neural network with at least 2 layers (enable learning of intricate patterns), used to process data in complex ways by implementing math modelling. This allows them to be powerful for image and text analysis tasks.
Convolutional Neural Network [94]		A type of DNN that is designed for processing grid data such as images. Its layers are specialized to learn and extract relevant features from input data, effective for image recognition tasks.
Recurrent Neural Network [95,96]		A type of neural network designed for processing sequences of data, such as time series or natural language. It has loops that allow information to be passed from one step of the sequence to the next, enabling it to capture temporal dependencies and patterns in the data.
Long Short-Term Memory (LSTM) [96]		A type of recurrent neural network (RNN) architecture designed to overcome the vanishing gradient problem and capture long-range dependencies in sequential data. LSTMs utilize memory cells and gating mechanisms to efficiently learn and store information over extended sequences.

Generative Adversarial Networks (GAN) [97]	A type of deep learning framework consisting of two neural networks, a generator and a discriminator, that work in tandem. The generator creates synthetic data samples, aiming to replicate real data, while the discriminator tries to distinguish between real and generated samples, leading to a competitive learning process that results in the generation of increasingly realistic data.	
Feedforward Neural Network (FFNN)	An artificial neural network where information flows in one direction, from input nodes through hidden layers to output nodes. It's mainly used for tasks like classification and regression, but lacks memory of previous inputs, making it less suited for sequential data.	
<b>ML Model / Architecture</b>	<b>Pros</b>	<b>Cons</b>
Linear Regression (LR)	Easy to use and good starting point for regression tasks. The model's coefficients provide insights into relationships between input and target variables. It is a computationally efficient way to work with large datasets.	Assumes a linear relationship between variables if that assumption is violated then inaccuracy can occur. Need regularization to avoid overfitting or underfitting. Sensitive to outliers, can affect performance and coefficient estimates. Violations of assumptions like normality of errors or constant variance can lead to unreliable predictions and skewed results
Decision Tree [88]	Easy to consume and understand due to hierarchical nature. There is flexibility because decision tree's do not require data preparation. They can be leveraged for classification and regression tasks. It can handle many data types that are otherwise obstacles for other classifiers like Naïve Bayes.	Not good at generalizing to new data, effectively it is prone to overfitting. Known to have high variance estimators; small variations in data can lead to a different decision tree and reducing variance is limited. It can be expensive to train this approach compared to other algorithms.
Random Forests [89]	The problem of overfitting is overcome because of the nature of random forests. This is a collection of decision trees, so the average of uncorrelated trees reduces the resulting variance and error. They also allow an individual to determine variable importance to the model (MDI & MDA) using Mean Decrease in Impurity and Mean Decrease Accuracy.	Although they can handle large data sets, this can lead to slow data processing due to each decision tree that needs to be computed. More resources are required to store and process data as a result.

Support Vector Machines (SVM) [90,91]	Uses subset of training points in the decision function resulting in high memory efficiency. Known to be effective in high dimensional spaces, even when dimensions are greater than the number of samples.	They excel in classification tasks, but they exhibit certain limitations. They are sensitive to feature scaling discrepancies and outliers, impacting their effectiveness. SVMs computational complexity and memory demands can be challenging for large datasets, and their hyperparameters necessitate careful tuning for optimal performance.
Deep Neural Network (DNN) [92,93]	Effective for learning complex features from raw data and capturing patterns. Can understand hierarchical relationships in data. They can adapt to various data types and data modalities.	Can be computationally intensive (time and hardware utility wise). Usually need large amounts of labeled data for effective training. Can be susceptible to overfitting. Configuration and tuning of parameters can be time consuming.
Convolutional Neural Network [94]	They can learn hierarchical features from images. Object detection and image classification are effectively performed. The convolutional layers in CNNs use parameter sharing, reducing the number of learnable parameters, and making them efficient for handling large image datasets. They exhibit translation invariance, meaning they can recognize patterns regardless of their position in the image.	Can be computationally intensive. Large data requirement for training. They lack human understandable interpretations of learned features and decisions. Therefore, explaining reasoning can be troublesome. Can struggle with overfitting and generalizing unseen data.
Recurrent Neural Network [95,96]	They can handle sequential and time-dependent data, great for speech recognition. They can model context and relationship that span across multiple time steps. Built-in memory to retain information allow for easy learning and remembering patterns from the earlier time steps.	Often experience problems maintaining information over long sequences. Computationally can be intensive and require large datasets for deep architectures. Lack of parallelism. Need careful initialization otherwise are prone to training instability.

Long Short-Term Memory (LSTM) [96]	Great for understanding context over extended time intervals because it can capture long range dependencies in sequential data. Common issues for RNN's including vanishing gradient are mitigated in a LSTM. They also have memory cells that allow for important context to be maintained across time steps. Its gating mechanism also allows for reduced overfitting.	Can be computationally demanding. They have a complex architecture difficult to understand. Training time can be longer due to complex structure. Needs to tune hyperparameters of the network for number of layers, units per layer and learning rate.
Generative Adversarial Networks (GAN) [97]	Can create high quality realistic data samples (images, text, image synthesis, style transfer, data augmentation). Can learn from unlabeled data. Great for a variety of tasks such as creative apps (art, music, etc.)	Training can be challenging due to hyperparameter tuning. Can be susceptible to mode collapse (limited output variety undermines diversity of resulting samples). Ethical concerns are to be aware of, including generation of deepfakes. For these large and complex models, it can be resource intensive.
Feedforward Neural Network (FFNN)	Can approximate complicated functions and mappings for modelling relationships within data. Parallel processing is an important feature to note. By using activation functions, FFNN's can capture non-linear patterns in data. They can also learn hierarchical features for manual feature engineering.	They lack memory of prior inputs, not great for sequential data tasks. Requires large amount of labeled data for overfitting. Selecting parameters for tuning can be challenging, a lot of manual experimentation and tuning needed. For high-dimensional data, feedforward networks might suffer from the curse of dimensionality, where the network's performance deteriorates due to data sparsity.
<b>ML Model / Architecture</b>	<b>Applications</b>	
Linear Regression (LR)	Sales prediction, treatment impact studies, age and income correlations, stock market prices and indicators, estimating patient outcomes based on medical parameters.	
Decision Tree [88]	Identifying disease based on patient symptoms, medical history, etc. Credit score and worthiness of people by analyzing financial information for loan approval. Image classification. Manufacturing quality control of products by analyzing defects and sensor data in processes.	
Random Forests [89]	Anomaly detection for fraud related crimes and network security. Assessing object recognition and image segmentation for medical diagnosis.	

Support Vector Machines (SVM) [90,91]	Classify emails as legitimate or spam based on various features. Medical diagnosis using conditions and disease prediction. Image classification in applications like vehicle and surveillance systems. SVM's can also be used for fraud transaction identification from online payment systems and banking data.
Deep Neural Network (DNN) [92,93]	Object recognition tasks and classifying within images such as satellite imagery analyzation. Crucial in NLP (natural language processing) tasks where machines can understand human language effectively. Autonomous driving where processing sensor data and detecting objects in order to make decisions is critical.
Convolutional Neural Network [94]	Identifying objects, animals, scenes within images, powering image search engines, tagging systems. Robotic development where object detection is necessary. Analyzing medical images such as X-rays, CT scans, MRIs for organ segmentation or identification.
Recurrent Neural Network [95,96]	Speech recognition where voice assistant applications or transcription services are needed. Time series analysis were predicting and modelling data over time for sequences is important. Great for stock market trend modelling and weather forecasts.
Long Short-Term Memory (LSTM) [96]	Used extensively in machine learning tasks where it is needed to capture long range dependencies and nuances in language. Speech recognition applications related to audio sequences and improving language to text features. Great for time series forecasting such as energy consumption predicting, weather patterns.
Generative Adversarial Networks (GAN) [97]	Widely used for image generation of artwork, realistic faces, landscapes, and content creation. Can also generate training data to improve robustness and generalization of machine learning models.
Feedforward Neural Network (FFNN)	Used for image recognition, speech, and character recognition as well to identify patterns in data. Can also be employed to predict numerical values in regression tasks such as sales forecasts and stock market trends. Interest rate determination and credit score analyzing is also an application of FFNN's using financial and personal factors.

### 2.3.3. Running and Evaluating ML Models

ML models are simple to create and utilize. The challenges come in refining parameters and hyperparameters to find the optimal configuration. This is due to the black box (gray box) model of ML models in that the internal workings are not visible. Nor is there a way to develop a strong understanding of the correlation between the input and output quantities. Thus, designing of ML models have an aspect of trial and error. ML models need data that is segmented into Training (to make the model), Testing (to test against to see if the model is good) and validation (a second set of testing data used to evaluate the final accuracy of the model, but that is typically

not included in training/testing. After a model is made, new data can be sent, and it can be used to predict new data for the future [24,83,86,98].

Evaluation of MLs is typically done with two main criteria, Mean Average Error (MAE) and Mean Squared Error (MSE). These criteria are particularly useful for regression tasks. MAE is the average of the absolute differences between predicted and actual value. It measures average errors without considering the direction. Its formula is shown in the equation below.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad [2-6]$$

MSE computes the average of the squared differences between predicted and weighted averages, where it gives higher weight to large values, as it is an exponential function.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad [2-7]$$

Lower values of MSE and MAE indicates a better fit for both metrics. MAE is more commonly used as it can resist against the effects of outliers, since it does not see direction of errors and does not magnify larger ones (as opposed to MSE).

## 2.4. Machine Learning and Computational Fluid Dynamics

### 2.4.1. Study by Mirzaei et al.

A study by Mirzaei et al. [99] demonstrated a method to utilize ML in conjunction with CFD to produce airborne concentration data. Later, they show the methodology for combining a simple Risk Assessment Model (RAM) and artificial neural networks (ANN) to develop a prediction model using only CFD results. They created a methodology to fit spatial-temporal risk data from CFD results to an ANN model to predict the risk cloud expansion as a function of time. They first used CFD to perform a simple simulation in an empty domain where they freely injected particles.

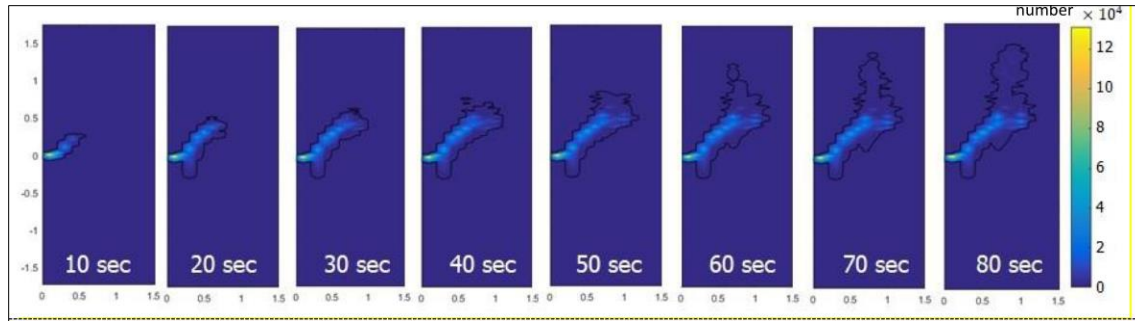


Figure 2-6: 2D Concentration plots over time for Mirzaei et al. Study [99]

Mirzaei et al. Study [99] then recorded the particle count in each location of a secondary mesh, which is overlaid in the domain. The secondary mesh records the number of particles passing through each cell. The result is the total particle count per cell.

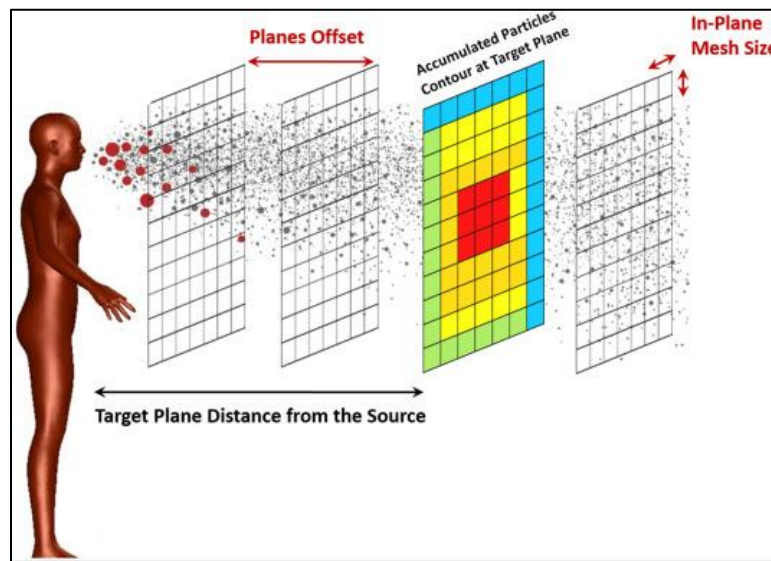


Figure 2-7: Graphic of Methodology for Mirzaei et al. Study [99]

Mirzaei et al. [99] then created an ANN with feed-forwards multi-layer perception architecture. They used a back-propagation learning paradigm to build the model. They used a sigmoid function with a smooth gradient. They plotted the results for various neural network (NN) architecture, but all the cases performed with relatively high values for percentage error. Their testing data performed better, but their validation data performed worse when compared against the training performance.

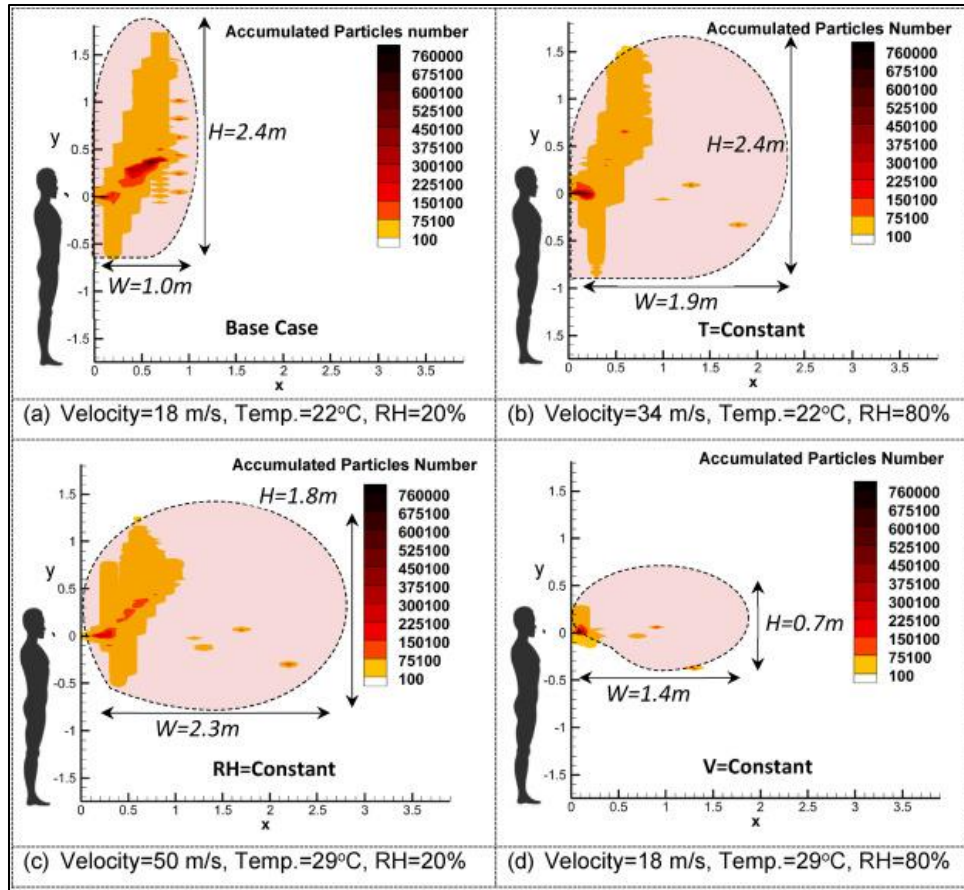


Figure 2-8: Dimensions Results from Mirzaei et al.'s study [99]

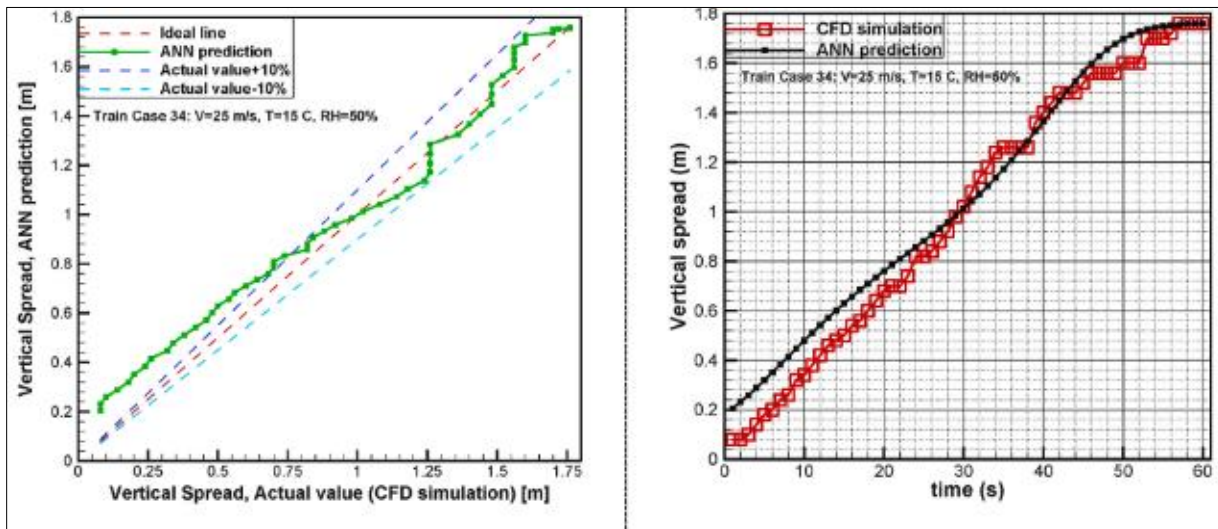


Figure 2-9: ANN Vertical Test Results from Mirzaei et al.'s study [99]

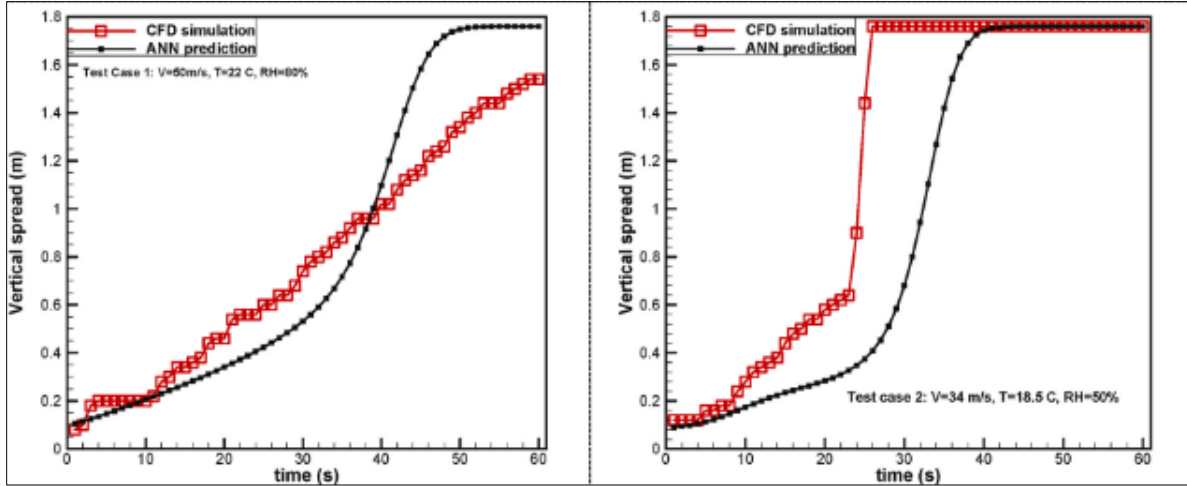


Figure 2-10: ANN Validation Results for Mirzaei et al. Study [99]

#### 2.4.2. Study by Mesgarpour et al.

A similar study was performed by Mesgarpour et al. [67]. Their study constituted multiple parts but only the elements dealing the ML model will be discussed. In a previous section of their paper, they discuss a methodology to gather spatial concentration data at various discrete planes. They want to create a MIMO (multi-input multi-output) model to predict concentration and velocity at different locations over time. The experiment consisted of around 1240 input samples. For prediction purposes, a Deep Neural Network (DNN) regression was constructed with varying input, hidden, and output layers. Sequential models in this study were built using the Sequential functions from the Keras library in Python. Keras is an open-source deep learning application programming interface (API) written in Python. Five layers were introduced using the Dense function. The initial layer acted as the input layer, acquiring attribute values like time, location, pressure, temperature, and density. The model employed three hidden layers with 30, 20, and 15 neurons respectively, with settings determined through experimentation. The Rectified Linear Unit (ReLU) activation function was applied to hidden layer neurons since it is widely used. The output layer, responsible for predicting concentration and velocity, used the Sigmoid activation function due to its adaptability for a range of values. The Adam optimizer was utilized to compute adaptive learning rates for each parameter, primarily to update hidden layer neuron weights. The model compilation was carried out using the Adam optimizer, while epoch and batch size values were determined. Following this, concentration and velocity predictions were made using the model.

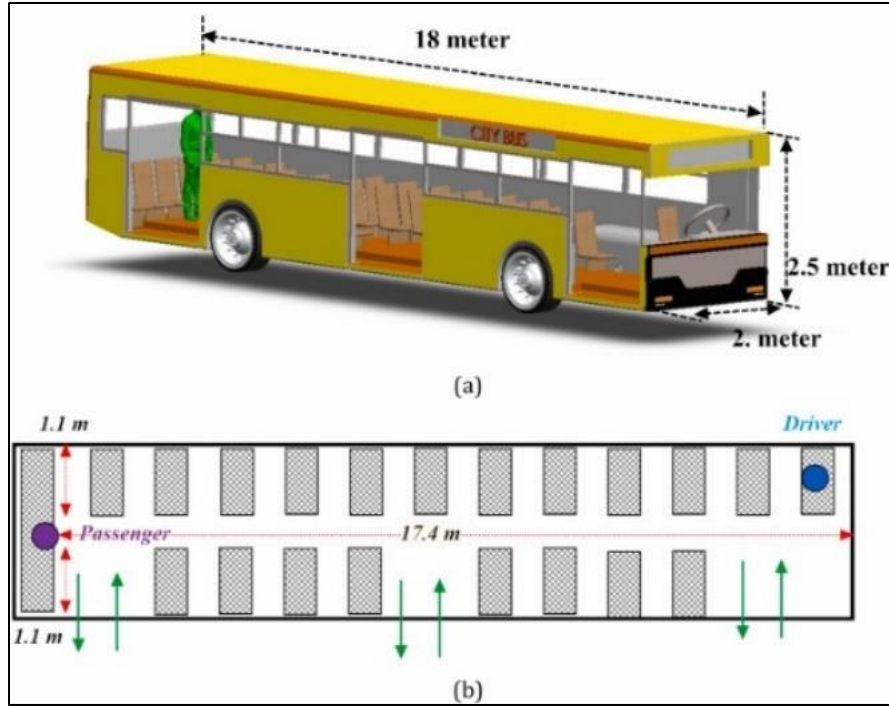


Figure 2-11: Bus Geometry from Mesgarpour et al.'s study [67]

Performance evaluation involved tenfold cross-validation, where samples were divided into 10 categories. Nine categories were utilized for training in each iteration, while one was reserved for testing. Epoch and batch size values were set at 200 and 5 respectively. Model accuracy was evaluated through Mean Square Error (MSE) and Mean Absolute Error (MAE) criteria calculations. The results of their study are shown in the table below. It shows that ML can make accurate models that predict physical quantifies, acting as a “gray box” model, as shown in the previous study.

Table 2-2: Results from Mesgarpour et al.'s study [67]

Mean Error	Various potential models							
	DNN	CFD error (%)	MLP	CFD error (%)	SVR	CFD error (%)	RBF	CFD error (%)
MAE	6.72 $\times 10^{-6}$	0.154	2.92 $\times 10^{-5}$	12.36	1.03 $\times 10^{-5}$	8.45	1.72 $\times 10^{-5}$	0.369
MSE	3.62 $\times 10^{-10}$	0.266	1.82 $\times 10^{-8}$	37.85	4.13 $\times 10^{-9}$	5.44	9.10 $\times 10^{-10}$	1.254

### 2.4.3. Summary

Both the Mirzaei et al. study [99] and the Mesgarpour et al. [67] studies demonstrate the practicality of using CFD data to predict dispersion quantities using ML. The Mirzaei et al. study [99] had a low percentage error for their training data, but the results of their testing showed large increases in percentage error in comparison to their training results. They also only considered a single ML model (DNN) and only had 35 cases. They did not have the presence of validation data set(s) to help generalize their ML model. In contrast, the Mesgarpour et al. study [67] used temporal and spatial data to have more input data (1200 samples) but were limited to only a few cases. This study aims to use the multiple case method, having different initial and boundary conditions, as well as the temporal and spatial approach, taking values at certain locations over time.

## Chapter 3. Further Utilization of CFD to Investigate Effects of Ventilation Strategies on Spread of Airborne Diseases in Small Office Spaces

### 3.1. Introduction

Indoor spaces are one of the fastest ways for the transfer of infections to occur. As outlined previously, because of the high transmissibility of the coronavirus, specifically COVID-19 and its variants, small random interactions can be sparks that start rapid infection events [56,64]. The choice of investigating small office spaces reflects this concern for indoor environments. Smaller spaces also make the creation of the simulation domain easier and quicker, and reduces computational time, as they take less time to compute. Many such spaces have been investigated previously in literature [6,20,71,73]. One such example was done previously by Khan [11]. They modeled an office space on York University Campus, in the facility offices of the Bergeron building (Office 437C). They analyzed three cases: no outlet, a singular overhead outlet and one far off outlet as shown in Figure 3-1.

This study seeks to continue to use CFD to investigate the effects of design variations on office spaces, but instead to focus on the ventilation strategy used. In industry, mixing and displacement ventilation strategies are most common [8,11]. While not new, the pandemic renewed interest in less common strategies. This study will thus contain three cases: Mixing, Displacement and Stratum Ventilation strategies. The rooms general outline includes an inlet, an exhaust door slip, and a human model among other fixtures for all three cases.

This study will contribute to literature that investigates both particle dispersion and performance indicators. An analysis will be done on a small portion of performance indicators (PI) as another metric to compare ventilation strategies. Namely, the heat removal efficiency, overall thermal sensation, and modified index of mixing. These indicators were chosen based on studies done by Zhang et al. [9] and Kabanshi et al. [100].

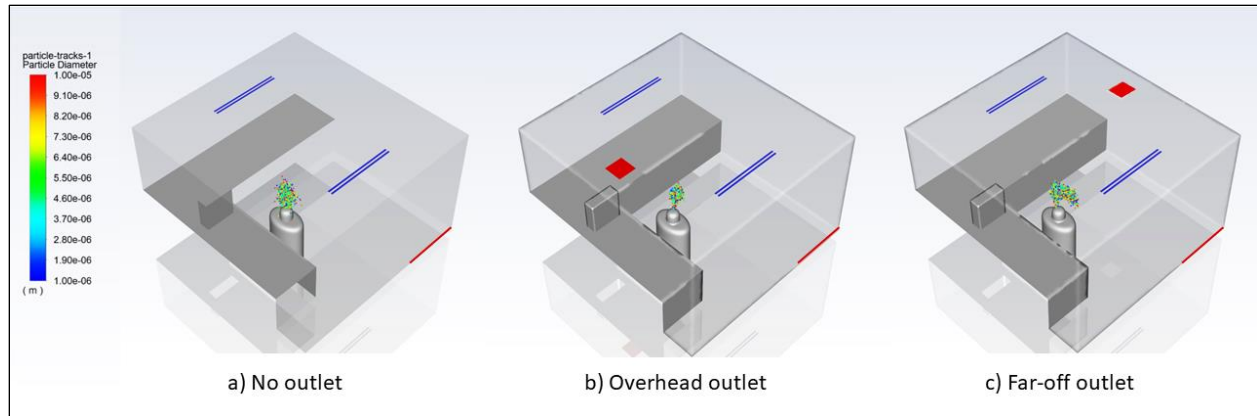


Figure 3-1: Previous Airborne Disease CFD Study by Khan [11]

## 3.2. Methodology

The creation of the simulation required a room geometry/domain, a mesh, and simulation parameters. The design methodologies for each case were virtually identical, with differences being in the refinement area(s) and the inlet velocity. This is due to the different locations of the inlet and size/number of the inlet slots.

### 3.2.1. Model Geometry

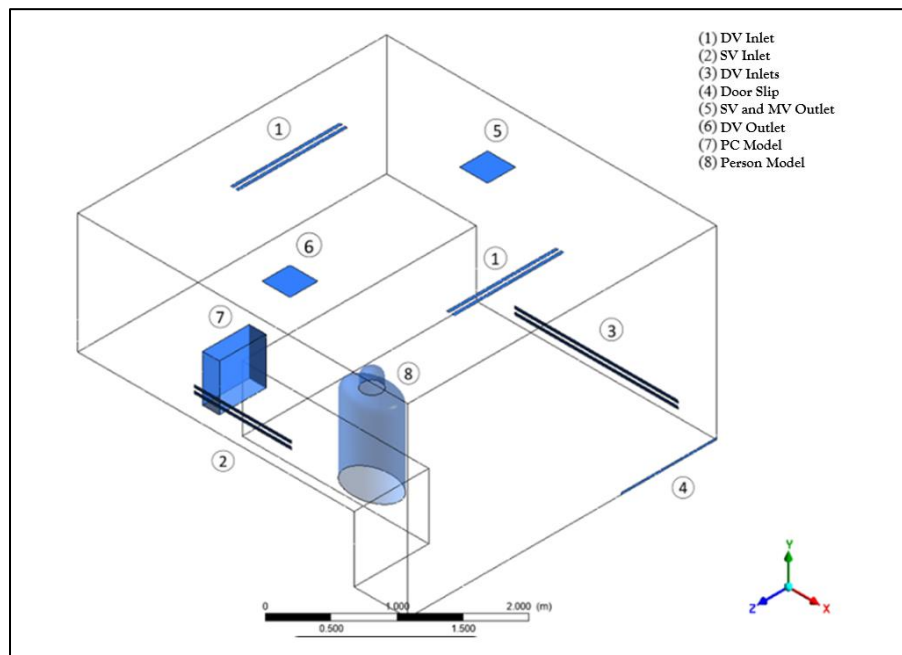


Figure 3-2: Domain and Boundaries of Berg Office Model

The office space model was based primarily on the model by Khan [11]. Slight variations to geometry importation and face placement were made, but no other significant changes were made. The model includes additional faces for the displacement ventilation (DV) and stratum ventilation strategies (SV). Slight changes were made to inlet velocities. The inlet slots have five separated slots with equal widths. Air only exits two of the five slots. The lengths of the inlets vary depending on the ventilation strategy used. The value for the domain is shown in Table 3-1. Other faces, such as the door slip, human model and PC were not changed.

Table 3-1: Domain Dimensions for Bergeron Office Space

Model Specifications		Value(s)
Room Size (L*W*H)		3.54 m x 3.31 m x 1.97 m
Inlet Slot Width		0.03 m (5 per inlet)
Inlet Slot Length	Mixing	1.17 m
	Stratum	0.97 m
	Displacement	1.35 m
Door Slip (L*W)		1.02 m x 0.02 m
Outlet (L*W)		0.3 m x 0.3 m
PC (L*W*H)		0.5 m x 0.18 m x 0.5 m
Seated Human Model (approx.)		0.6 m x 0.4 m x 1.08 m

### 3.2.2. Meshing Design

The mesh design of the office space used a uniform mesh distribution with mesh refinement near faces with associated BC. Because of the small domain and general configuration that allows for rapid particle escape or deposition, a highly refined mesh was not required, nor generally desirable as it could exponentially increase simulation time. The impact of fine mesh size on computational time would be compounded when including particle injections for DPM. Due to concerns about long simulation run times, a mesh refinement was done to find ways of reducing element number while maintaining mesh quality.

#### 3.2.2.1. Original Meshing

The original meshing was a fine mesh with a total count of 891,338 elements. The meshing comprised only of body sizing with face sizing refinement around all major BC (inlets, outlets, PC, heat flux from human and PC models, injection location, walls). The meshing values for the domain are shown in

Table 3-2.

Table 3-2: Original Values for Office Space Mesh Element Sizing

Model Area	Element Size (m)
General Area	0.065
Inlet	0.00933
Outlet	-
Door Slip	0.01
PC	0.03
Person Model	0.025

There was an overshoot in refinement of elements on multiple faces based on the large mesh value, which could be targeted for further improvement. This overshoot was also observed in the tightly packed faces. To analyze the original mesh, parameters including element quality, orthogonal quality, skewness, and aspect ratio were taken into consideration for further improvement. These metrics are known to be the fundamentals of element quality analysis. In general, tetrahedral elements were used as they can offer good quality cells for this application where no complex geometries are present within the dynamic region. Computational time was also reduced with this decision and therefore an iterative mesh optimization approach was taken using the quality mesh metrics discussed below. Different mesh features were implemented to determine if it was viable to replace face sizing without losing noticeable cell shape and quality.

When analyzing element quality, a fine mesh with an average quality of 0.848 is typically desirable, but not optimal if running a large set of simulations for data collection, where shorter core-hours are more practical. When considering that the geometry of the environment is particularly simple, open areas in the domain will not require tightly packed cells. The office space in general is comprised of a person (where particle injection begins from), computer PC (heat source), desk, velocity inlet and pressure outlet sources. Despite the simple nature of the room and its components, there was an emphasis placed on the body element sizing which increased simulation run times. There were also a handful of elements that were critically low in quality where geometry issues were present, namely near the injection point (head), which were seen in the form of sharp angles and small edges. Clusters of elements that changed in size dramatically within a small distance, where gradual smoothening may have provided fewer element quality issues. This is a potential area of future refinement and was not investigated further in the scope of this study.

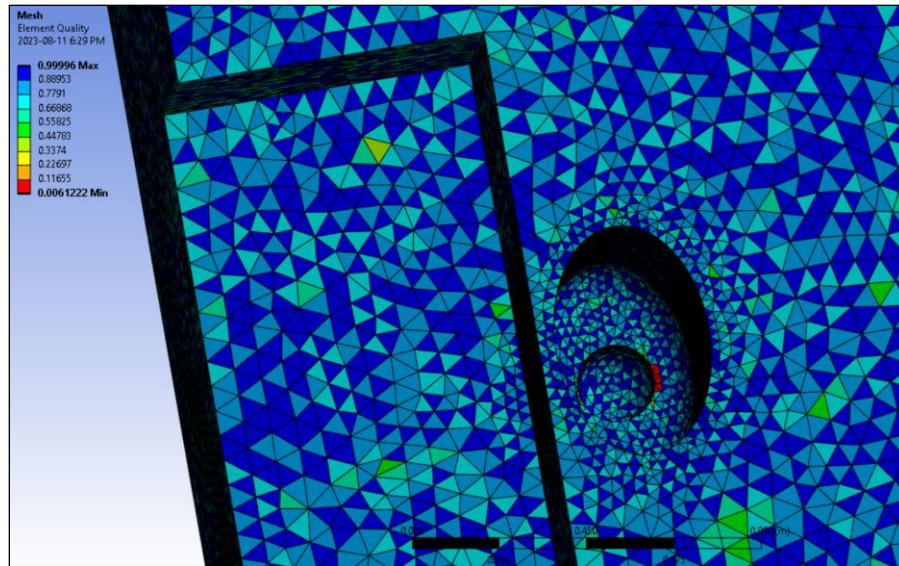


Figure 3-3: Element Quality for Original Mesh– Human Model

Skewness evaluates the extent an edge and angle of a created mesh deviates from the desired or ideal state. To prevent degenerative cells and a good shape of the tetrahedral elements, an excellent skewness range is known to be 0 – 0.25. The original mesh had roughly 200,000 elements with a skewness range of 0.35 – 0.95 concentrated around the human body specifically. This rose a concern with the mesh and geometry of the environment features as well. A large aspect ratio is acceptable where little to no strong transverse gradient (*i.e.*, boundary layer) is present, preference was still given to create as many of the mesh elements in an ideal shape. The average aspect ratio of the original mesh was found to be 1.8, meaning that average cell was stretched (elongated) along one direction compared to its other dimensions. A high aspect ratio among elements in a CFD setup can lead to inaccurate simulation results and cause numerical errors, fail to capture dynamics near boundary layer areas and affect the visualization of the overall result (due to distorted cells and flow patterns).

Orthogonal quality measures how well mesh facing are perpendicular to the flow direction at a given location. Effectively, its value quantifies the deviation of an element from a square shape, a metric where close to one is a common benchmark. This meaning that there will not be any numerical errors where strong gradients or flow direction changes, as well as instabilities in divergence. The original mesh had well over 10,000 elements with an orthogonal quality below 0.5, concerning because they were located at the inlet regions most of all.

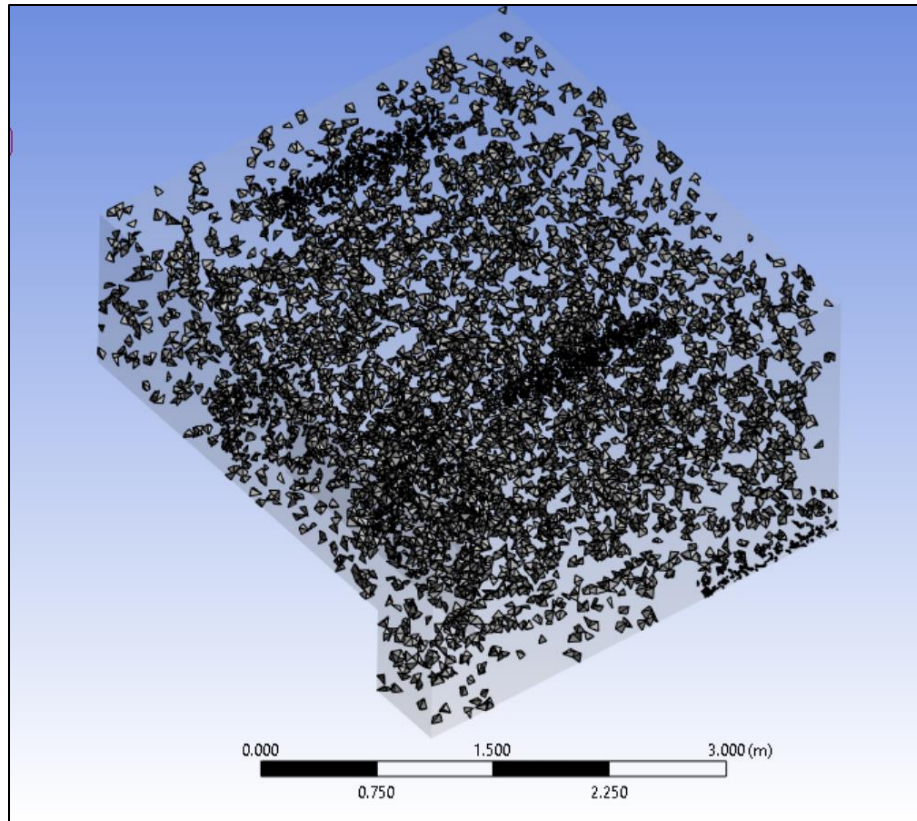


Figure 3-4: Cell Orthogonal Quality for Original Meshing

The skewness and quality of the individual elements did not have major areas for improvement. The orthogonal quality was average except for one of the occupant faces (which is to be expected, due to complex geometry).

#### 3.2.2.2. Meshing Improvements

The overarching goal of this mesh refinement was to reduce the overall element count while retaining the quality of cells. More specifically, the goal was to optimize the meshing further using an approach different than simply assigning a very small element size for critical faces. The total element and node count in general regions while trying to maintain the high element count in important regions (*i.e.*, BCs) would be reduced while trying to reduce the change to meshing quality. The reason for this was to reduce the run time of simulations, as was required for the study performed in Chapter 5. The heavy use of face sizing was eliminated in this new mesh iteration, because of other mesh features that allowed for similar quality traits and a lower computational time. The updated element sizing is shown in Table 3-3.

Table 3-3: New Values for Office Space Mesh Element Sizing

Model Area	Element Size (m)	Refinement Value	Edge Sizing
General Area	0.07	-	-
Inlet	-	3	25 divisions/length
Outlet	0.065	-	-
Door Slip	-	3	-
PC	-	1	-
Person Model	-	2	-

As opposed to redefining the element size for each face and attempting to reduce element count in the general mesh area (outside critical components), the mesh refinement feature for boundary conditions, native to ANSYS was used. Refinement features can specify controls for faces, edges, and vertices on a simpler scale of 1-3 which indicates the factor that the local mesh area is multiplied by for each level. Essentially, this is an exponential increase in refinement that can produce high quality mesh regions, especially since this feature can consider curvature of nearby geometries including boundaries. Therefore, a general body sizing feature controlled the mesh element sizes until each important geometry was assigned a refinement level or edge sizing.

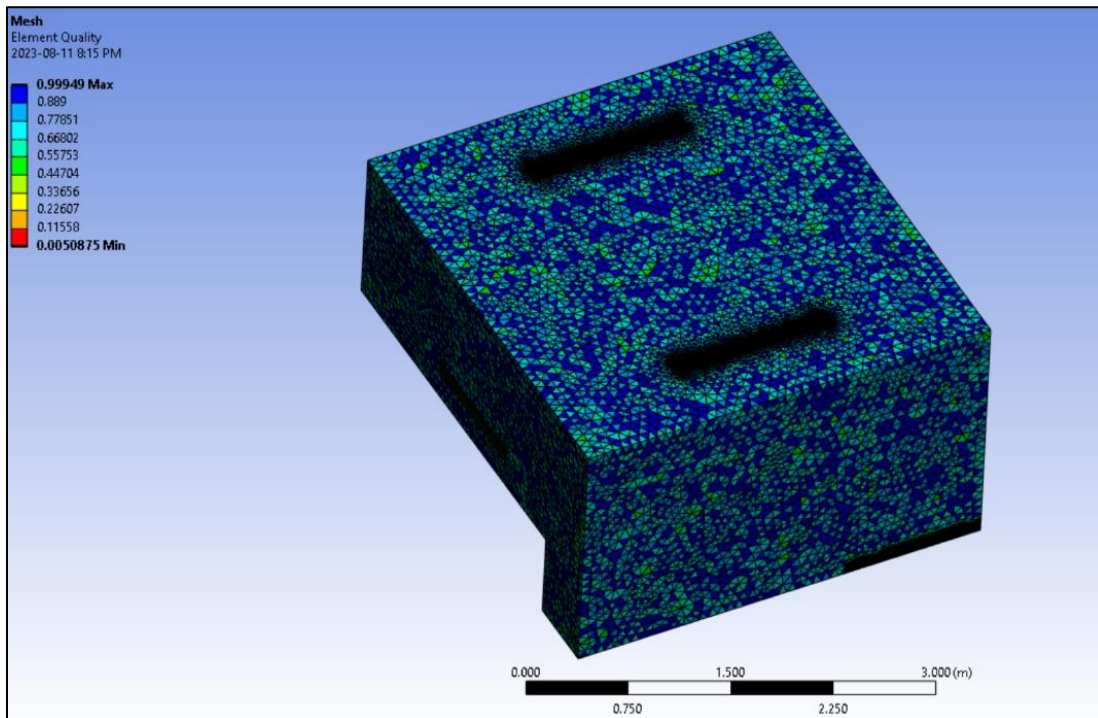


Figure 3-5: Mesh with Refinement Features Replaced Face Sizing

The new mesh prioritized the use of tetrahedral elements since sweeps and multizone meshing were not the most optimal for the room geometry. The lack of complex geometry confirmed that more nodes on each element would not benefit the overall simulation but would greatly increase the computational time. Tetrahedral elements allow for flexibility to handle irregular shapes more easily than hexahedral elements, in this case sharp corners and intricate features such as the various inlet/outlet geometries would be well suited to the tetrahedral element type.

Table 3-4: Mesh Metric Comparison (Initial vs New Mesh)

Mesh	Element Count	Orthogonal Quality	Skewness	Aspect Ratio	Element Quality
Original	891338	0.787	0.211	1.81	0.848
New	622361	0.734	0.262	1.66	0.811

The orthogonal quality of faces was improved to develop more accurate results near the boundary layer areas of the mesh where velocity and temperature values are to be recorded. Since the inlet and outlets are near corners/walls, the mesh needs to be developed particularly well to observe accurate metrics at these locations. The local mesh refinement was efficient at resolving mesh tangling and areas where gradients were observed (near pressure outlet and adjacent walls). Mesh smoothing also increased orthogonal quality issues once the element count had been reduced, sufficiently producing an average element value to that of the original simulation mesh. To maintain an excellent average skewness value for elements in the mesh, a target value was selected and prioritized in the mesh settings. Making sure that the mesh inserts (i.e., sizing, refinement, etc.) had to follow smoothing and adaptive meshing strategies to specifically hold a skewness value of 0.25 automatically generated sufficient faces and edges as a result.

The average element quality of the initial mesh was not an issue to resolve or specifically target, instead retaining critical areas of high quality was important. Velocity inlet, pressure outlets, and other boundary conditions were targeted with high refinement whereas other areas of the body were set with a higher cell size and growth rate. This still allowed for a uniform grid among majority of the simple areas of the mesh, meanwhile high-resolution refinement mesh inserts were placed on the inlet & outlet faces. In addition, edge sizing was added to the inlet velocity areas specifically (set to a division of the length by 25) to emphasize the surrounding mesh area. As a

result, the critical areas retained a high element quality with the general element, and an element quality over 0.8 was achieved.

### 3.3. Numerical Model

For the numerical analysis, ANSYS Fluent © versions 2021 R1 and 2022 R2 were used interchangeably (2021 compatible with 2022, not vice versa). While there were no known major differences between simulations that were run with either version, no in-depth comparison of simulation speed for the different ANSYS versions was conducted. It is noted that simulations run in ANSYS 2021 ran considerably faster, and generally generated better results, especially considering residuals and other values used for convergence. As such, very few simulations were made in 2022 and it was mainly used for plot and graphic generation. The model used in this thesis follows closely with the model used by Khan [11]. A general outline is still provided below with major changes outlined explicitly.

For this study, the flow was analyzed in steady state and converged after 5000 iterations (very slight change in residuals). The real flow time was around 0.1 seconds per 100 iterations. The unsteady particle tracking was done every 10 flow iterations (0.01s) and occurred at 0.001s per particle advancement for 100-time steps, for a total of 0.1s particle time per 0.01s of flow time. Due to this counterbalance of time advancements, the injections were started after significant convergence of the steady flow field. For the purposes of clarity, the difference in time scales will be noted when referenced for values associated with the particle time domain. For steady state flow quantities, the time will be referred to as *flow time* and for unsteady DPM quantities as *particle time*.

#### 3.3.1. Numerical Solver

The model used a Eulerian-Lagrangian to model the fluid flow and particle interaction. The numerical solver was pressure based with RANS. Turbulence was modelled with standard RNG  $k - \varepsilon$  with standard wall functions. The RNG  $k - \varepsilon$  model is useful for analyzing turbulent flows with rotating or recirculatory flows. It is generally used when modeling indoor air simulations. Pressure-velocity coupling was solved using the PRESTO! pressure interpolation scheme. The fluid flow was solved with steady state initialization and was deemed completed by tracking residuals, mass flow rate and temperature probes. The particles were then injected with transient decoupled particle tracking using the ANSYS DPM functionality. The energy equation

was also used to model thermal effects from the PC and human models. Buoyancy was modeled with the Boussinesq model with standard density and thermal values.

### 3.3.2. Boundary and Initial Conditions

The model injects air through the inlets at a specified temperature. Air is released from the domain via two pressure outlets, the door slip, to model air flow under a doorway, and the exhaust. The human and PC models emit a constant heat flux throughout the room. The air diffuser acts as inlets for air into the domain. The volumetric flow rate for all three cases is similar at 57 L/s (0.057 m<sup>3</sup>/s). The area for each inlet is different, however, to account for the room geometry and to allow for better airflow. The walls act as a no slip adiabatic boundary condition. For this simulation all walls besides the human and PC model reflect particles and do not allow for deposition. The PC and human wall boundary trap injected particles, simulating deposition on the surface. The values for each boundary condition are outlined in Table 3-5. Values are truncated for ease of reading.

Table 3-5: Boundary Conditions for Office Space CFD Simulation

Boundary Condition		Value
Ambient temperature (°C)		23
Inlet air temperature (°C)		18
Occupant heat flux (W/m <sup>2</sup> )		64.6
PC heat flux (W/m <sup>2</sup> )		71.4
Diffuser Inlet (m/s) [57 L/s]	MV	0.45
	SV	0.98
	DV	0.58
Door slip		Pressure-outlet, escape
Exhaust (outlet)		Pressure-outlet, escape
Walls (all non-inlets/outlets)		Adiabatic, no slip, reflect/trap

### 3.3.3. Injection Conditions

Particle injection was modeled to simulate low velocity biological injection of particles (such as a cough or sneeze). The particles interacted with the fluid in continuous phase in an unsteady Eulerian- Lagrangian model. The model also included the Thermophoretic force and Saffman [101] lift forces to simulate particle lift effects near walls and due to temperature gradients. While this study does not go further into varying particle injection velocity, this is explored more in Chapter 6. The DPM and injection conditions are outlined in Table 3-5 and Table 3-6, below.

Table 3-6: DPM Conditions for Office Space Simulation

Injection Condition	Value
DPM Iteration Interval	10
Particle Time Step Size	0.001 s
Physical Models	Thermophoretic Force Saffman Lift Force
Parallel Processing Method	Hybrid

Table 3-7: Injection Conditions for Office Space Simulation

Injection Condition	Value
Injection Velocity [mag] (m/s)	6
Injection Position (m)	[2.12, 0.95, 2.17]
Temperature (K)	310
Relative Start and Stop Times ( <i>flow time</i> )	[0, 0.4]
Azimuthal start and stop Angle (deg)	[0, 360.0000521392]
Max and Min Diameter ( $\mu\text{m}$ )	[50, 1]
Mean diameter ( $\mu\text{m}$ )	10
Spread Parameter	3.5
Number of Diameters	10
Number of Streams	2

### 3.3.4. Performance Indicators

The numerical study done by Zhang et al. [9] used Performance Indicators (PI) to develop decision making strategies in selecting different ventilation strategies in different scenarios. The study evaluates mixing (MV), displacement (DV and stratum (SV) ventilation under various office spaces. The study utilized various performance indicators (or metrics) and utilized them in a Z-Score for standardization and ranking. The indicators used include: HRE (heat removal efficiency), CRE (contaminant removal efficiency), ADPI (air diffusion performance index), and a modified index of mixing (IOM\*). The study found that for HRE and IOM\*, displacement performed the best in independent work whereas SV performed the best for discussion and meetings. When all four indicators were factored, SV performed the best in meetings while MV performed the best in all other scenarios.

HRE is an indicator for ventilation systems that quantifies its capacity for heat removal in an isolated system(s). It is the ratio of temperature change between the exhaust and supply to the

temperature difference of the room and supply air. HRE starts at one, and greater values indicate higher removal ability. IOM is representative of the energy loss of the ventilation system. The original equation was designed for use in evaluating heat removal in data processing equipment. The equation measures the effect of mixing by using max and minimum temperatures in local zones. The effect of mixing will usually mean that the room is heating due to the distributed thermal energy. This implies an energy loss as the ventilation is not carrying warm air away from the Cooling zone. The modified equation is used in this study, which uses.

An experimental study by Kabanshi et al. [100] attempts to quantify an occupant's thermal perception. The study states that: "the participants estimated overall thermal sensation (OTS), local thermal sensation (LTS), thermal comfort, thermal preference and acceptability as a function of velocity and room air temperature". The framework of the study was with the use of a newly developed intermittent air jet strategy (IAJS) in spaces with high occupancy (like classrooms).

Of the equations used in both studies, the ones chosen were the HRE, OTS and IOM. The three factors cover a wide range of qualitative as well as quantitative metrics useful in evaluating a HVAC system's performance. These equations will be utilized in all three cases, with differences being in temperature point and surface creation. Due to the open-ended nature of the location selection for these indicators, the exact locations are not outlined here. The equations are outlined below in Equations [3-1], [3-2] and [3-3].

$$HRE = \frac{T_e - T_s}{T_r - T_s} \quad [3-1]$$

Where,  $T_e$  is the exhaust air temp,  $T_s$  is the supply air temp, and  $T_r$  is the occupied zone air. This equation has a theoretical minimum value of 1, where the exhaust temperature is equal to the occupied zone air, implying immediate removal of warm air from the zone. This is a positive indicator, which means that higher values are a good sign. It implies that the exhaust air is absorbing much of the heat ( $T_e > T_s$ ) in comparison to the energy difference between the zone and the supply ( $T_s > T_r$ ). If the supply air is as warm or warmer than the zone, it is adding heat to the system (poor performance). In contrast, if it is blasting really cold air into the system, but the system exhaust temperature is still low in comparison to the zone, it means the system is not removing warm air effectively.

$$IOM = \frac{T_r - T_{in}}{T_{out} - T_{in}} \quad [3-2]$$

Where,  $T_r$  is the occupied room temp,  $T_{in}$  is supplying air temp, and  $T_{out}$  is the exhaust air. This indicator starts centred with a theoretical value of 0. It is the ratio difference in room and supply temperature to the difference in exhaust and supply temperature. If this value is high, it implies that the change in room temperature is small relative to how much energy is being inserted into the system. Note that room temperature, implies the average room temperature as opposed to zone/occupied temperature used for HRE. For cooling applications, one would expect this value to be negative as the inlet air will be colder than the output air. The higher the absolute value of IOM is, the better mixing (and thus ventilation) is being performed. IOM and HRE measure different but similar quantities. HRE measure the local heat removal while IOM measure the average heat removal. In some contexts, IOM is a quantity to minimize, if optimizing heating/cooling for example, without wasting energy. In other contexts, it is a good indicator, especially for certain ventilation modes (i.e., Mixing). Equation [3-3] shows the OTS relation.

$$OTS = 0.31 * T - 1.72 * V - 7.15 \quad [3-3]$$

Where,  $T$  is the supply air temperature and  $V$  is the air flow speed 1.1 m from the ground. This  $OTS$  indicator is a linear equation that indicates the discomfort felt by occupants in a ventilated zone. It centres at 0, with 0 means the occupant will be completely comfortable (no discomfort of any kind is felt by the occupant). The  $OTS$  value will become positive or negative depending on if the occupant feels cold or hot, respectively.

### 3.4. Sensitivity Studies

This section of the thesis outlines a brief sensitivity analysis to investigate the influence of mesh resolution and numerical schema on the accuracy and convergence behavior of the CFD simulations. This was done to have a better understanding of solution accuracy and model fidelity. The meshing used for this study employed basic body and face refinement techniques. The numerical solver utilized a Pressure-Velocity coupling scheme. The Spatial Discretization used least squares cell based gradient and PRESTO! for pressure. For both momentum and energy, second order upwind scheme was used. For turbulence, the kinetic energy and dissipation rates

were both calculated using a first order upwind scheme. The simulation was steady state, and thus used a global time step method for temporal discretization (meaning the time step was varied and was software controlled).

### 3.4.1. Mesh Sensitivity Study

A mesh sensitivity study was conducted by varying the general body element sizing as well as the face element sizing for faces related to boundary and initial conditions (*i.e.*, inlets, outlets, human model, computer model). Four meshes (rough, course, fine and very fine) were generated with cells numbers ranging from around 60,000 cells to 600,000 cells. Four probes were setup to evaluate the average room temperature, the occupied zone temperature, the exhaust temperature, and the door slip temperature. The four temperatures probes were monitored at the end of the simulation when convergence was reached, which was typically after 1000-2000 iterations. The reason for the relatively short convergence time was due to the small number of complex geometries and small domain, which reduced the number of Core-Hours needed for simulation completion. While typically time-averaged values are plotted for steady state simulations with complex geometries or turbulent flows, instantaneous values are plotted here to observe and identify areas that may have transient elements or that may need further mesh refinement. The locations of the probes are shown in Figure 3-6.

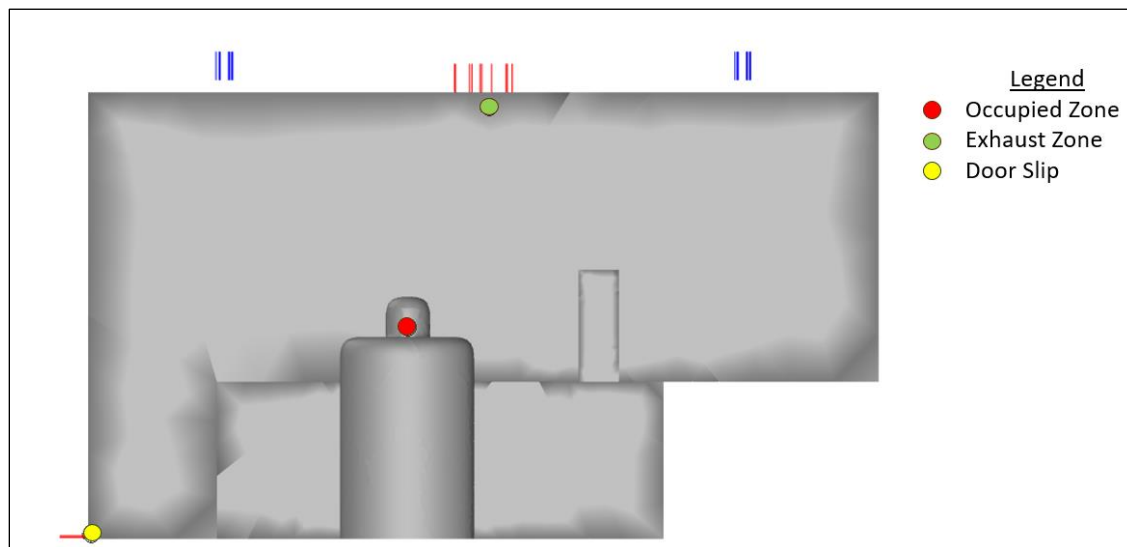


Figure 3-6: Location of Temperature Probes

The results of the refinement study are shown below (Figure 3-7). The results show that the occupied room temperature starts converging very quickly at around 200,000 cells. This was

because this region was very close to the heat flux emitted from the person model and not near any high turbulence regions, meaning there was little effect of turbulence fluctuations. The optimal number of cells was above 400,000 cells for the other three indicators, which were near high velocity regions. These regions had not converged completely, but the discretization error was not significantly decreasing past this point. Further studies may benefit from a larger number of probes, probing differing quantities or using line probes instead of point probes. For the purposes of this study, meshes with 400,000 cells or more was deemed acceptable.

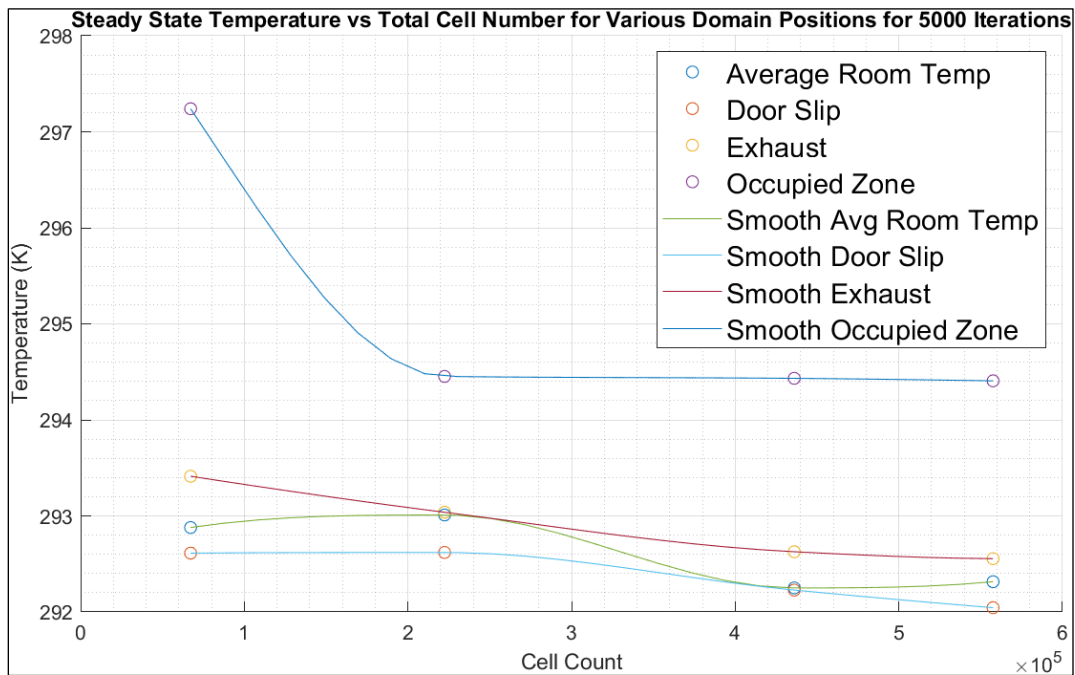


Figure 3-7: Steady State Temperature vs Total Cell Number for Temperature Probes

### 3.4.2. Numerical Study

The ventilation strategies inject high velocity air into the domain. At high Reynolds numbers the effects of turbulence effects are significant. To investigate this, four different simulations were run to analyze the discretization effects of the kinetic energy and dissipation rate terms when using first order or second order upwind schema. The terms are obtained from the RNG  $k-\varepsilon$  turbulence model. The convergence plots of the residuals of the fine mesh are shown in Figure 3-8 and Figure 3-9 for the base case of first order upwind schema for kinetic energy and dissipation rate. The figures show that there was likely a combination of transient flow effects in

addition to turbulence mixing, which resulted in larger numerical fluctuations. The effectiveness of a transient flow simulation was not investigated in this thesis.

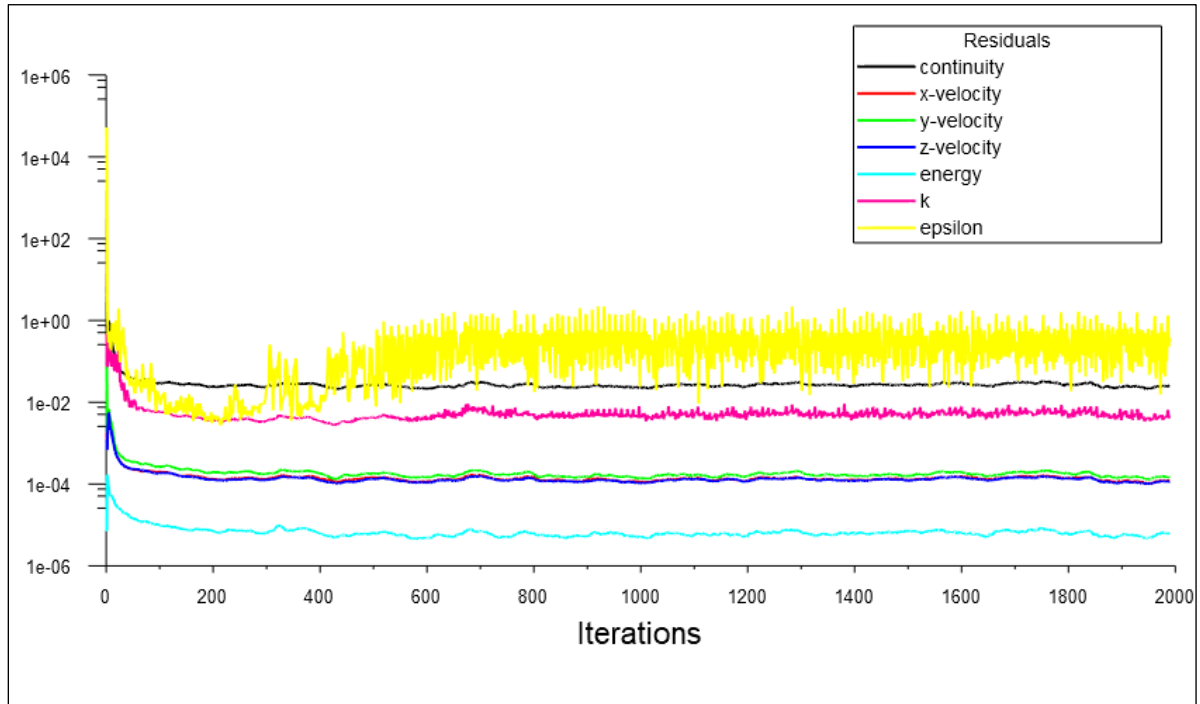


Figure 3-8: Plot of Residual Values for Fine Mesh (400,000 cells)

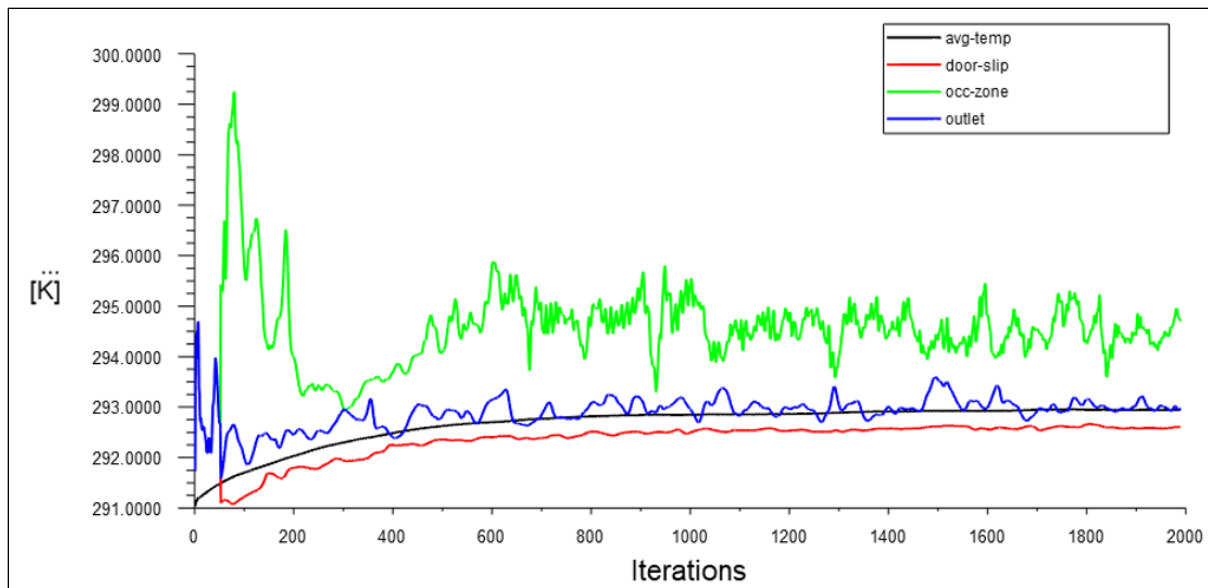


Figure 3-9: Plot of Temperature Probes for First Order Turbulent Term Approximations

In contrast, the results of velocity probes for the enhanced case, where second order upwind was used for the turbulent terms, is shown in Figure 3-10. Higher order approximation of these

terms produced more accurate numerical approximations (reduction in truncation error) and helped to converge the simulation results.

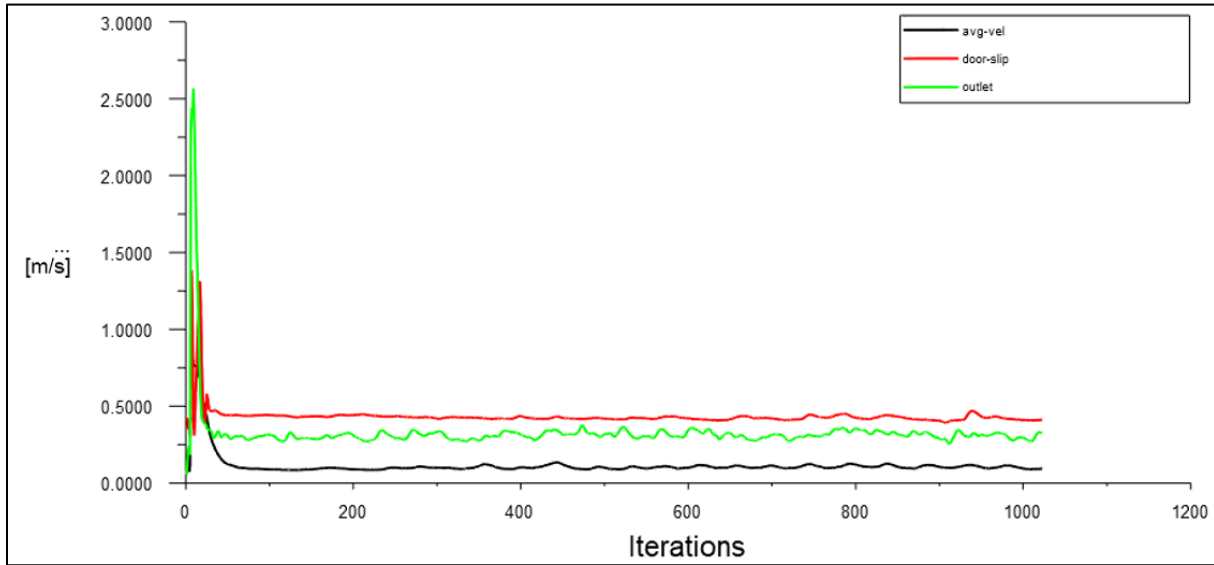


Figure 3-10: : Plot of Velocity Probes for Second Order Turbulent Term Approximations

This numerical sensitivity study compared four simulations, the results of which are presented in Table 3-8 below. The last simulation (Simulation 4), which used numerical schemes with second order upwind for both turbulence schemes, served as the numerical benchmark. Simulation 4 was assumed to be the ideal computational value for comparison and can be used to estimate the change in simulation fidelity due to discretization of the partial differential equations (PDEs).

The error percentage is the absolute percentage difference between the ideal and actual values calculated, where actual refers to the results of the benchmark simulation. The velocity magnitudes are taken from the point probes near the exhaust and the door slip, and the simulation was run for 3000 iterations or until convergence was achieved. The values shown, are time averaged based on values 500 iterations after the simulation has converged. With this analysis, an estimation on the truncation error done due to the reduced order scheme can be approximated.

Table 3-8: Results of Numerical Sensitivity Analysis for Turbulence

Simulation Number	Numerical Scheme		Results			
	<i>Kinetic Energy (k)</i>	<i>Dissipation Rate (<math>\epsilon</math>)</i>	Exhaust Velocity		Door Slip Velocity	
			<i>Value (m/s)</i>	<i>Error (%)</i>	<i>Value (m/s)</i>	<i>Error (%)</i>
1	First Order	First Order	0.3109	9.74	0.4161	2.51
2	First Order	Second Order	0.3005	6.07	0.4366	2.30
3	Second Order	First Order	0.2784	1.73	0.4274	0.14
4	Second Order	Second Order	0.2833	—	0.4268	—

The results show that the two turbulent terms contribute to the discretization error. Particularly, it is observed that the dissipation rate had a much stronger effect on thermal effects, like temperature, while kinetic energy had larger effects on viscous and velocity effects. This was because the kinetic energy term refers to the portion of chaotic kinetic energy associated with turbulent flows. These flows are heavily influenced by large velocities. Thus, areas with lower velocities will see these effects reduced. The dissipation rate deals with the transformation of turbulent kinetic energy into internal energy (heat). This means this term is associated with thermal effects as well as with the dissipation of turbulence. If temperature gradients are not heavily present, this term reflects mainly viscous dissipation.

Because of the relatively low temperature gradients observed in this problem, it is expected that the changes to the kinetic energy term, as opposed to the dissipation rate, reduced the (approximated) discretization error of the simulation. Based on these results, the configuration of the third simulation, with second order upwind kinetic energy and first order upwind dissipation rate, was used for the remainder of this study.

### 3.5. Results

#### 3.5.1. Mixing - Velocity and Temperature

The right-side view of the room and the velocity vectors can be seen in Figure 3-11. The effect of mixing can be through large but slow-moving rotational vortices, mixing the incoming upper cool air and the warmer air. The exhaust can be seen drawing air away from the room into the HVAC duct located at the back of the room.

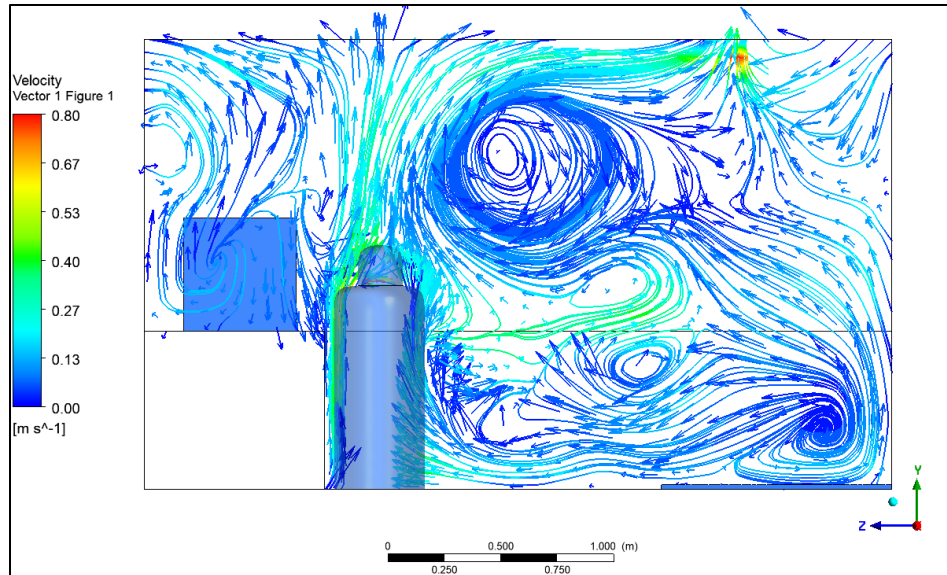


Figure 3-11: Mixing – Right-side View

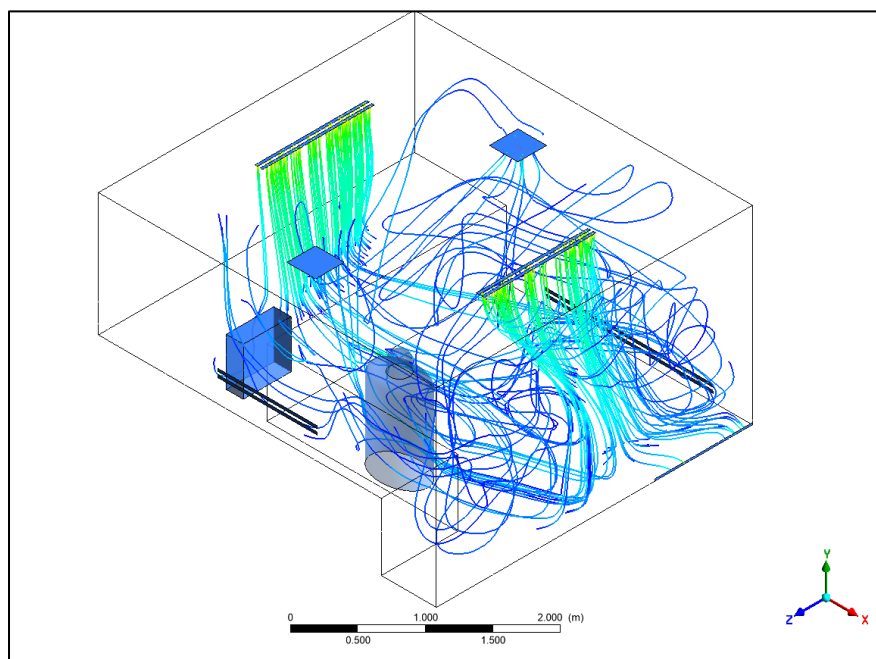


Figure 3-12: Mixing - Isometric View

In Figure 3-13, situated in the midpoint of the room, the rotational vortices are not limited to the longitudinal planes of the room. Transversal vortices can be seen further enhancing the effects of mixing. The high velocity air can also be seen as it enters from above.

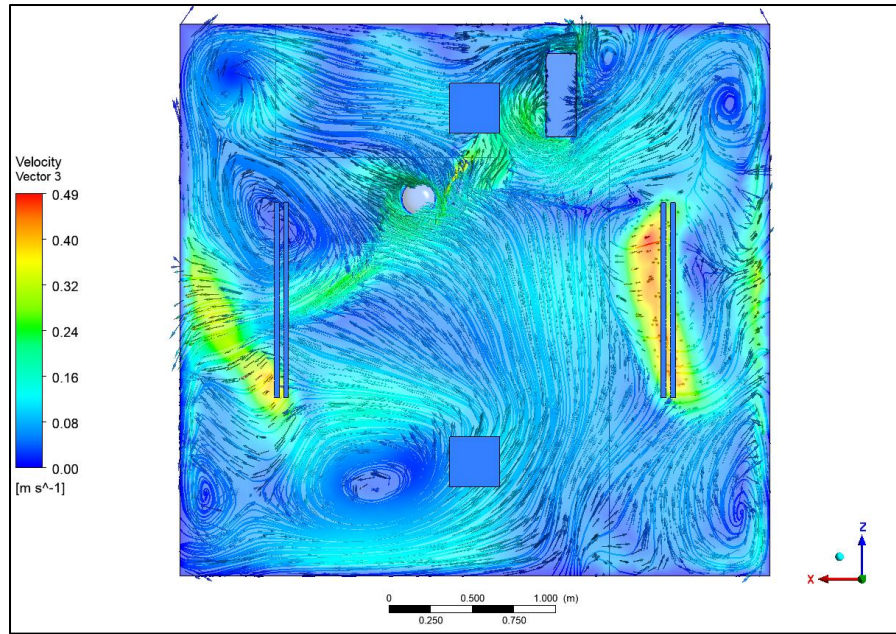


Figure 3-13: Mixing – Top View

As expected, the temperature contour of the room (Figure 3-14) shows a thorough mixing, with warm air plumes moving upward. The upper portion of the room is slightly warmer, than the lower portion of the room, but mixing does a good job of merging the stratified layers of cool and warmer air.

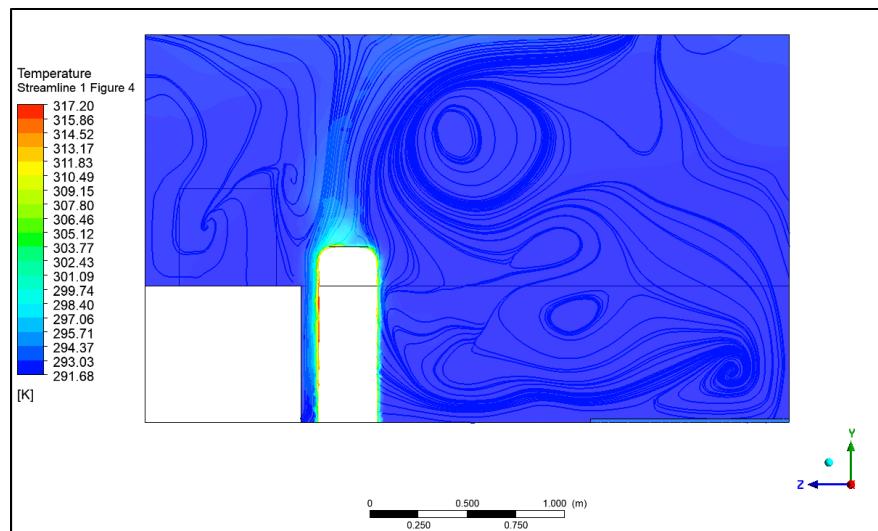


Figure 3-14: Mixing – Temperature Right view

In contrast to mixing (Figure 3-11), the displacement ventilation configuration (Figure 3-15) shows a steadier stream of air moving towards the back to the front of the room. The cool

air, immediately moves upwards, likely due to the combined effects of the low inlet velocity and the effects of buoyancy due to the cooler air. There are some small vortices seen in the back of the room, but the velocities are low. The fast-moving air moves in the upper portion of the room and makes its way to the exhaust. However, there is a small recirculation effect that is due to stagnant slow and warm air around the PC and human model (above the table). This air is then passed by faster, cooler air moving towards the exhaust. The effects of buoyancy and the low pressure of the moving air creates a pressure differential that forces air towards the room centre. There is a small region of large velocities around the room centre, in front of the room. The magnitude of the forces is unexpected, but it is a combination of the incoming air and the warm air plumes generated by the human model. The warmer air wants to move upwards, and this creates a relatively intense recirculation region. This is the result of the meshing quality due to the complex geometry of the region. The mesh may need further refinement to create a better boundary layer and simulate a more realistic airflow.

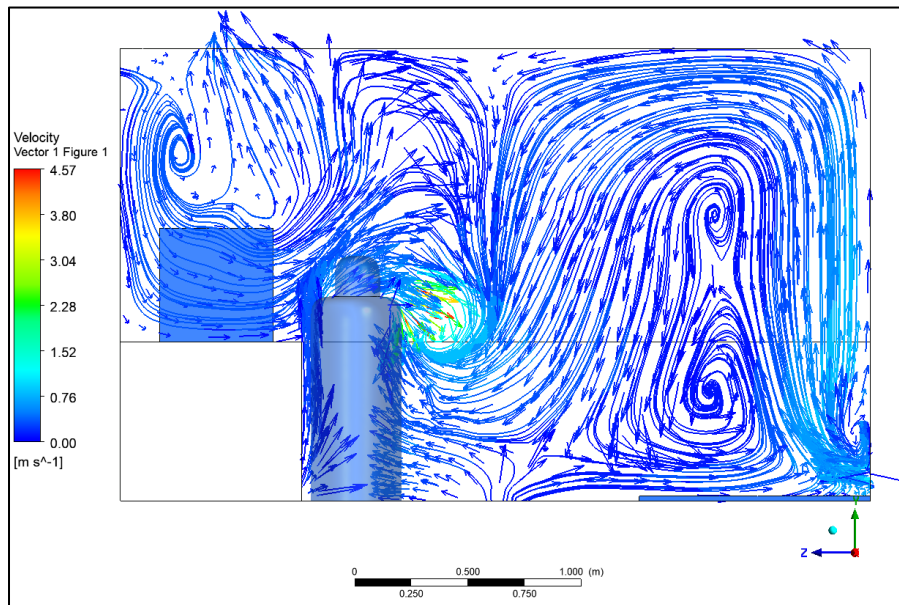


Figure 3-15: Displacement Ventilation – Right View

In Figure 3-16, the top view of the room is shown. It mirrors the previous view (Figure 3-15), with an artifact of large velocity magnitudes near the centre of the room. However, it also shows that the air is moving from the back of the room to front around the room walls, and that the air slows as it nears the front of the room.

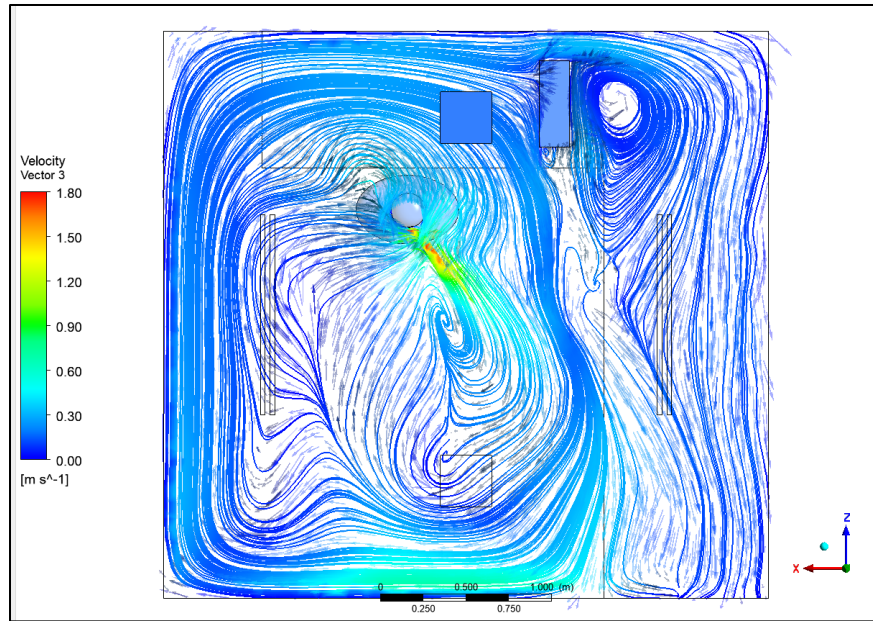


Figure 3-16: Displacement Ventilation – Top View

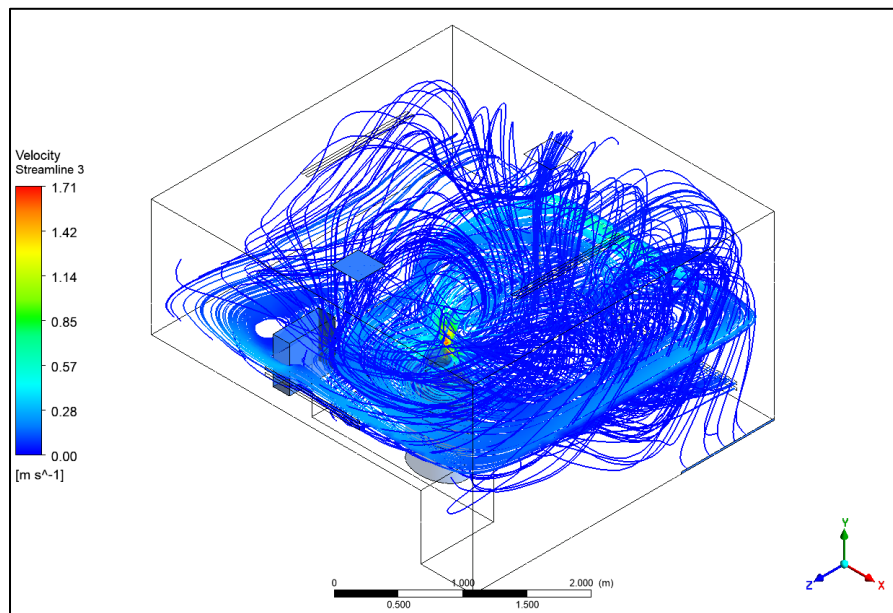


Figure 3-17: Displacement Ventilation – Isometric View

The results of displacement ventilation (Figure 3-15 to Figure 3-18) can be seen in full effect, as it effectively pushes warmer air to the upper and front portions of the room. Regions around the person model, seen in Figure 3-18, are surrounded by cooler air (darker means colder in this context), which will result in good cooling effects, as will be discussed in further detail in a later section.

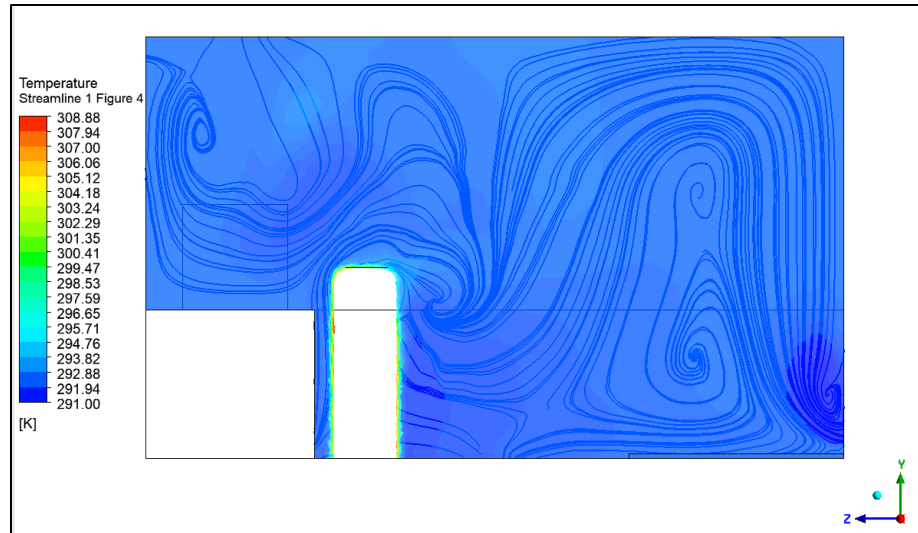


Figure 3-18: Displacement Ventilation – Temperature Profile

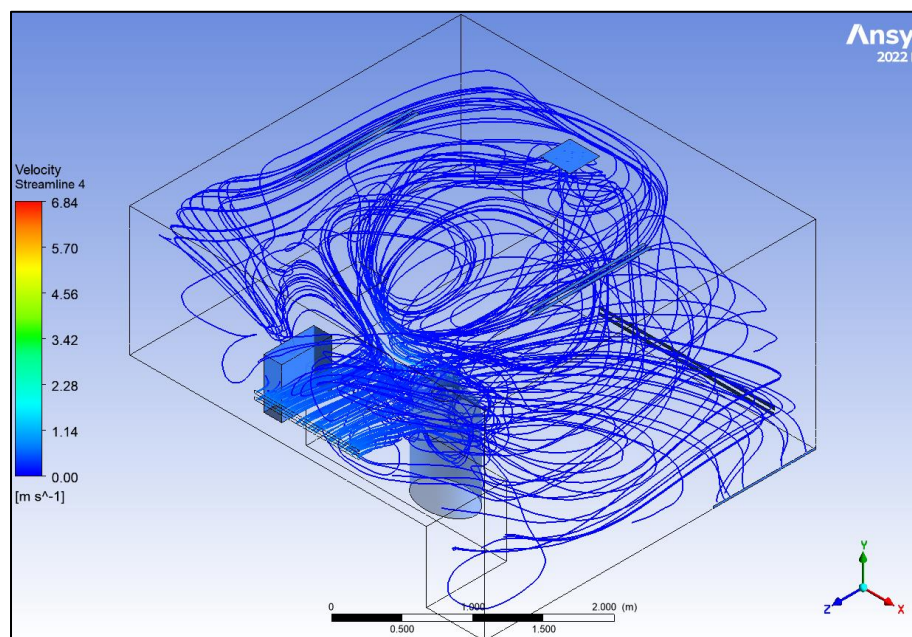


Figure 3-19: Stratum, Isometric View

The Top view of the stratum ventilation configuration, seen in Figure 3-20, shows extremely fast-moving air towards the person model. The air streams are disrupted by the PC and human models, and splits as it warms and slows down rapidly. This split causes the generation of one large recirculation vortices that is likely further induced by the location of the exhaust. Smaller

vortices can be seen along the room corners. The left wall is moving towards the back along the recirculation region while a smaller long form velocity stream, along the right wall, brings air back.

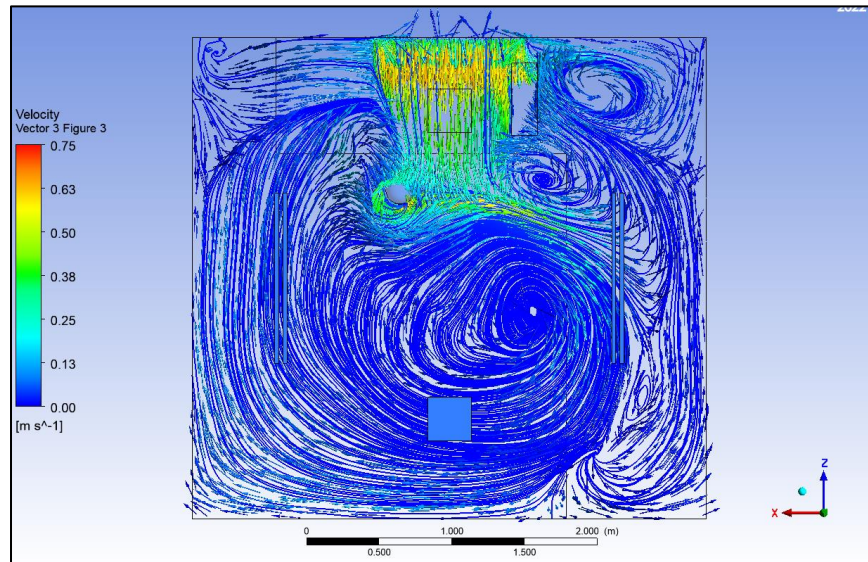


Figure 3-20: Stratum – Top View

The Stratum Ventilation Right View, seen in Figure 3-21, shows two large counter flowing eddies, above the inlet, caused by the high velocity air, which is inducing turbulence in the flow. The air in front of the human model (facing the room centre) is moving downward. This flow removes warm air and pushes it towards the floor and back of the room where a large recirculation region moves it upwards towards the exhaust.

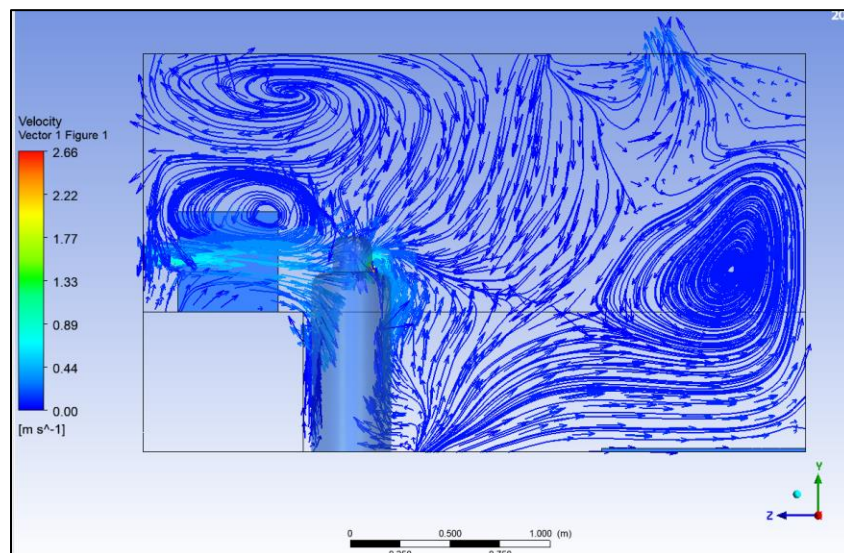


Figure 3-21: Stratum Ventilation Strategy, Right View

The temperature contour of stratum ventilation, seen in Figure 3-22, shows that the warm air is indeed being pushed towards the room centre due to the air flow. However, unlike the other ventilation strategies (configurations), it does not provide uniform cooling across the human model. However, it does a much better job of pushing cooler air forward and warmer air away. The magnitude of which, however, is hard to compare with these graphics alone.

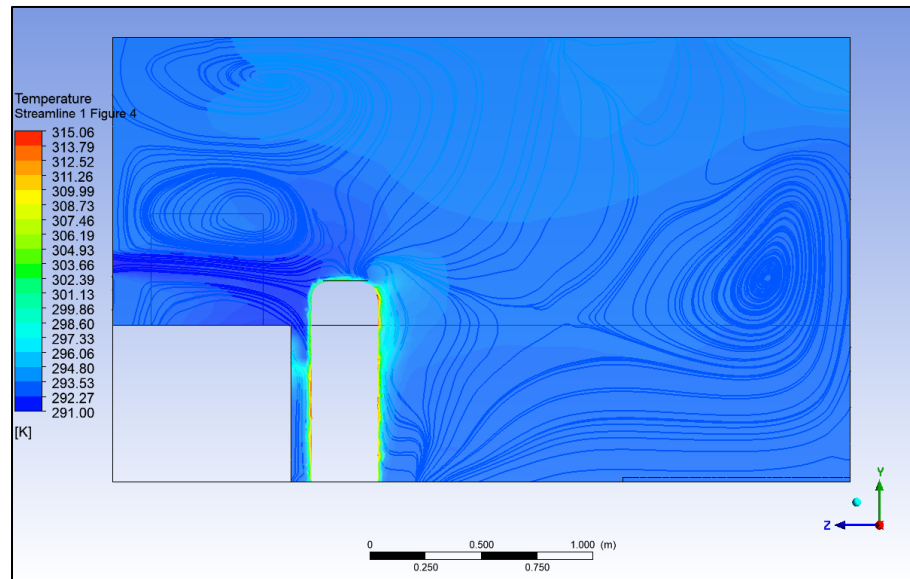


Figure 3-22: Stratum – Temperature Profile

### 3.5.2. Particle Dispersion

The time lapse isometric view of the displacement ventilation can be seen in Figure 3-23. Mixing ventilation studies of an office room were previously conducted by A. Khan [11], a previous Master's student in the Freire-Gormally lab, and the displacement ventilation configuration was not found to be particularly effective for removing particles. The results in this study of the displacement ventilation configuration (Figure 3-18), in comparison, does a much better job of removing particles from the breathing zone. The particles were forced upwards, due to high velocity air moving from the bottom of the room. Thermophoresis effects caused the lighter particles to rise while heavier particles fell as they rapidly cooled. However, most of the light particles were immediately sucked into the exhaust and floated in the upper strata of the room, away from the breathing zone. These results are in agreement with Khan's results [11], in that the presence of ventilation doesn't significantly affect the quantity of particles removed (escaped/trapped in her case). However, the ventilatoin configuration does effect the movement

and groupings of particles short term. This means ventilation plays an even more important role when looking at short term dispersion events (less than 10s).

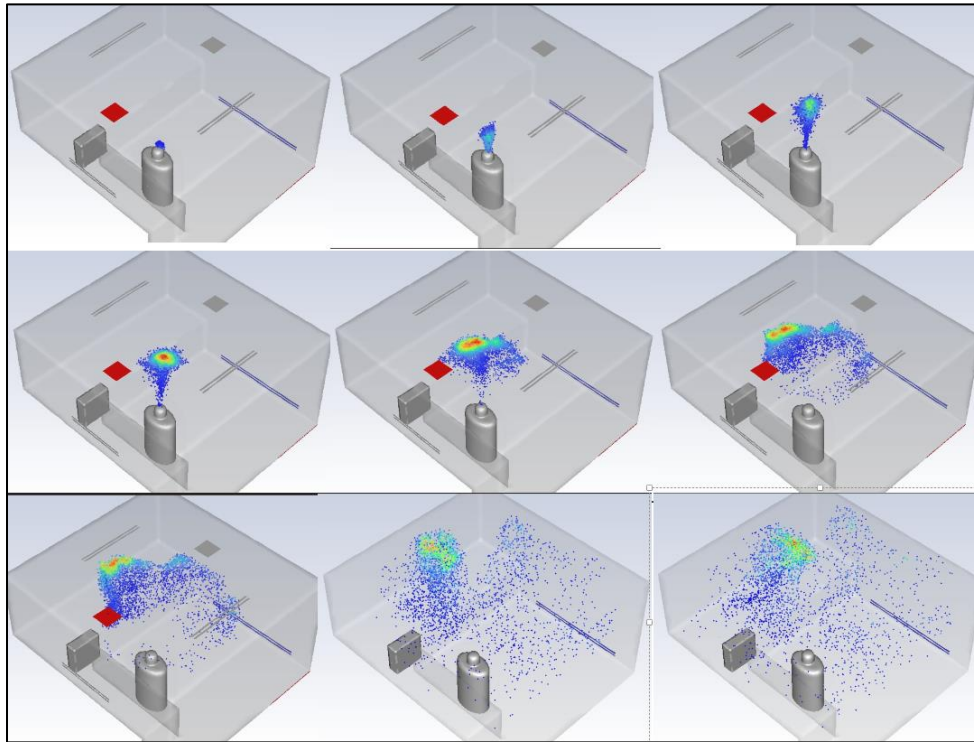


Figure 3-23: Displacement Ventilation - Particle Advancement over 10s particle time

The dispersion of SV is shown in Figure 3-24. While the particles did rise initially, due to that high velocity recirculation zone, the particles encountered the two large counter flow eddies that are in the front of the room. If the particles had moved more towards the centre, they could have entered the large, low velocity flow stream near the ground. Instead, the particles entered these zones and were flung towards the room front before moving higher up. As the effects of diffusion occurred, the particles entered this region and the smaller lower velocity region near the exhaust. Some particles were ejected at this time. However, more particles were spread towards different areas of the room. The velocities in these regions near the exhaust were slow, taking longer to remove the particles.

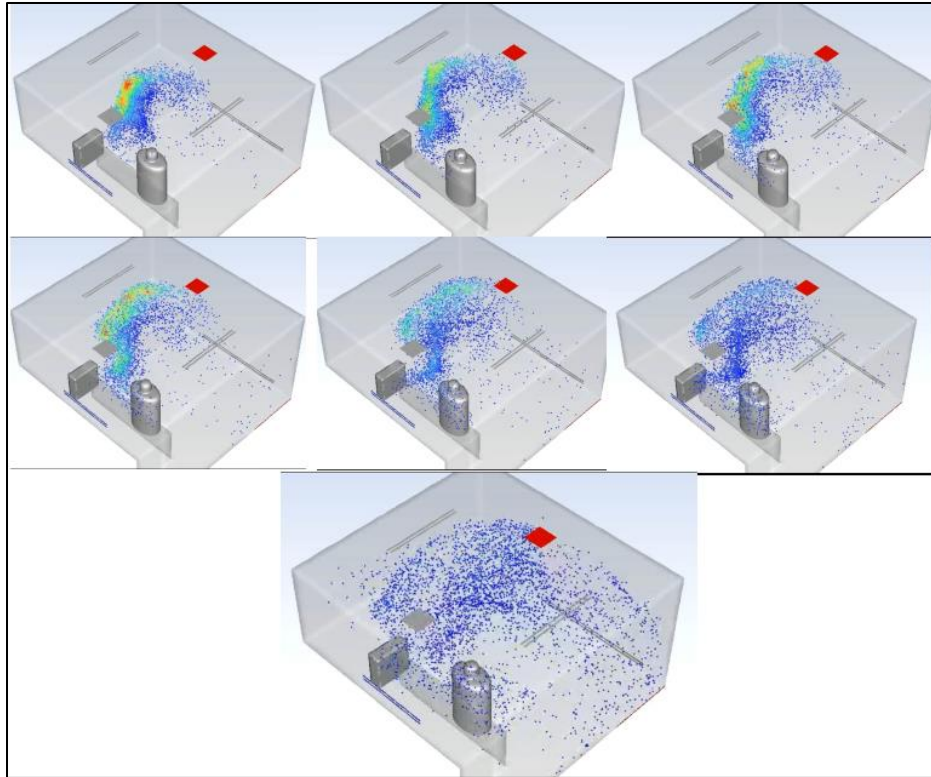


Figure 3-24: Stratum - Displacement - Particle Advancement over 10 s particle time

### 3.5.3. Performance Metrics

The results in Table 3-9 show some values that add more context for the simulation results. Of note, MV and SV had the lowest average temperature and occupied zone (around the human model) than DV. This is likely a result of the very low velocities input by DV, which is also reflected in its exhaust velocity.

Table 3-9: Noted System Values for Ventilation Strategies after 80 s Flow Time

Ventilation Strategy	Average Room Temp. (°C)	Average Occupied Zone Temp. (°C)	Average Occupied Zone Velocity (m/s)	Average Exhaust Temperature (°C)	Average Exhaust Velocity (m/s)
Mixing	19.16	19.18	0.0789	19.34	0.4330
Displacement	28.07	27.17	0.1053	29.88	0.0317
Stratum	19.78	19.49	0.1403	20.26	0.5657

The results of the performance indicators are shown in Table 3-10. SV and DV performed well for HRE, DM performed the best for the IOM and the results varied for the OTS, but occupants would be the least uncomfortable in SV compared to the other ventilation strategies. It is interesting to note that while DV performed well for HRE, meaning it removed heat efficiently, as shown by the steady state average values, it did not keep the room very cool, in comparison to the other configurations.

Table 3-10: Performance Indicators for each Ventilation Strategy after 80 s Flow Time

Ventilation Strategy	HRE	IOM	OTS
General Meaning	Above 1, positive indicator	Above 0, Negative indicator	Neutral indicator, 0 is perfect. <small>Changes based on discomfort type</small>
Mixing	0.82327993	-9.8556746	-1.2523673
Displacement	1.290746	-0.51113092	1.22505267
Stratum	1.1854424	-2.4548295	-1.036891

Table 3-11, seen below, displays the injected, escaped, and trapped number of particles after ten seconds of simulation time. This study ignored tracking deposition effects (trapped particles) to focus investigations on the number of escaped particles. To this end, the PC and person model DPM wall interactions were changed to reflect. It was found that mixing and displacement performed reasonably well.

Table 3-11: Quantities of Particles of different after 10s Particle Time

Ventilation Strategy	Injected	Escaped	Remaining
Mixing	4844	2682	2543
Displacement		2424	2420
Stratum		1688	3156

## 3.6. Discussion

### 3.6.1. Particle Dispersion

When it came to the movement of particles, SV performed the best early in the simulation. However, after the initial escape of particles, Stratum began to rapidly spread and fling the particles away from the room. In contrast, both mixing ventilation and displacement ventilation performed well in removing particles, both removed almost half of the particles, gradually throughout the course of the simulation. The possible reasons for this are varied, but two likely candidate

explanations are the room geometry and the nature of the stratum inlet velocity stream in contrast to mixing and displacement. On average, stratum removed less than 800 particles when compared to the other strategies.

The room geometry, as noted early, is unique since the exhaust for all three cases were located at the top later (ceiling) of the domain. This was done to stay faithful to the present construction and design limitations that the simulation domain and geometry is based on. The issue is that stratum usually sees exhaust in the same layer or lower than the inlet stream. In other words, both faces were parallel with one another, inducing a flow stream in the breathing zone. This will cause particles to flow along the breathing zone away from occupants and into the exhaust before reaching the upper or lower strata, where it can hang for mins to hours afterwards [5,19,56,70] or be deposited on surfaces [5]. In this case, the faces of the exhaust and inlet were perpendicular. As shown in the contour/vector plots in the results section, this combined with the high velocities, incurred horizontal vortices flinging the airborne particulates (heavy particles still dropped) before many could escape into the exhaust. The inlet and exhaust were near each other, meaning flow streams in other parts of the room were minimal and caused the particles to lack the ability to enter flow streams near the exhaust.

Stratum ventilation works only by injecting air from the midstream (typically) of the room. This means that for small and medium rooms, that have low roofs, that the flows tend to split and cause twin opposing recirculation vortices. This works to effectively mix particles in the room. In mixing ventilation, the inlets blast cool air vertically. However, it also has an exhaust at the ceiling. This flow instead creates areas of vertical vortices, that can cause air to move upwards towards the exhaust. Thermal buoyancy effects help to raise both the air and the particles upward, as the warmer air rises. Displacement does this to a higher degree, have cool air blasted on the floor which will create flow streams upwards, which helps to explain the lower particle concentration. The air is blown upwards at a constant but low velocity. This reduces turbulent mixing effects and keeps the particle suspended generally on the upper strata of the room for a considerable time before diffusion effects start to spread the particles further.

An additional note is that this study only analyzed particles after ~10 s of particle time (which can be taken as real time for the purposed of this study). This means that it is possible that further simulation run time would see different results if it was run on the scale of minutes rather than seconds.

### 3.6.2. Performance Indicators

When analyzing HRE only, DV performed the best. Displacement ventilation works with buoyancy forces with forced convection to move warm air upwards while injecting cool air in the lower portions of the room. This, in turn, induced a constant stream of cool air in the lower region moving upwards. This air is then removed from the room, making DV an efficient tool for targeted cooling in this case. SV was a close second, so it is hard to say if the results are due to simulation residuals. SV performed well due to presence of turbulent rotational velocities around the occupant due to high velocity inlet air, that quickly take warm air away from the occupant. Warmer air will then flow towards the exhaust. Velocity views from the XZ plane (Top view) confirm this, with minimal rotational vortices, at least at the midpoint of the breathing zone. There are still vortices in the corners of the room, but that is to be expected for all cases. The major difference is the location of the inlet and the inlet velocities. Stratum injects high velocity air in the breathing zone while displacement injects low to medium velocity air at the lower portion of the room. This phenomenon of air stratification is useful for cooling as well as removing air contaminants, as was found in the previous section. Since it aids in air movement, it is useful for increasing energy efficiency of HVAC systems. Mixing ventilation had the worst performance as high velocity air is injected downwards. By relying on turbulence and diffusion effects, cool air is mixed with the warm air. However, uniform mixing is not energy efficient since the goal is typically to cool occupants directly. However, it should be noted that mixing does not deviate significantly in performance regarding thermal energy removal. This study only analyzed specific exhaust placements on the room ceiling. Mixing HRE would perform better if placed on the lower strata of the room, where the flow velocities and induced negative pressures (not visible due to steady state condition) could quickly remove the incoming air.

When looking at the IOM, MV performed the worst. As a reminder, the IOM (index of mixing) is a measure of how well a ventilation system reduces the amount of (thermal) mixing, as the goal is to cool the room and mixing will store thermal energy into the overall room, raising its average temperature. Because of this, lower scores are worse. As implied by its name, mixing ventilation encourages thermal mixing, since it is mainly used in larger rooms, or where occupants are spaced widely. Stratum performed poorly but not to the same extent. This is likely because the inlet was in the center strata of the of room, encouraging some mixing effects. It can be noted that while IOM and HRE are closely linked, they don't measure the same effects. Heat removal

measures the difference in the injected air and exhaust air temperatures to the supply and *supply zone* temperature, making an inference on the efficiency of thermal energy being removed from a specific zone. In contrast, IOM looks at mixing effects in the entire room, not just the occupied zone. Displacement performed the best, likely due to reasons discussed for its HRE performance. Forced convection, buoyancy forces, and lower velocities (which limit mixing due to turbulence effects) contribute to the efficient removal of warmer air and an overall lower room temperature.

OTS stratum performs notably better, and still has good HRE ratings. The reason for this, especially the OTS, is due to the position of the inlet vent in the room. However, this caused much of the air to miss the exhaust and raise the room temperature (lowering its HRE value). It is interesting that the IOM is higher considering that it is usually higher in MV. This is likely just a result of the specific room orientation.

One thing to note is that this specific iteration of SV, is more akin to a raised DV design. This raises some concerns on classification of the design as stratum, although as a loose definition it just requires the inlet in the breathing zone of the occupant(s). This makes it difficult to assess if the good results are due to aspects of stratification due to DV or SV (or both). Another thing to note is that it is difficult to compare the relative values of the indicators themselves without some reference. A very small decrease in HRE for example could constitute a large deviation in when compared to more nominal cases. Most of literature agrees that mixing performs the best generally when considering many factors (including implementation, use in heating applications, use in larger rooms, etc.) and that displacement and stratum generally perform well for niche cases. This is also the case for rooms with small amounts of occupants [11,12].

### 3.7. Conclusions

#### 3.7.1. Summary

This study found that regarding HRE that SV performed the best, with a value of 1.19 and MV performed the least with a value of 0.82. This stems from high velocity air blowing directly in front of the occupied zone which combined with buoyancy effects, helps stratify air. This brings warm air to the top layers where is expelled from the room. In contrast, MV is lower due to turbulent vortices which helps to mix thermal layers to achieve a uniform thermal distribution. MV did however have the best IOM with a value of -9.86. Displacement performed the worst with a value of -0.51. As stated previously, turbulent mixing of thermal layers made MV very efficient in

this regard. Displacement on the other hand, relies more on low velocities. As such natural convection due to buoyancy forces help more gentler (sometimes) laminar flows upwards in the room. The experimental factor OTS had MV and SV produce cold discomfort with values of -1.25 and -1.03 respectively and DV a warm discomfort with a value of 1.23. Because DV relies mainly on more on natural convection than forced convection, it has both low velocities and low average room temperatures, leading to an occupant likely to feel warmer (for cooling cases). In contrast, SV relies on high velocity air directly in the breathing zone and MV relies on through mixing of the cooling domain, resulting in lower average temperatures in both cases, which contributed to the cold discomfort. SV performed the best with the value closest to 0 (ideal case).

When looking at the strategies in the presence of airborne droplets, it was found that stratum performed poorly when compared to mixing and displacement when looking at overall qualitative flows, and in the ability to remove airborne particulates. Of a total of 4844 particles, MV, DV and SV removed a total of 2682, 2424 and 1688 respectively. This is likely due to the injection position as well as the positioning of the inlet relative to the exhaust. Implementation of stratum ventilation must pay careful attention to the configuration of these to ensure proper removal of particles, which may limit its practical application in the wider HVAC industry. Stratum ventilation requires positioning in the breathing zone, which can limit its practical use in larger rooms or rooms with complex geometries. Analysis of a more limited time window (simulation start and end times) will play a role in the specific results obtained.

### 3.7.2. Final Thoughts

Ventilation strategies can have variation and cross over in design and this study demonstrates the importance of good design when considering more experimental design orientations. However, real situations often include more complex elements such as heating and cooling, adjustment/monitoring systems, grill design/shape and others. This study has found that stratum can be an acceptable strategy when compared to more commonly used strategies of mixing and displacement, but that its effects are minimal and require careful considerations of geometric positioning. Performance analysis (relating to comfort) is highly sensitive and contextual and often equations like the OTS and similar can be difficult to use or compare with more strict indicators like the HRE. However, they are useful to distinguish how occupants in a room will ‘experience’ the HVAC system that is present. Regardless of these limitations, stratum does show promising

signs of having good, and sometimes better performance when compared to conventional ventilation strategies. However, it is limited due to the precedent in HVAC companies design process, practical limitations, and a lack of experimental testing in countries with colder climates (like Canada) where heating is more practical.

### 3.8. Future Work

Future work will consider more complex scenarios: including various inlet and outlet locations, varying initial conditions, longer simulation time durations, shorter time steps or multiple vents' slits (*i.e.*, MV case). In addition, a wider range of performance indicators can be used to establish a wide range of performance metrics. These indicators can be categorized together for better classifications between the ventilation's strategies. The strategies themselves can also be compared with different configurations, allowing for a single optimal configuration, or for different optimal modes, depending on the required criteria.

## Chapter 4. Development of Novel Method to Aid in Particle Dispersion Analysis using Parallel Processing and HPC

### 4.1. Introduction

When using tools like CFD tools like ANSYS, it is often difficult to extract or format data in the way that is useful or not repetitive. Additionally, it often becomes harder to tweak models in ways that simplify or add complexity without thorough knowledge of theoretical or mathematical models in addition to the specific numerical models used or available in ANSYS (Fluent). However, the software comes with many tools that allow users to enhance the usability of their simulation. Some of these, specific to ANSYS, include things like Journals or SCHEME. Which are systematic commands used to perform scripting or other specific tasks. Additionally, it also contains a text user interface (TUI) in addition to its GUI that can also be used by its scripting commands. However, these generally are used for high level simulation tasks, such as saving files, looping iterative processes etc. To interact directly with model values, cells or even insert custom models, a UDF (User Defined Function) is required. An UDF is a dynamically loaded function (s) that enhance standard features of ANSYS code. They can compile in C/C++ (generally) and include custom defined Macros. Other tools that ANSYS provides, which is more common when running larger sized or number of numerical models, is parallel processing. It allows multiple CPUs, GPUs or even computers to solve simulations simultaneously locally or over a network. This study combines elements of UDFs and parallel processing in addition to other techniques. The goal is to help aid in efficient gathering of dispersion data that cannot normally be obtained via the conventional use of ANSYS Fluent. For the purposes of this study, the process or tool will be referred to as the Cell Based Particle Tracker (or CBPT). In addition to being compatible with ANSYS, this tool is also functionally able to run using the online computational resources on the Advanced Research Computing platform (a service provided by the Digital Research Alliance of Canada) [102,103]. This study will detail the development and implementation of this tool to gather unique CFD data relating to particle dispersion. While it will include some elements of cluster and UDF utilization, it is not intended as instructional material for using UDF, ANSYS or HPC cluster nodes.

The major goal for this study is to create a tool or workflow that utilizes CFD, UDF, parallel computing and HPCs. This tool will be used in further studies and can also serve as a guide or starting point for other researchers. The subgoals are:

- Create a UDF that can extract particle data from CFD simulations.
- Have the UDF be compatible with parallel processing modes in ANSYS Fluent
- Create a process flow to run CFDs that contain UDFs on a cluster that utilizes HPC.

## 4.2. Architecture Methodology

The CBPT comprises of three processes: Setup, Data-Processing and Data-Storage. The *Setup* involves two minor processes required to use the UDF: ANSYS *Loading* (including Compilation) and *HPC-Utilization*. The *Loading* stage, which is operating system and compiler specific, transforms the UDF by first compiling it into code that is readable by ANSYS Fluent and its relevant systems. Then it *loads* the UDF into the relevant ANSYS (in this case Fluent) Models or events (called hooks). This allows for the customized ability to change various system and model parameters, like properties, time steps, boundary or initial conditions or even meshing conditions. This is done by *hooking* the UDF into the models in question or by assigning an intermediate variable for ANSYS/Model to communicate with the UDF. This is further explained in Section 4.5. *HPC-Utilization* (High-Performance Computing) refers to the process of using specialized computing systems for data-intensive problems. In this case and many other applications, it involves the use of parallel processing to accelerate computational performance. Compute Canada, and its related organizations, have clusters of HPCs available to Canadian researchers. Such cluster(s) were utilized to enhance the process, although the UDF can also be run without the use of parallel processing, as will be discussed.

*Data-Processing* occurs in the UDF and involves the gathering of Fluent variables, calculation and transformation of required output data and sending of this data (if required) back to ANSYS or to other sections of code for Data Storage. *Data-Storage* is simply the process of saving the data locally on the node (or core) that is running it. This typically can be done by either ANSYS or by the UDF. This section outlines the development and implementation of the processes surrounding the CBPT. It will not go over the specific processes done in the UDF portion of the CBPT, although it will be referenced. The entire process architecture is outlined below.

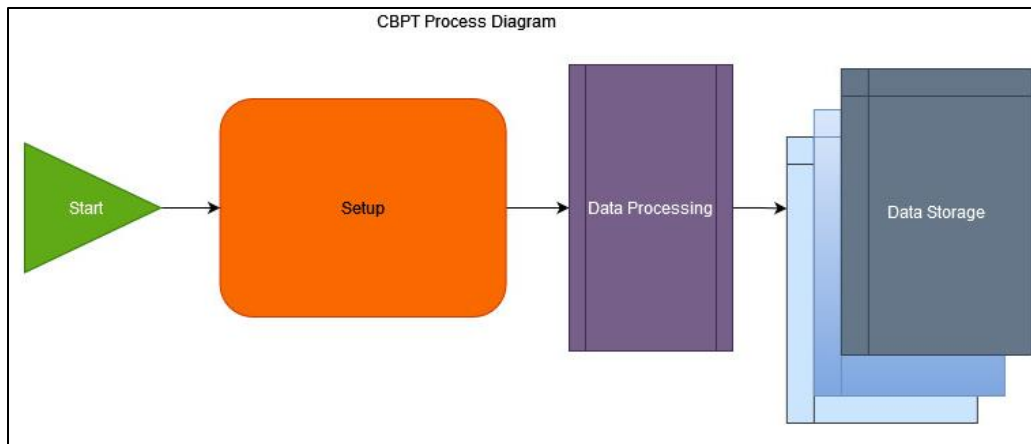


Figure 4-1: CBPT Overall Process Diagram

### 4.3. Getting Started with Parallel Computing

When utilizing parallel computing, knowledge of the structure of ANSYS node architecture is paramount. ANSYS node architecture consists of a *Host node*, *node-zero*, and the support nodes (or compute nodes). The *host-node*, known as cortex, communicates only with *node-zero*. It does not *see* (hold or have access to) most computational data. The *host-node* is responsible for data writing and passing high level commands to *node-zero* who in turn will relay it to the other nodes. Node-zero can communicate with all nodes and can also see data stored. It also does computations and has its own mesh partition. The other nodes are strictly computational nodes, and they typically don't write to files. When passing commands to ANSYS in parallel, these commands can all be seen by all nodes. This means the commands must be filtered so only the required nodes can see them. Each node has access to unique memory and information. To access them, the memory (or memory addresses) need to be passed to *node-zero* who will in turn, pass it to the *host-node* for general computation. This is completely case dependant and there are cases where no data needs to be passed. Figure 4-2 shows a graphical representation of the node architecture.

To utilize CBPT, a basic understanding of how serial processes run as compared to parallel processes is required, since logic and ideas that would typically work in serial processing, will not necessarily run in parallel processing, or vice versa. To help with this, ANSYS has custom state variables that can be used to determine the state of the system. These statements are conditional, meaning code can be written that will only execute for a computational node, or only for cortex (host-node).

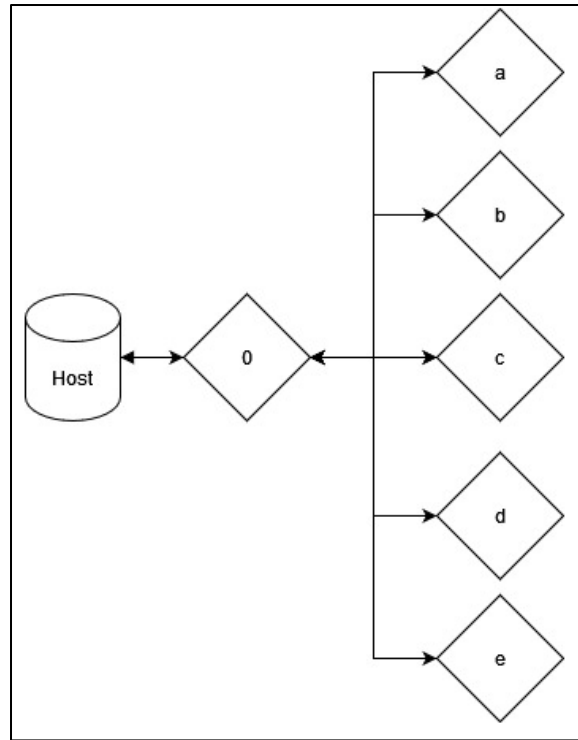


Figure 4-2: Node Architecture for ANSYS Parallel Computations [104]

## 4.4. Setup

### 4.4.1. HPC Initialization and Utilization

The CBPT is designed to operate on the Compute Canada GRAHAM cluster. To submit a job, simply use `sbatch`, for batch jobs, `srun`, for a single run, or `salloc`, to allocate memory and cores and run jobs using a VNC or `srun`. A special script file is needed in most cases for running batch jobs. Jobs can also be specified directly in the command prompt, similarly to how `salloc` allocates memories for jobs. The user can simply submit the memory, number of CPUS or GPUS, number of tasks (if using more nodes) and number of nodes, time of the allocation, and more options outlined in the technical documentation of the Digital Research Alliance of Canada [105]. The *module* needed to run the job must also be specified. *Module* refers to the program that the user wishes to run. Once a job or memory is requested, the user must wait for that memory to be allocated. The wait time depends on several factors, with the most important factors being the length of time and number of resources requested [105].

When the requested memory becomes available, the job is run, hence, the job will begin, or the user can simply start ANSYS Fluent, and the job will commence. If a job runs out

before it is finished, or the file is not saved in any way, the files will be deleted, unless autosave in ANSYS was enabled. When submitting an ANSYS Workbench file, a shell file and a journal file is needed. The journal file outlines to ANSYS what tasks to perform and what to save, amongst other tasks. A basic journal file is included that can be easily changed to meet various needs. Any errors in the journal, WB file, shell file need to be resolved prior to submitting it to ANSYS or it will likely result in a server disconnect (kicking from terminal or node).

#### 4.4.2. Loading ANSYS and UDF

If the submission process has no errors, either in the shell file, journal file or WB/Dat file (such as different versions), then ANSYS will load and follow the instructions outlined in the journal file and UDF. In terms of ANSYS, this means that the simulation will now be loaded, either through the ANSYS Workbench or directly through ANSYS Fluent, depending on how the shell file was configured. The CBPT has shell (bash) files for both options.

Compiling the UDF should be seamless, but it may not always be. Compiling fails for a myriad of reasons. If using a version of ANSYS 2021 or later, the issue is likely due to the code itself (syntax, logic errors, segmentation faults etc.) or directly related to the environment (the compiler, OS). If compiling via journal commands proves an issue, it is recommended to compile using Virtual Network Computing (VNC). Using the ANSYS GUI allows for easy compilation as it comes with a built-in compiler. Simply save the compile library and copy to directories where there are instances that require the UDF. It is important to note that, a .cas or WB file cannot be saved with a UDF loaded as it will try to reference a library on the computer in which it was loaded.

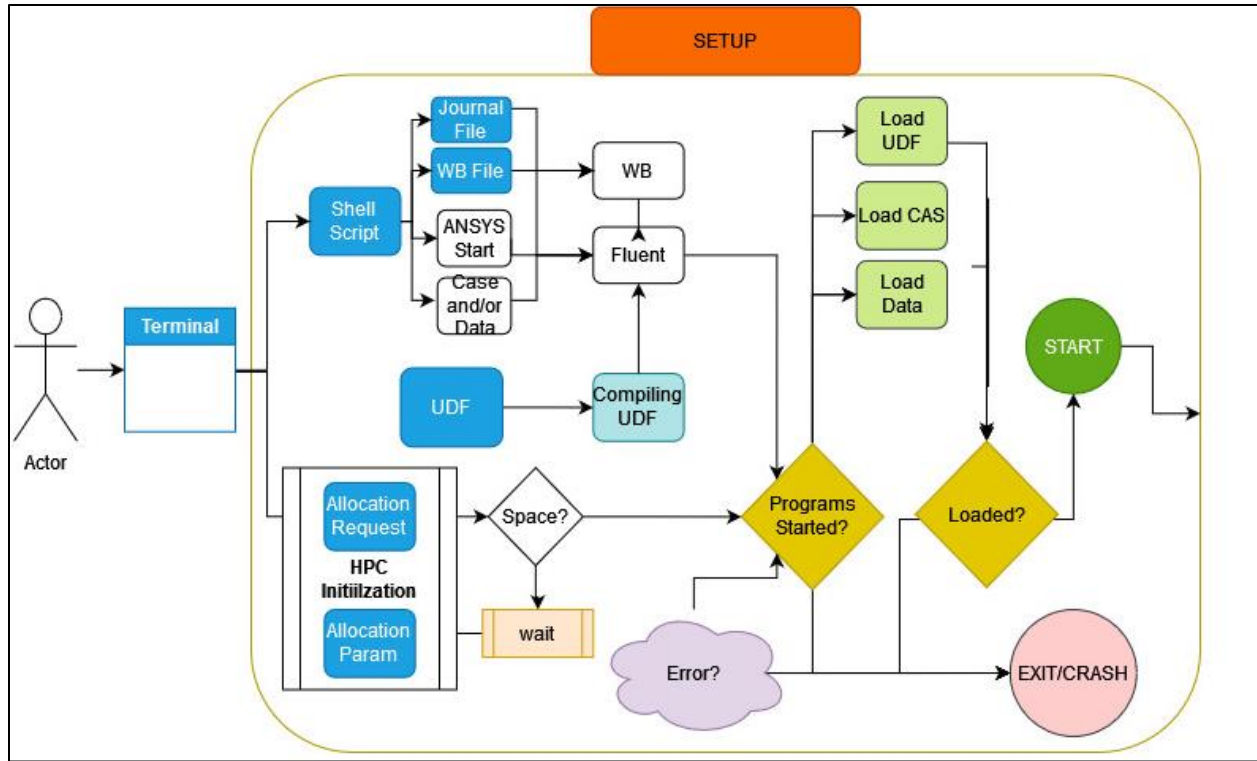


Figure 4-3: Setup - Flow Diagram

## 4.5. Data Processing

When the setup is completed and the job has been queued (or being prepared to start), the data processing stage of the workflow will begin. Data processing is almost entirely an automated process by ANSYS or by the loaded UDF. Regarding the CBPT, the data processing occurs in three stages: Data input, data output and file saving. When the CBPT was ran, the journal file or shell file specified if it was to run in parallel or serial (meaning that cores don't run in parallel). In series, ANSYS compiles and utilizes the UDF *normally*, meaning there is no need for special consideration for how to run the code. This changes if the code is run in parallel, but only for ANSYS side. The user of the CBPT will still receive the same performance from the UDF.

When operating in parallel, ANSYS divides the mesh into partitions that are assigned to each node. This means that each node can only see the partition it is assigned to as illustrated in Figure 4-4. While the mesh is divided, the domain and threads are mirrored in each compute node, allowing all nodes to access them, as if the program was run in serial.

To communicate via ANSYS and UDF, ANSYS has unique *define* macros which act as event handlers for the simulation. Event handlers are functions that run after a certain event occurs.

For example, the *execute\_on\_demand* macro runs whenever the relevant function is called in the ANSYS GUI or TUI. In the case of the CBPT, it uses a myriad of different macros but has three main ones: *execute\_on\_end* (executed at end of an iteration), *DPM\_Scalar\_Update* (runs after every particle update) and *define\_on\_demand*, (runs when called).

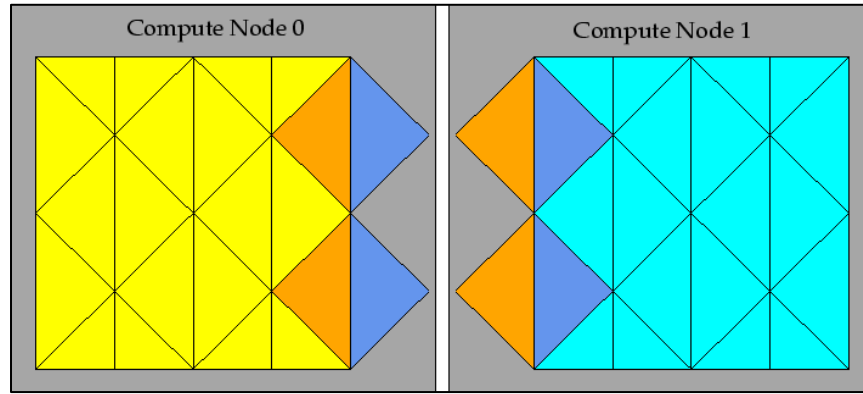


Figure 4-4: Partitioned Mesh Distributed Between Two Compute Nodes [104]

#### 4.5.1. Initialization

When first loaded, the CBPT will initialize any variables used in calculations. It will instantiate two user defined memory (UDM) locations that are used in calculations. These values are set to 0 for future use. UDM contains values that can only be changed by a UDF (at least directly).

#### 4.5.2. Data Input/Output

Data input in this context, refers to reading data in serial mode. The CBPT gathers data about the UDM values and performs computations on them. In parallel, each node will perform computations by using *DPM\_Scalar\_Update*. Each time a particle updates it will use code to update a UDM value. Since cells are stored in each node, the node with the particle will be responsible for loading the particle data and saving the new UDM value to the required thread and cell. To this end, the nodes do not need to communicate.

#### 4.5.3. Node Data Transfer

In serial, a single 2D array stores the values of all the output parameters specified. At the end of a specified time iteration, or when called directly, the UDF will save a file with a custom name based on the time step and save interval defined. A Boolean condition, called autosave, can

be triggered in ANSYS to turn this feature on and off. In parallel, the code logic works similarly, however, due to MACRO limitations, nodes can only share 1D arrays. A specific coding technique called encoding, or flattening, is used to ‘compress’ the 2D array into a one-dimensional (1D) array. The UDF takes out memory and creates a new array of pointers. These pointers point to new locations in memory based on the size of the 2D array. A simple linear equation can be used to store the row values for the pointers based on the size of the data type and the number of columns. The pointer simply stores the location of the first element in the array and can access the rest with the linear equation and the size of the columns. This is demonstrated in Figure 4-5.

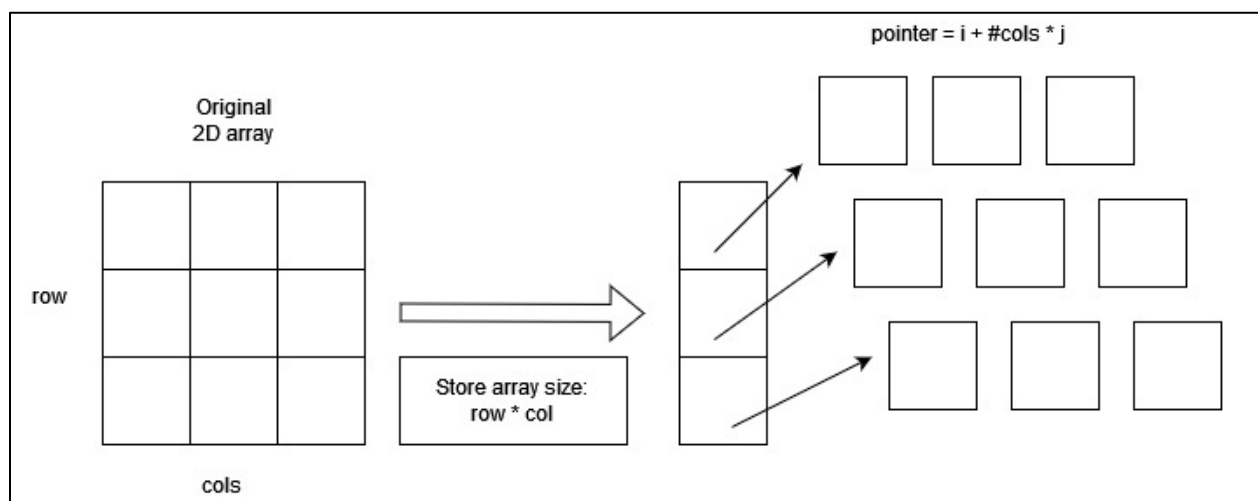


Figure 4-5: 2D to 1D Array Encoding

This 1D encoded array is then used to pass data to the other nodes and or host. The reverse processes can be done, known as decoding, where the memory locations are referenced using the 1D and a new 2D array is built.

#### 4.5.4. File Saving

To move data around to prepare to save it, the host node and follower nodes must all communicate and compile the data in a central location. This occurs in 4 stages: 1) Initialization, 2) Variable Storage, 3) Node to Node Data Transfer, .

##### 4.5.4.1. Stage 1 – Initialization

In Stage 1, various variables are initialized for use in storing variables. The number of variables being transferred, and containers for those variables (data) are initialized based on the

size of the data and datatypes. Initial memory allocation may also be done at this stage, although it isn't necessary. The follower nodes and host node each have different variables as they will need to store different things.

#### *4.5.4.2. Stage 2 – Variable Storage*

In Stage 2, the computational (follower or compute) nodes begin to gather the data requested from their storage. In the case of the CBPT (the specific code used in this thesis), it has two UDM values. These UDM values store the total particle count and the weighted particle count in each cell location. For this specific use case, the stored data is filtered, and the filtered data set is then stored. This filtering occurs by each node looping through all the cells in their partition. The compute nodes then store the values and position of cells with values greater than zero. This indicates a particle has, at one point, passed through the cell. After looping through all the cells, each compute node stores this value in a temporary array which is then encoded as described in Section 4.5.3.

#### *4.5.4.3. Stage 3 – Node to Node Data Transfer*

As described in Section 4.5.3, the follower nodes will begin to send their data to node 0 and node-zero will begin to accept their data and store it in a temporary location. In ANSYS giving and receiving data must be stated explicitly, and both nodes must have a copy of the container of the variable being sent. Immediately after receiving a value, node-zero will send the data to the host-node (cortex). After sending the data of all the other nodes, node-zero will send its own data to the host-node.

#### *4.5.4.4. Stage 4 – Node-Zero to Host Node Data Transfer*

In Stage 4, the host-node will read the data from each node in chunks. Since it cannot hold all the data (size of data and datatype is unknown) each node sends the size of its data before sending the data itself. The host-node then creates a container for the data and then saves this data to a file buffer. When all the data is read the host-node deletes all the stored data, and the other nodes follow suit to restore allocated memory.

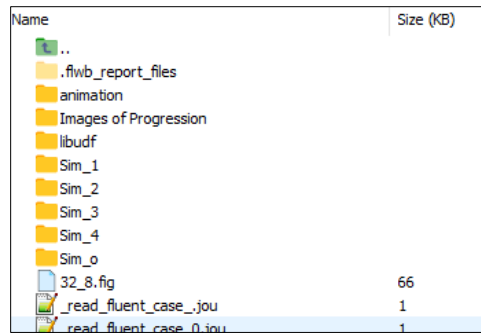
### **4.5.5. Data Storage**

When running on a local computer, file storage is very trivial. However, on a cluster, where the moving of files can be a long process, proper storage formatting is paramount. The

CBPT stores files in similar ways for both serial and parallel computing. The host-node writes file names to a folder. It saves based on the save interval value defined at the beginning of the code. The file names use a prefix followed by “t\_X” where X is the time interval number (this can be changed). A header is used so that other scripts (shell, journal, other UDFS) can change the value and thus create new files and file folders. The data is also saved in a folder relative to the location of the UDF so it will work in any location.

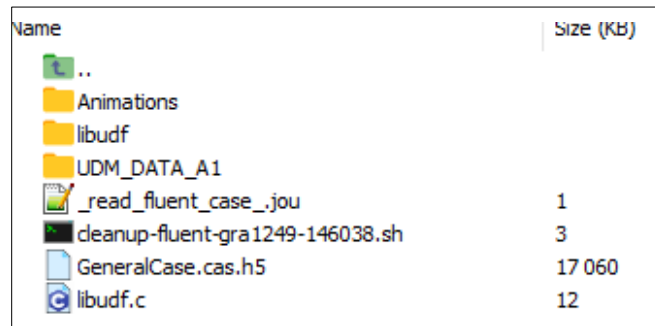
#### 4.6. Results and Discussion

Figure 4-6, Figure 4-7, Figure 4-8 showcase a SSH (Secure Shell) terminal instance that is connected to the compute Canada GRAHAM cluster. Inside, it shows the organization of files as well as the presence of the organized and labeled data files. The code for the UDF is available in Appendix C of this thesis.



Name	Size (KB)
..	
.flwb_report_files	
animation	
Images of Progression	
libudf	
Sim_1	
Sim_2	
Sim_3	
Sim_4	
Sim_o	
32_8.fig	66
_read_fluent_case_.jou	1
_read_fluent_case_0.jou	1

Figure 4-6: GRAHAM Cluster Showcase – Folder Overview



Name	Size (KB)
..	
Animations	
libudf	
UDM_DATA_A1	
_read_fluent_case_.jou	1
cleanup-fluent-gra1249-146038.sh	3
GeneralCase.cas.h5	17 060
libudf.c	12

Figure 4-7: GRAHAM Cluster Showcase – UDM DATA Folder

t ..	
UDM_DATA_A1_t_1.txt	0
UDM_DATA_A1_t_10.txt	0
UDM_DATA_A1_t_100.txt	3 328
UDM_DATA_A1_t_101.txt	3 397
UDM_DATA_A1_t_102.txt	3 459
UDM_DATA_A1_t_103.txt	3 528
UDM_DATA_A1_t_104.txt	3 592
UDM_DATA_A1_t_105.txt	3 660
UDM_DATA_A1_t_106.txt	3 726
UDM_DATA_A1_t_107.txt	3 800
UDM_DATA_A1_t_108.txt	3 861
UDM_DATA_A1_t_109.txt	3 924

Figure 4-8: GRAHAM Cluster Showcase – DATA Files

An example of the UDM data from the perspective of ANSYS is shown in Figure 4-9. The particle dispersion cloud is shown as a representation of the cell dispersion “risk cloud.” The colored gradient is a visual representation of a low threshold manifold as shown in Figure 4-9 and Figure 4-10 at various time steps. The UDM value will be explained further in Chapter 6. It can be thought of as a summed WPC as is explained in more detail in Chapter 6 when discussing the Weighted Particle Count in Chapter 5 (Section 5.2).

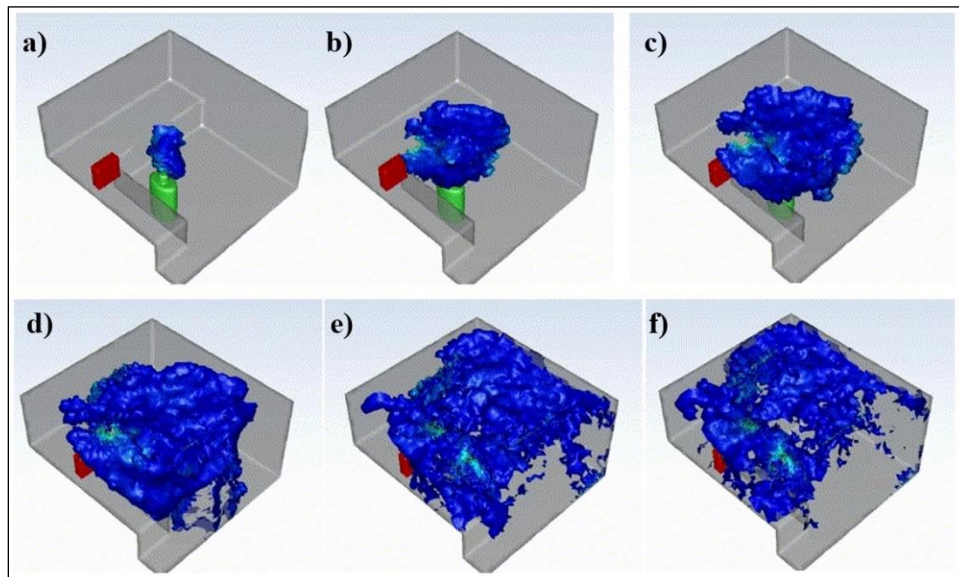


Figure 4-9: Progression of Domain WPC at Particle Time at: a) 1 second, b) 2 seconds, c) 3 seconds, d) 5 seconds, e) 8 seconds, f) 10 seconds.

The Spatial RAM model is shown in Figure 4-10. These images were generated in MATLAB and represent graphical representations of the UDM propagation over time. These images go past 10 s particle time until around 15 seconds. This was done to investigate the presence of significant dispersion artifacts once the particles have started dispersing along the room's outer edges. No such artifacts were noted for the investigated cases. The risk factor, calculated using the UDF, is shown at various cell locations in Figure 4-10 and 4-11.

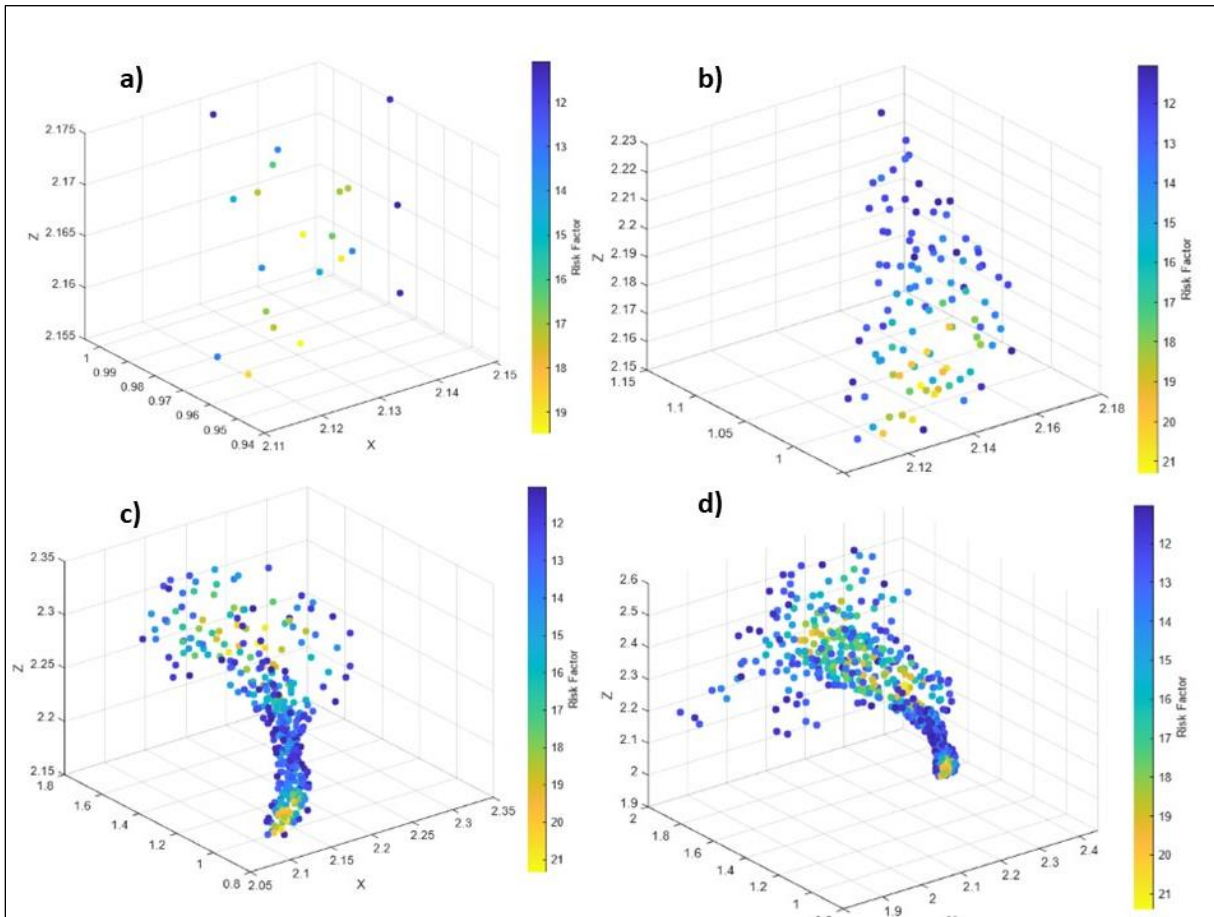


Figure 4-10: Spatial-Temporal Risk Model for Particle Time at: a) 0.1s, b) 0.2s, c) 0.5s d) 1s

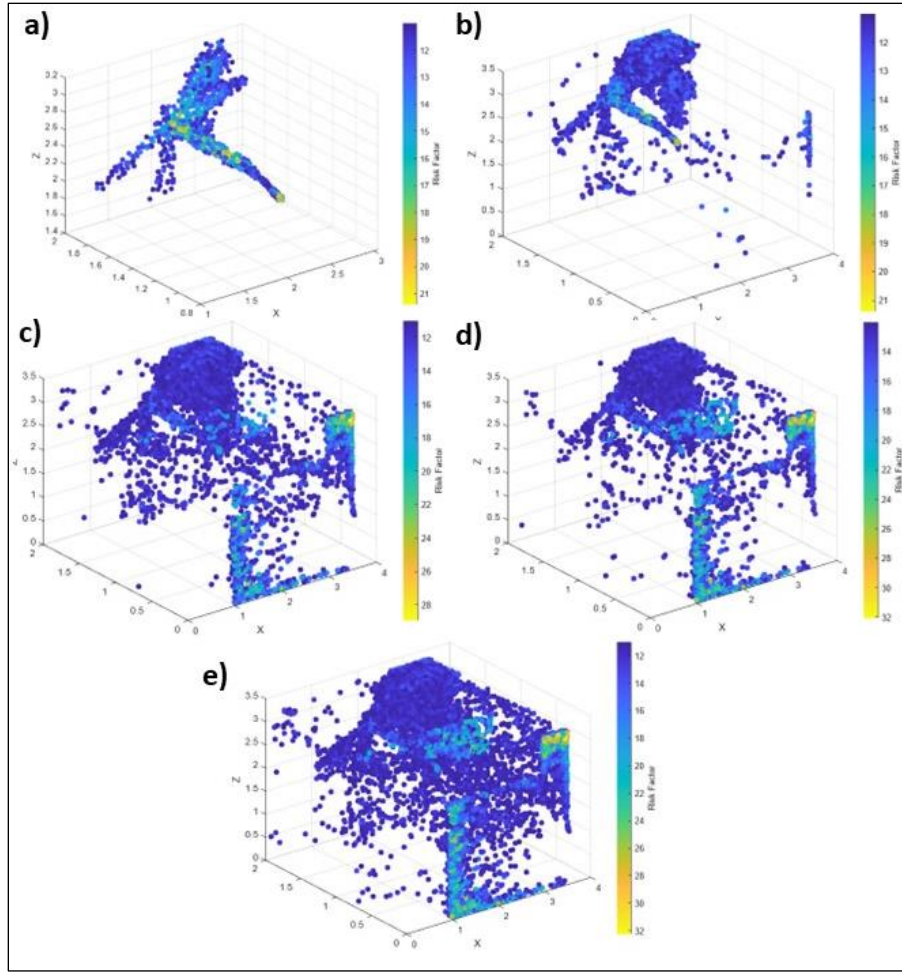


Figure 4-11: Spatial-Temporal Risk Model for Particle Time at: a) 2s, b) 3s, c) 5s d) 10s e) 15s

## 4.7. Conclusions

### 4.7.1. Summary

A UDF was created to work in ANSYS Fluent that successfully extracts and transforms particle dispersion data. This means that future UDFs can fine tune more elements of the simulation by adding in other aspects, such as empirical models or simulating the effects of natively built-in ANSYS models. By using this UDF in ANSYS Fluent complex computations can be performed more easily by saving data space and reducing data processing times. This UDF was also converted successfully to be compatible with both serial and parallel processing modes and it can also write output files in both serial and parallel processing modes. Finally, the UDF works in parallel on the HPC cluster nodes, specifically those of Compute Canada. This is a huge step because it means that large amounts of data with high fidelity can be run at accelerated speeds, and the data can be

stored on the HPC cluster, with its massive storage allocation. Future simulations will not suffer from limiting computational or storage-based resources and can therefore focus on advancing computational dynamics in all fields, especially CFD.

#### 4.7.2. Future Work

Future versions of the CBPT will expand its ease of use for those unfamiliar with programming, scripting, or using virtual HPC resources. The use of this CBPT tool can also aid in rapid data gathering for use in other advanced optimization algorithms, like machine learning. The UDF itself can incorporate experimental data [46,53,106–108], for example, including empirical evaporation data into the dispersion model. By comparing particle values to experimental data related to particles like settling time, and evaporation time, experimental based pseudo-evaporation models can be developed. Similar things can be done for other types of models [109–111] as well, not just for aerosols and droplets. It can also be done for different fluids, solids and has reaches in other ANSYS software or computational software as well, such as, ABAQUS [49,112,113], COMSOL, STAR CCM+ [47,48,114], or other custom software [50,52,115,116]. The use of parallel processing can also be expanded by using C functions and scripting. More recent versions of ANSYS, such as ANSYS 2023, allow for compiling UDFs in C++ which can be used to further enhance the computational capabilities of the CPBT tool developed in this thesis.

## Chapter 5. Development of Novel Methods for Predicting Temporal-Spatial Particle Dispersion using CFD and ML

### 5.1. Introduction

This chapter outlines the use of the CFD and the CBPT tool previously developed in Chapter 3 and Chapter 4 to run high fidelity simulations for use in a Machine Learning (ML) model to predict the particle Risk Factor. Previous studies have done much work in refining the process of creating and running CFD simulations [15,19,93,101]. This study will supplement the use of CFD by creating ML models, like those done in literature [67,99], to predict the spatial spread (dispersion) of airborne particulates.

The computational power and parallel capability of Compute Canada accessed via *MobaXTerm* [117] was harnessed to run extensive simulations. Bash scripting was employed to specify the resources used, allowing for precise control over the simulation environment. This setup facilitated the execution of complex simulations, optimizing resource allocation. It was used to run a system of 4 simulations of the mixing ventilation configuration design case. Each case had a different ambient temperature. The results were then processed using a variety of data processing techniques to extract the core dispersion features and calculate the risk factors throughout the domain. This data was then transformed into regional data by finding the high concentration volumetric region, represented as a bounding box, and the location of the centre of that box. These two models, called the Spatial-Temporal model and the Regional-Temporal model were used to train ML models of various archetypes. The general ML Development framework is shown in Figure 5-1, and the final RAM model development process overview is shown in Figure 5-1 and Figure 5-2.

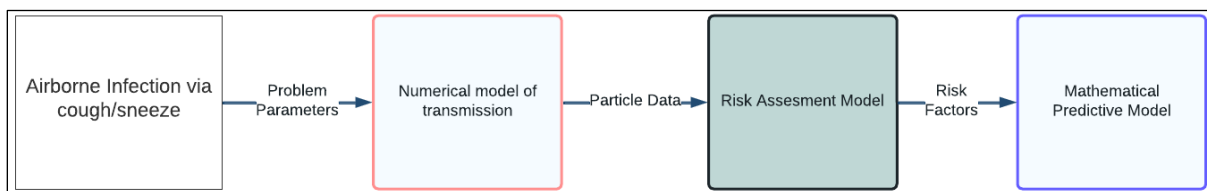


Figure 5-1: ML Model Development Overview

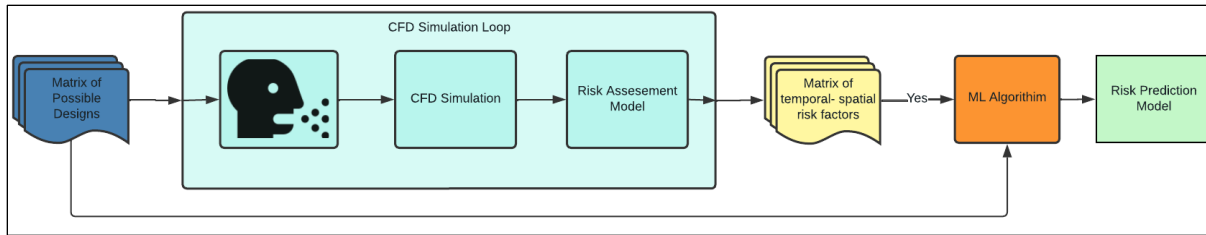


Figure 5-2: RAM Model Development Overview

## 5.2. Methodology

### 5.2.1. Data Extraction

Utilizing ANSYS Fluent, simulations were run for 10,000 iterations and detailed data was generated in a spatial-temporal format, inclusive of concentration data and stored in text files. The process was done using the Cell Based Particle Tracker (CBPT), as described in Chapter 4. The CBPT was used to extract the *Total Particle Count* per cell (TPC). This data was saved in an array and then saved to a file algorithmically over the course of the simulation. This data was subsequently processed through Python scripting, transforming it into a structured CSV format. Additional attributes, such as simulation ID, time step, and temperature, were incorporated to enrich the dataset.

Four simulations were run with temperatures ranging from 17 °C to 25 °C. Convergence of the flow field was achieved before the simulation proceeded with particle injection (cough or sneeze). At certain time steps, as set by setting the save interval in the CBPT, the file saves the UDM data (TPC) at all the cells that are nonzero. In other words, it will save the values of cells that aren't zero. An example of how this cell works is shown in Figures 5-3 and 5-4. Particles move across the boundary and after every time step, the mesh records the number of particles in each cell across the domain.

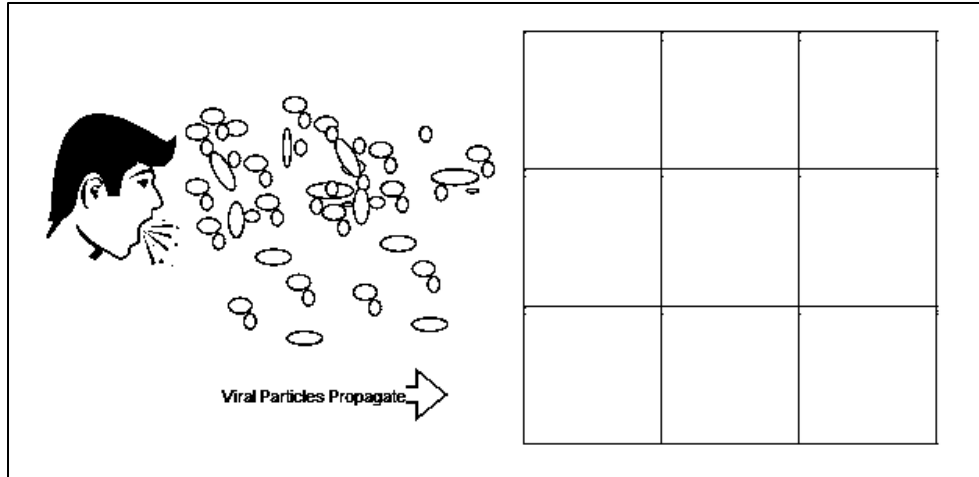


Figure 5-3: Cells with no particles

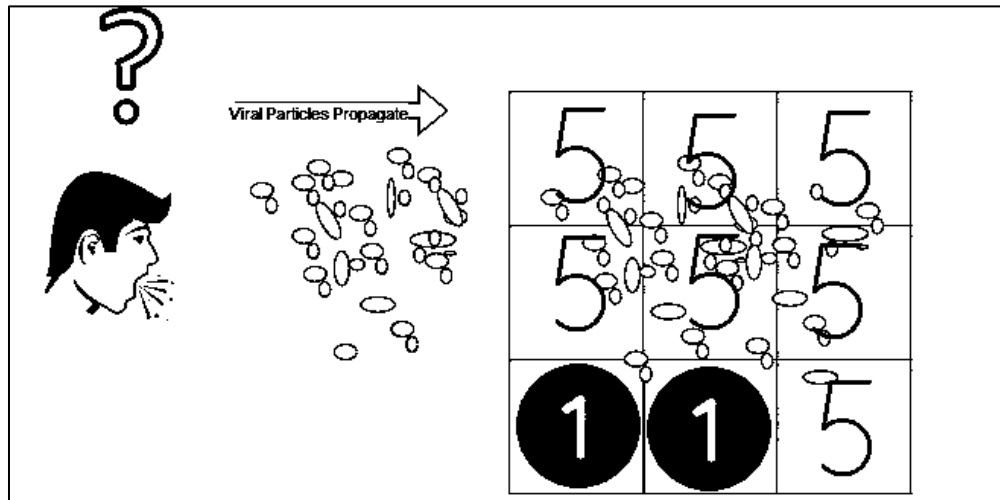


Figure 5-4: Particles reach cell elements and get counted.

The particles are counted using a combination of a *Particle Count Value* (PCV) which is set to a value of 0.1. This is the base worth value of any one particle and is added to allow for more fine tuning of particle worths in future versions of the code (*i.e.*, having multiple particle variants with different risk or worth values). Particle diameters are sent into a weighting function to obtain a *Particle Diameter Weighting* (PDW). This is analogous to the mass of the particle. It weights the value of the particle based on its normalized diameter. This normalized diameter is based on minimum and maximum particle diameters of 1-100  $\mu\text{m}$  [11,59]. The function used is shown in Equation [5-1] and plotted in the Figure 5-5. In this case,  $x$  is the normalized particle diameter.

$$PDW(x) = \frac{1+0.1}{0.85+e^{-6(x-0.6)}} \quad [5-1]$$

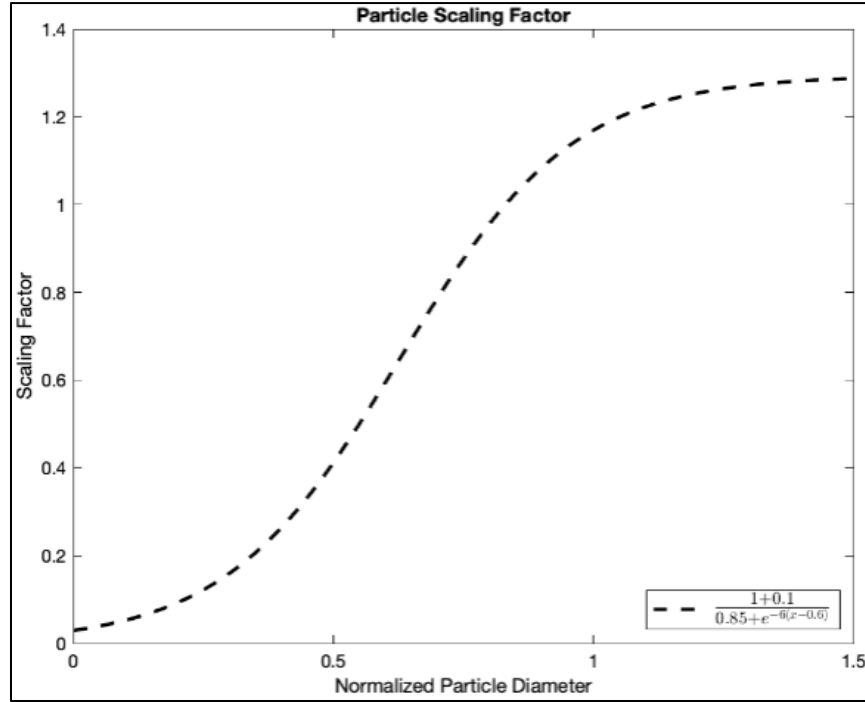


Figure 5-5: Particle Diameter Weighting Function

The Particle Diameter Weighting Function is based on the mathematical distribution made by Khan [11]. A transformed sigmoid function distribution was fitted to the values [11]. The weighted data is referred to as the *Weighted Particle Count* (WPC). It is calculated by the summation of the product of the PCV and PDW each time step. The WPC data is sent to MATLAB and Python for imaging and processing. The equation for the WPC is shown in Equation [5-2] where  $P_w$  is the cell particle weighted value,  $P_v$  is the particle value, and  $P_d$  is the particle diameter weighting.

$$P_w = \sum_{i=1}^n (P_{v,i} \cdot P_{d,i}) \quad [5-2]$$

### 5.2.2. Data Processing

The raw data was first converted to a CSV file using a custom script. Then the data was processed in two stages. The first stage converted the Weighted Particle Count (WPC) into the Particle Risk Factor (PRF). The second stage filtered this data to obtain the cell locations with relatively high-risk, as well as to reduce the total number of data points.

#### 5.2.2.1. Stage One – Conversion of WPC to PRF

The WPC has a wide range of values, due to the large number of time steps, as well as the large number of particles. The original data sets had sample values from  $10^{-1}$  to  $10^5$  orders of magnitude. To evaluate the risk, a Risk function was used to turn the weighted particle count (WPC) into the Particle Risk Factor (PRF). The PRF factor is made using a logarithmic transformation function to normalize the data, into values of  $10^0$  to  $10^2$  orders of magnitude. The functions are shown in Figure 5-6 and Equation [5-3] below. The values of C and B are based on trial on error, to limit the function output (and thus the risk factor range) from 0 to 100. The value A is used to raise the value in the natural log functions, so it is always more than 1. Values of A that are lower than one would result in a negative risk factor, which was not desired for this study. For this study, A was set to a value of one for simplicity. In this case,  $x$  is the WPC as discussed in section 5.2.1.

$$PRF(x) = C^3 \cdot \ln(x + A) + B \quad [5-3]$$

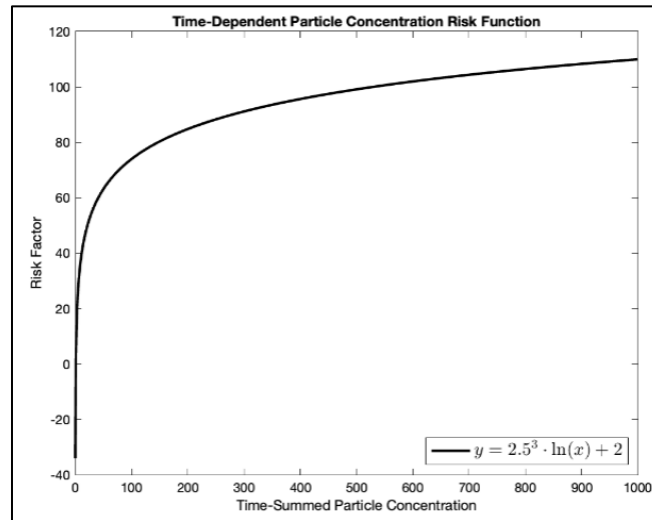


Figure 5-6: Particle Risk Factor as a function of time-summed particle concentrations

With this risk factor, temporal regions can be compared in terms of relative risk. The data can now be qualitatively and quantitatively analyzed over time. This data can then be used to build machine learning models to predict temporal and spatial risk factors.

#### 5.2.2.2. Stage Two – Data Filtering

To identify the high-risk regions and to reduce the number of data points, a data processing technique was used to filter the data based on a minimum threshold. This algorithm had a global risk minimum threshold that started at 10. The algorithm goes through the data and keeps every PRF datapoint at each time step (text/csv file) to a maximum of 500 points. If the algorithm notices that the file goes over 500, it increases the global risk minimum threshold and then rescans the file, removing points that do not meet the minimum threshold. After this data filtering process, each file will have a maximum of 500 lines of text, dataset consistency helps when training the ML models.

### 5.2.3. Risk Models

#### 5.2.3.1. *Spatial-Temporal Risk Model*

The Spatial Temporal Model represents a detailed view of particles in a 3D space, capturing concentration, temperature, and temporal aspects. While providing a rich dataset of information, this model is computationally demanding, making it a significant consideration in the overall analysis. The complexity of the Spatial Temporal Model offers a nuanced understanding of the simulation but comes at the cost of computational efficiency. The model offers a 3D gradient of the risk in 3D space based on time and Temperature as an input parameter. The model then can predict the risk factor at a location when given the relevant parameters. It is possible this model may not be able to create a correlative model with a low percentage error for large complex spatial distributions. This is due to factors such as spatial dependence, data resolution and complex relations (*i.e.*, turbulence). Because the data gathered by the CBPT is Time-dependant at specific Cell locations, having many data points is not an issue.

#### 5.2.3.2. *Regional-Temporal Risk Model*

The Regional-Temporal Spatial model predicts the risk factor (analogous to the summed concentration over time) of the particles in space over time. However, it would be difficult for the ML to predict such a model, especially if larger domains are used. In contrast, the Regional-Temporal Model simplifies the representation by grouping particles into bounding boxes in 3D space, each with a centroid and concentration at specific time points. This model reduces computational overhead while retaining essential spatial and temporal information. The difference between the Spatial Temporal Model and the Regional-Temporal Model is the trade-off between computational efficiency and the granularity of information.

#### 5.2.3.3. Generation of Machine Learning Models

When utilizing ML models, it is common practice to separate the data sets. Usually, this is done with three groups: training, testing and validation. Training is used to build the initial model and testing uses the loss function to temper the ML first iterations. After the ML determines it cannot generate a better model, the new model is validated with the loss function (or other metrics) at the end with the validation test set. The validation tests can also be repeated for further enhancement of the model. Besides the three general groups (training, testing, validation) for building the ML models, there are two distinct data sets that were used to create two distinct risk models: the spatial model and the regional model. The two datasets were made via eight simulations, four for the spatial model and the other for the regional model. The various ML models were trained on the first simulation using 100 epochs and a batch size of 10 or 100 for the regional and temporal models respectively. The choice of batch size was done manually. The regional model had one order of magnitude less data than the spatial data. Thus, ten was chosen for its batch size. The model was tested using batch sizes ranging from 5-20 and no major difference in performance was found, for sizes lower than ten. Batch sizes higher than ten, performed similarly but not as well. Similarly, the temporal model used batch sizes ranging from 50-200. It was found that performance started to fall off after 100, where the model began to fail to find meaningful correlations in the data. Batch size values less than 100 did not find any meaningful difference in performance. All models had a two distinct data sets using spatial and regional data. Each data set had 260,000 datapoints and 1,000 input samples, respectively.

The input features consisted of '*timestep*' and '*temperature*' for both RAMs (Risk Assessment Models). Three machine learning models were employed: Deep Neural Network (DNN), Support Vector Regression (SVR), and Radial Basis Function (RBF). The models were evaluated using Mean Squared Error (MSE) and Mean Absolute Error (MAE) in conjunction with the percentage average error (PAE), which will be referred to as just percentage error. The percentage error expresses the average difference between the MAE of the model. The equation is shown in Equation [5-4] below:

$$PAE = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100 \quad [5-4]$$

The choice of models and error choice was done for comparison between the models themselves and as a comparison with the results found by Mirzaei et al. [99] and Mesgarpour et

al. [67]. Both studies utilized these errors in their results. Mirzaei et al. [99] used a Deep artificial neural network (DNN) with back propagation and used the sigmoid function with smooth gradient. They tested various architectures and found the 10 x 10 DNN (10 layers with 10 neurons) delivered the best results. They used a data set with 2,100 training input samples. Mesgarpour et al. [67] utilized varying ML models to create a MIMO model for their study with 1,240 input samples. They also used a DNN as well as MLP, SVR, and RBF. To mirror both studies, a Deep Neural Network (DNN), Support Vector Machine (SVR) and Radial Basis Function (RBF) will be utilized. This study found that the methodology used by Mesgarpour et al. [67] produced much more consistent results for the architecture of the neural network (5 layers, two for input, two for output and three hidden layers), with some changes, as will be discussed.

The SVR model was implemented using Scikit-learn's *MultiOutputRegressor* wrapper. The kernel used was 'linear.' The RBF model was implemented similarly to the SVR model but with the kernel specified as 'rbf.' The input features and target variables were the same as in the SVR model. The RBF model regularization factor ( $C$ ) was set to 2 and the kernel coefficient ( $\gamma$ ) was set to 0.01. These values were found using a small random search for parameter optimization.

The DNN model was implemented using Keras and TensorFlow, as was done in previous studies [67,99]. The DNN model architecture consisted of a Sequential model with three hidden layers containing 5, 15, and 30 neurons, respectively. Testing found that progressive stacking of the neurons produced consistently good models. The input layer had  $n$  neurons with  $n$  input dimensions, and the output layer had  $m$  neurons. The values of  $n$  and  $m$  were the number of input and output parameters, respectively. The activation function for all layers was the rectified linear unit (ReLU). This activation function is widely used in DNN regression models for nonlinear models as it does not become saturated with large numbers of inputs, and it is advantageous because errors are back propagated [67]. Unlike the Mesgarpour et al. study [67], the output layer did not use the sigmoid function, as the output values were normalized and none of the values were negative. The model was compiled using the Adam optimizer with Mean Square Error (MSE) loss and Mean Absolute Error (MAE) metrics. The Adam optimizer is popular and widely used due to its ability to adjust the learning rate adaptively and automatically. The values of the parameters of the Adam optimizer were left as default. The use of MSE as the loss function and MAE as metrics is common for regression models. The Loss function is used for training and testing, while metrics are used for validation cases. MSE emphasises the difference between values by giving weight to

larger errors making it very sensitive to differences in data sets. The MAE, in contrast, calculates the absolute difference, hence, it is not as sensitive and better for evaluating the general performance of ML models. To that end, MSE is better for loss as it can effectively shape the ML model during training and testing, while MAE can give a more general view of the model and avoid overfitting the ML model to the validation test case. The MAE provides a means to evaluate the general performance of the ML model.

### 5.3. Results and Discussion

The results of the ML models for both the Spatial-Temporal Risk (STR) and Regional-Temporal Risk (RTR) models are shown in Sections 5.3.1 and 5.3.2, below. The models were evaluated using Mean Squared Error (MSE), Mean Average Error (MAE) and Percentage Average Error (PAE). The validation results of the various ML models were used for each evaluation parameter (MSE, MAE and PAE). The validation results were chosen since they give less biased interpretation of performance of ML models, as opposed to training and testing datasets.

#### 5.3.1. Spatial-Temporal Risk Model

The results for the Spatial Temporal Risk (STR) Model are shown in Figures 5-9 to 5-11 and in Table 5-1. The DNN and RBF produced the best results with SVR (a linear model) performing notably worse. These results mirror the results of Mesgarpour et al. [67]. The reason for the DNN and RBF models performing better than the SVR is likely due to the stark similarities between the two models, both are nonlinear. The models only had to predict a single output parameter, making the prediction task easier. The ML Model in this study also has an abundance of raw data in which to train and test against, on the order of  $10^6$  datapoints. If the ML model were to perform poorly, the effects of the filtering can be relaxed (reduce threshold increment) or the cap on the number of points per file (simulation iterations) increased. The table below, Table 5-1, shows the maximum MSE, RMSE (root of the MSE), MAE and the PAE validation results from this study.

Table 5-1: Validation Results of STR Machine Learning Models

Model	Output Variables	MSE	RMSE	MAE	PAE (%)
DNN	PRF	10.01	3.16	2.65	12.36
SVR	PRF	58.16	7.62	6.60	27.93
RBF	PRF	14.38	3.79	5.12	18.12

The results of the DNN are shown in gradient *slices* at a set height in example figures (see Figures 5-9 to 5-11). They are 2D visualizations of a portion of the 3D risk map generated by the ML model. For reference, the injection location is at (2.12 m, 0.95 m, 2.17 m), the approximate height of the nose and mouth of a person seated at a desk, which is typically in the middle of the XZ plane. The time value referenced is an internal time used for the ML model. One unit corresponds to 0.5 s of real particle dispersion time.

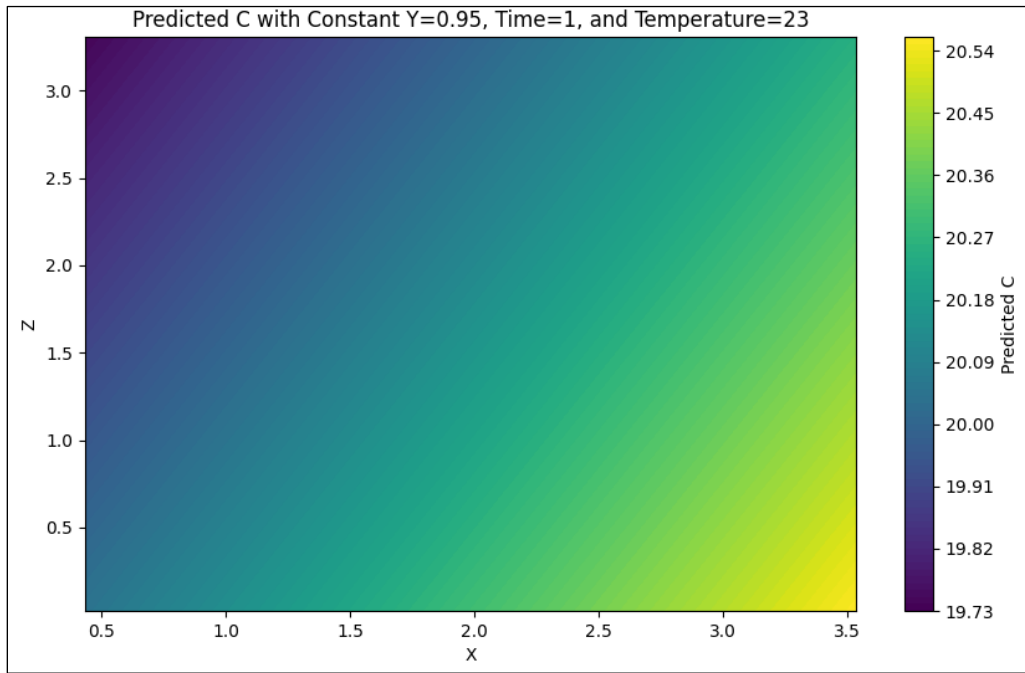


Figure 5-7: The results of the DNN model's XZ Risk Gradient at  $y=0.95$  m and 0.5 s

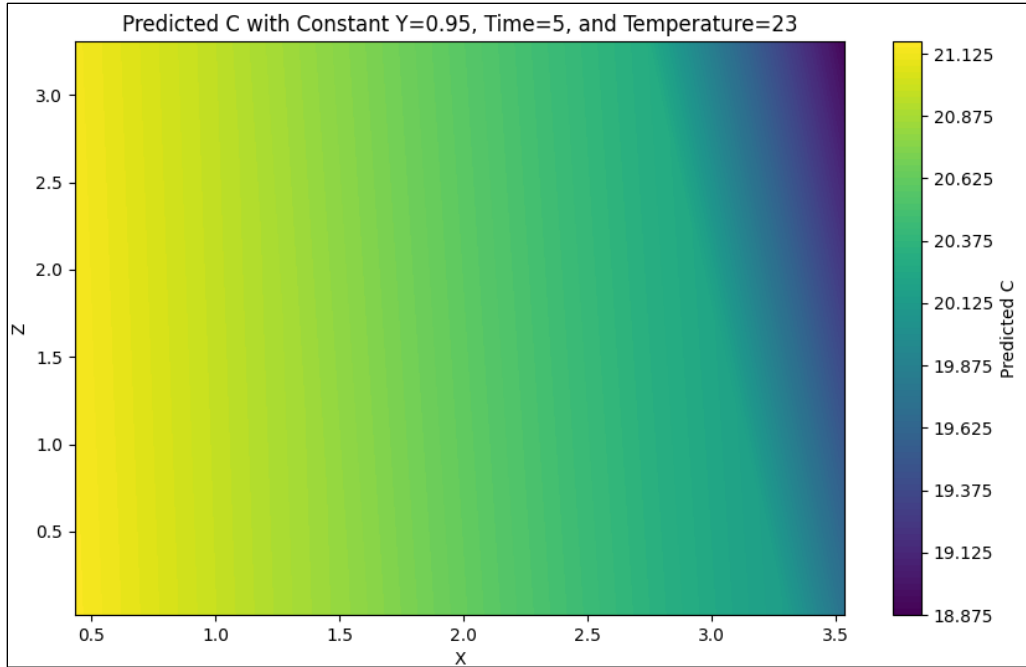


Figure 5-8: The results of the DNN model's XZ Risk Gradient at  $y=0.95\text{m}$  and  $2.5\text{s}$

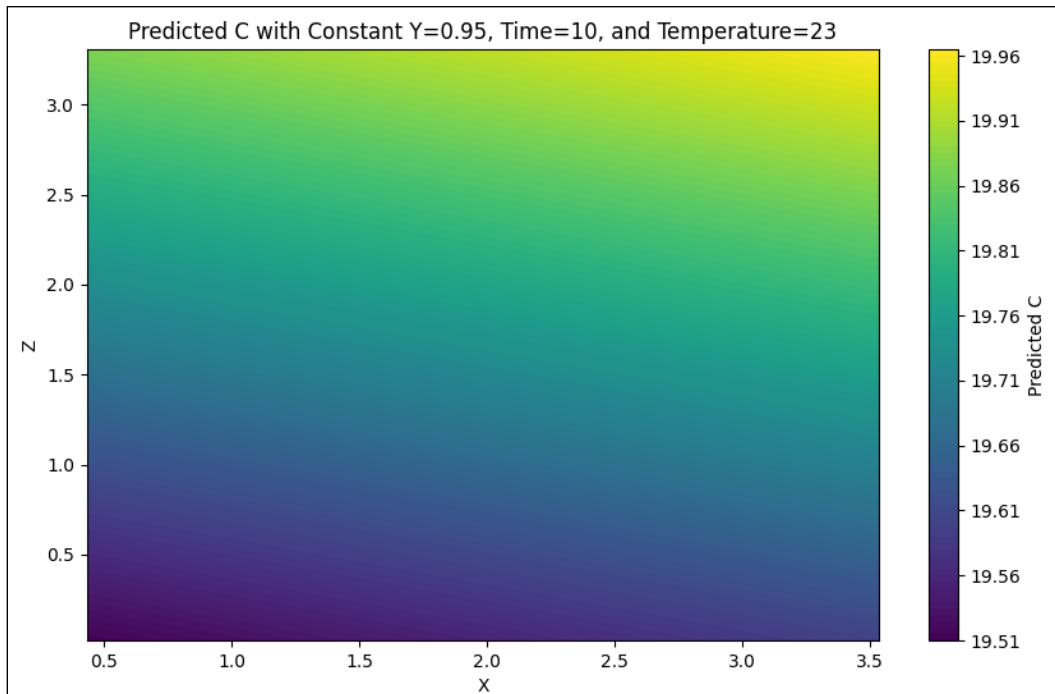


Figure 5-9: XZ Risk Gradient at  $y=0.95\text{m}$  and  $5.0\text{s}$

A gradual progression of the particles through elements of the domain for a mixing ventilation configuration can be seen in Figures 5-9 to 5-11. Initially, the risk is clustered at regions

very close to the injection. For this first 2D slice (Figure 5-9), the gradient is concentrated in the bottom right quadrant. This is likely due to a combination of the particles moving in the Y direction as well as the ML model not having much data in this region, so it is extrapolation information based on the data it does have. This same pattern is also observed in the other two 2D risk maps (Figure 5-10, and 5-11). In Figure 5-9, a heavy concentration in the left side of the room can be seen. This is due to particles being injected in a linear stream, but they are heavily favored to the left side of the room due to the fluid flow in the room. As investigated previously (Figure 3-6 and 3-8), Mixing tends to push the particles upwards and towards the left and right of the room. Due to the position of the exhaust, most of the particles were sucked upwards towards the left (into the exhaust). In Figure 5-11, the particle concentration is clustered in the top right corner, as the particles are now clustered at the back region of the room. The risk map is the ML's mathematical model based on the limited spatial data provided by the CBPT.

### 5.3.2. Regional-Temporal Risk Model

For the RTR Model, three different configurations were tested. The first configuration used  $[L, W, H]$ , which are the size of the high-risk region (bounding box). The second configuration used  $[X, Y, Z]$ , which is the location of the centre of the bounding box (so the geometric centre of the high-risk region). The third configuration used all six input variables combined  $[L, W, H, X, Y, Z]$ . It was anticipated that 6 variables would perform the worst, since it is generally harder to predict as the number of inputs to output variables changes. However, the results show that the first configuration,  $[L, W, H]$ , prediction performed the worst in all three cases and the third configuration,  $[L, W, H, X, Y, Z]$ , prediction performed in between the other two. This means that all models found prediction of the size of the bounding box the hardest, which is to be expected. Like the previous ML risk model, SVR performed the worst and RBF and DNN performed the best, with RBF having the best performance. This likely due to the benefits of the radial basis function in predicting spatial based data due to their strength in local approximation, meaning they can easily capture local patterns in spatial data.

Table 5-2: Validation ML Results for Regional-Temporal Model

Model	Output Variables	MSE	RMSE	MAE	PAE (%)
DNN	[X, Y, Z]	0.007	0.084	0.041	1.55
DNN	[L, W, H]	0.130	0.361	0.337	13.32
DNN	[X, Y, Z, L, W, H]	0.036	0.190	0.053	9.37
SVR	[X, Y, Z]	0.144	0.379	0.344	13.43
SVR	[L, W, H]	0.228	0.477	0.340	38.90
SVR	[X, Y, Z, L, W, H]	0.186	0.431	0.342	26.16
RBF	[X, Y, Z]	0.015	0.122	0.084	3.16
RBF	[L, W, H]	0.014	0.118	0.105	6.50
RBF	[X, Y, Z, L, W, H]	0.012	0.110	0.099	5.00

Figure 5-12 shows an example of the DNN predicting a bounding box at a given time and temperature. The ML model tends to over predict the size of the region. This is because the algorithm is predicting a single large cluster of particles with a simple geometry (rectangular prism) as opposed to looking at multiple clusters or predicting the risk using more complex shape geometries (spheres, ellipsoids, convex hulls). The algorithm used to generate the bounding box data uses local maxima of the high-risk cell locations on each axis for each time step, further simplifying the dimensionality of the region in question. Complexity would make the region predictions more specific but require more information to build the model. A simpler region would require less information to predict but loses the cloud shaped intricacies one would expect particle dispersion to exhibit. The reason for the choice of a simple region is because a simpler model is much more applicable for risk prediction models, especially when looking at larger temporal and spatial domains. For example, if this model were scaled up to a larger room such as a lecture hall or lunchroom, a more generalized model would become increasingly more desirable. Approximating a dispersion area as a cubic risk cloud means individuals at risk of infection could be easily included or excluded from consideration. In these cases, its better to over predict than under predict.

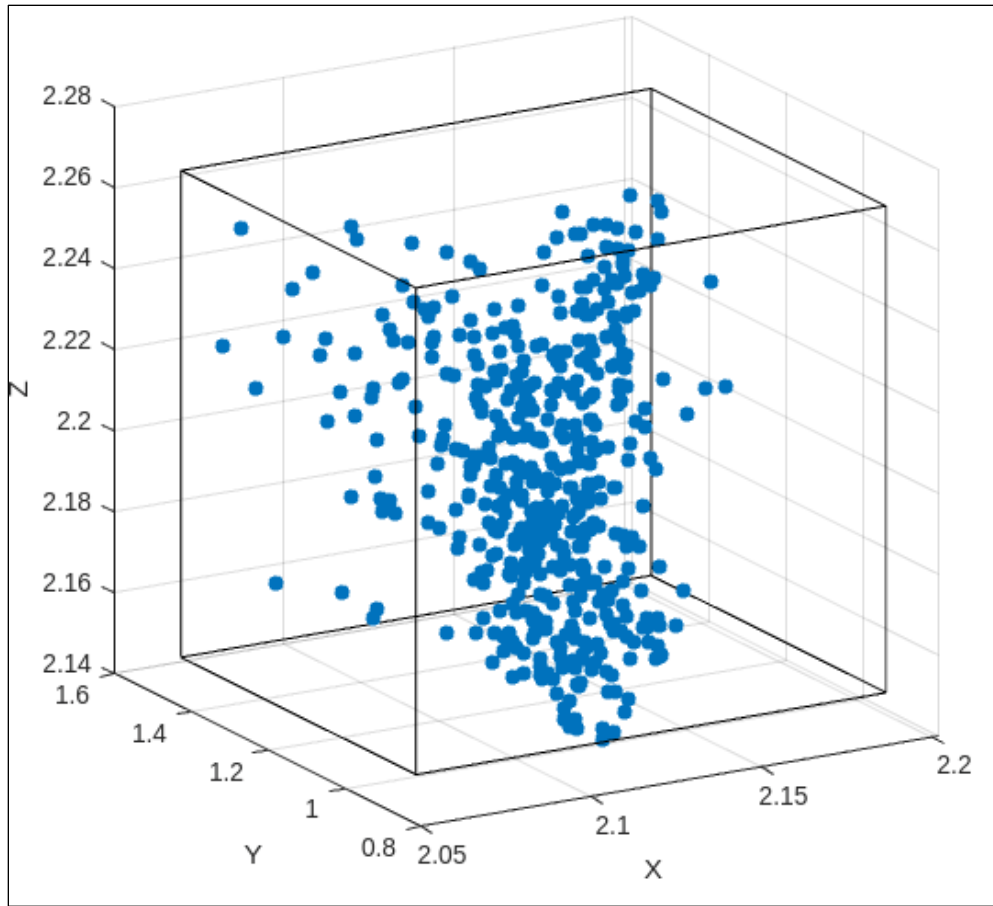


Figure 5-10: Example of the bounding box Created by Regional Risk Model

### 5.3.3. Summary of the ML Results

Based on the results discussed above, the Regional Model (RTR) performed much better than the Spatial model (STR) in almost all cases. This is because the Spatial model predicts risk values across a large dimensional manifold, and it must predict risk values in both the spatial and temporal domains. In contrast, the regional model utilizes dimensionality reduction, predicting the size of the space over time, and not the gradient of the values in it. If the spatial data was utilized in conjunction with the Regional Model, it could then pinpoint exact locations of high risk, but this would require large amounts of generated data. In contrast, just the regional risk model identifies a general area of high risk while requiring a relatively small amount of data samples at the upper margins of the risk profile, at the cost of knowing the actual locations of the risk.

## 5.4. Conclusions

### 5.4.1. Summary

In this study, it was shown that utilizing various post-processing techniques allows the conversion of particle count data into two distinct RAM models. The data was first generated by running simulation data in ANSYS Fluent and recording the weighted particle count at cell locations in the domain. These cell locations were locations that a particle had passed through, at any time step. Afterwards, the data was converted to a Risk factor using a logarithmic function and the data was filtered to keep all files to a maximum of 500, or less, data points or lines of data. Using this data, the Spatial-Temporal Risk Model (STR) and the Regional-Temporal Risk (RTR) Model were created. The STR was made using a ML model based on the spatial (x, y, z), temporal (t) and ambient (T) variables as input parameters and the associated risk factor as the output variable. The STR model performed well because it creates a continuous 3D risk space where the risk at any location can be predicted. However, the percentage average error (PAE) of the model was generally higher when compared to the Regional-Temporal Risk (RTR) model. The RTR model was made using an algorithm to find the size of the high-risk bounding box (finding the Length, Width and Height of filtered data) as well as the location of the box (X Y Z), represented as its centroid, as output layers and time and temperature as input layers. The RTR model results show that the PAE was lower overall compared to the STR model. However, the RTR model performed well due in part, to the square geometry of the room. A drawback of the RTR model is that it cannot provide what the actual high-risk region is, instead it simply provides an estimate based on the relative high-risk elements that were prompted by the initial values. In other simulations these ‘high-risk’ regions could be low, but the RTR model does not provide this information. The user needs to determine this manually by checking the data. The benefit of the RTR model is that it does not require as many data points as the STR Model.

### 5.4.2. Future Work

Future work will machine learning will incorporate a wider range and use of input and output parameters into the ML models. For example, velocity, pressure, among others, could be included. The start and end times of the simulation data that is used to build the ML model could also be much smaller. The datasets at the end of the simulation time step, were cluttered with similar values, often thousands. With that large amount of redundant data, it often leads to low variance,

thus extensive cross validation studies would be required for validation of data correlation. The spatial and temporal range of the simulations analyzed could also be much smaller, since the regional model has the bounding box values converge quickly as particles reach the bounds of the room. Smaller time steps in the ANSYS simulations will allow for more accurate temporal correlation between the input and output variables. In the same vein, looking at a more limited time window, will also limit the spatial area that the particles travel in. This will allow for much higher rates of variance in the data, and thus more accurate ML models. The general ANSYS simulation can also be changed by expanding the geometry, evaluating multiple scenarios, and investigating other indoor environments.

## Chapter 6. Conclusions

### 6.1. Summary

This thesis studied the airborne transmission in indoor office environments using CFD and ML. The effects of ventilation strategy particle dispersion were studied in addition to comparing various performance indicators. A comparative study was conducted evaluating the particle removal for stratum ventilation (SV), mixing ventilation (MV) and displacement ventilation (DV). To facilitate these types of CFD analyses, a novel workflow was created, and outlined in the thesis, that combined CFD and parallel processing to perform accelerated high-fidelity simulations. This workflow was then used to create a novel method of combining ML and CFD to create Risk Models to predict particle concentration-based risk for a Spatial-Temporal Risk (STR) model and a Regional-Temporal Risk (RTR) model. This methodology was applied to the Mixing Ventilation (MV) HVAC strategy. Both models (STR and RTR) showed a high degree of accuracy in predicting the risk, with a minimum PAE of 12.36 and 5.00 for the DNN and RBF models, respectively.

### 6.2. Contributions

#### 6.2.1. Comparative Study of Ventilation Strategies

The first contribution of this thesis was the comparative study of three ventilation strategies for indoor environments, stratum ventilation (SV), mixing ventilation (MV) and displacement ventilation (DV). The research results showed that the stratum ventilation strategy performed the worst regarding particle removal, removing, on average, 800 particles less than other strategies from a total pool of 4,844 particles. For each of the three performance indicators Heating removal index (HRE), index of mixing (IOM) and occupants' thermal perception (OTS), it was found that DV, MV and SV each performed the best in each category respectively.

#### 6.2.2. Advanced Combined Use of UDF, CFD and ML for Airborne Transmission

The second major contribution from this thesis was the development of a combined CFD, UDF, and ML approach for airborne transmission. Previous CFD and ML combined studies looked only at particle dispersion or contaminants. Traditional CFD methods rely on extensive computations, and it is difficult to combine experimental or empirical data into these CFD

simulations. The method developed in this work streamlined the process by combining CFD, UDF, and ML.

Users of this new CBPT method, developed in this thesis, can directly incorporate experimental data or experimental relations without having to start from scratch in ANSYS. Users can work with the direct quantities of simulations and easily obtain the associated data. Users can also more readily writing code if they have distinct areas to work with where they know they can perform computations or do write data to files.

#### 6.2.3. Resources for Researchers to use HPC and CFD

The third major contribution from this thesis was a step-by-step workflow and methodology for doing parallel processing on high-computational clusters (HPC). Specifically, the HPC used was ‘Compute Canada’ also known as the Digital Research Alliance of Canada. Users can easily follow the general steps to run many computational dynamics simulations by utilizing HPC clusters for things like CFD and MD. This study explains the basic concepts and techniques so that users can utilize parallel processing effectively.

#### 6.2.4. Development of Risk and Predictive Models

The last major contribution was the development of two distinct risk models and their use in creating two predictive models with the aid of Machine Learning. The first model was the Spatial-Temporal Risk (STR) model which uses risk factor values at respective cell locations over time to predict a 3D risk gradient over the domain. To create this STR model, the location, ambient temperature and time were used as input parameters and the particle risk factor was an output parameter. The second model was the Temporal-Regional Risk (TRR) model which filters the risk data for the highest relative risk values and determines the location and size of the region by approximating a rectangular prism region, referred to as a bounding box. To create this TRR model, the ambient temperature and time were used as input parameters and the particle risk factor, location and size of the bounded region were the output parameters. Both models (STR and TRR) were able to predict the relevant output parameters with reasonable percentage errors.

## 6.3. Recommendations for Future Work

### 6.3.1. Comparative Study of Ventilation Strategies

While this thesis investigated an office room with a single occupant, future studies can expand to larger office rooms, classrooms, or scenarios with more occupants, and more complicated geometries. The time range of the simulations analyzed could also be much smaller, since the regional model develops low variance quickly as particles reach the bounds of the room. This is because the values for the bounding box will converge, as they are physically constrained by the size of the room domain. Smaller times would aid in more accurate temporal correlation between the variables can be established in the ML model.

### 6.3.2. Advanced use of UDF and CFD

Many researchers have conducted experimental studies for various theoretical models that can model complex flow phenomena relating to the flows [118–120]. Theoretical models to describe complex processes, such as evaporation and condensation, are built- into CFD simulations are useful and often very good, but they rarely match experimental results. Future UDFs can couple experimental results using empirical models to other theoretical models describing processes, such as evaporation or condensation, within the CFD simulation. These UDFs could add correction factors based on empirical data to various elements of the CFD simulation, or the empirical data could be used to simulate simplified lower fidelity models. Other uses for CFD can be to simulate events, flow profiles, or other conditions. For example, the breathing velocities can be modeled from experimental data, or breathing velocities of other occupants can be simulated and controlled, allowing CFD simulations to simply simulate injection or breathing *events* of multiple participants without having to utilize GUI based design. Velocity profiles at various locations in the model can be manually added as boundary conditions. Injections can also be done at certain times or trigger due to complex logic conditions. The specific particles from specific injection events can also be tracked.

### 6.3.3. Resources for Researchers to use HPC and CFD

Many first-time users of HPC clusters face a myriad of issues when learning to use the technology. Future work could focus on developing resources for a broad range of researchers, so that they have an easy way to learn about the technology instead of having to comb through

disparate documentation to learn how to open a file or learn formatting for a specific cluster text input. Easy to use scripts, macros or GUIs can be created so that first-time users can have an easier time utilizing the HPC technology. Guides for creating parallel processing systems, using multiple desktop computers, for local researchers who may not have access to larger organizations computational resources could also be developed.

#### 6.3.4. Expanding Input and Parameters of ML

The ML model used in this study was limited in the variables explored. Future work could aim to incorporate a much wider range of input and output parameters in the ML model. For example, velocity, pressure, humidity, ventilation location, index patient location, occupant locations, among others.

## References

- [1] CDC Museum COVID-19 Timeline, (2023).  
<https://www.cdc.gov/museum/timeline/covid19.html> (accessed July 23, 2023).
- [2] Covid-19 Pandemic Timeline Fast Facts | CNN, (2021).  
<https://www.cnn.com/2021/08/09/health/covid-19-pandemic-timeline-fast-facts>.
- [3] SARS-CoV-2 and COVID-19 | BCM, (2023).  
<https://www.bcm.edu/departments/molecular-virology-and-microbiology/emerging-infections-and-biodefense/specific-agents/sars-cov-2-and-covid-19> (accessed July 30, 2023).
- [4] WHO Coronavirus (COVID-19) Dashboard | WHO Coronavirus (COVID-19) Dashboard With Vaccination Data, (2023). <https://covid19.who.int/> (accessed July 30, 2023).
- [5] WHO, Transmission of SARS-CoV-2: Implications for Infection Prevention Precautions. World Health Organization., MedRxiv. (2020) 1–21.  
<https://www.who.int/news-room/commentaries/detail/transmission-of-sars-cov-2-implications-for-infection-prevention-precautions> (accessed July 30, 2023).
- [6] S. Shao, D. Zhou, R. He, J. Li, S. Zou, K. Mallery, S. Kumar, S. Yang, J. Hong, Risk assessment of airborne transmission of COVID-19 by asymptomatic individuals under different practical settings, *J Aerosol Sci.* 151 (2021) 105661.  
<https://doi.org/10.1016/j.jaerosci.2020.105661>.
- [7] L. Zou, F. Ruan, M. Huang, L. Liang, H. Huang, Z. Hong, J. Yu, M. Kang, Y. Song, J. Xia, Q. Guo, T. Song, J. He, H.-L. Yen, M. Peiris, J. Wu, SARS-CoV-2 Viral Load in Upper Respiratory Specimens of Infected Patients, *New England Journal of Medicine.* 382 (2020) 1177–1179.  
[https://doi.org/10.1056/NEJMC2001737/SUPPL\\_FILE/NEJMC2001737\\_DISCLOSURES.PDF](https://doi.org/10.1056/NEJMC2001737/SUPPL_FILE/NEJMC2001737_DISCLOSURES.PDF).
- [8] Y. Yin, W. Xu, J.K. Gupta, A. Guity, P. Marmion, A. Manning, B. Gulick, X. Zhang, Q. Chen, Experimental Study on Displacement and Mixing Ventilation Systems for a Patient Ward, *HVAC&R Res.* 15 (2009) 1175–1191.  
<https://doi.org/10.1080/10789669.2009.10390885>.

- [9] W. Zhang, W. Zhang, K. Mizutani, H. Zhang, Decision-making analysis of ventilation strategies under complex situations: A numerical study, *Build Environ.* 206 (2021). <https://doi.org/10.1016/J.BUILDENV.2021.108217>.
- [10] Y. Choe, J. shup Shin, J. Park, E. Kim, N. Oh, K. Min, D. Kim, K. Sung, M. Cho, W. Yang, Inadequacy of air purifier for indoor air quality improvement in classrooms without external ventilation, *Build Environ.* 207 (2022) 108450. <https://doi.org/10.1016/J.BUILDENV.2021.108450>.
- [11] A. Khan, A NUMERICAL MODELLING APPROACH TO STUDY THE IMPACT OF VENTILATION CONFIGURATIONS ON AIRBORNE TRANSMISSION IN INDOOR ENVIRONMENTS, (2022).
- [12] C. Huan, F.H. Wang, Z. Lin, X.Z. Wu, Z.J. Ma, Z.H. Wang, L.H. Zhang, An experimental investigation into stratum ventilation for the cooling of an office with asymmetrically distributed heat gains, *Build Environ.* 110 (2016) 76–88. <https://doi.org/10.1016/J.BUILDENV.2016.09.031>.
- [13] A. Khan, F.M. Mohee, M. Freire-Gormaly, Modeling droplet and aerosol spread for minimizing airborne transmission of COVID-19 and other diseases in public spaces, *CSME Bulletin.* Fall (2020) 11–12. <http://csme-scg.mcgill.ca/sites/default/files/BULLETIN-FALL-2020.pdf>.
- [14] A. Henriques, N. Mounet, L. Aleixo, P. Elson, J. Devine, G. Azzopardi, M. Andreini, M. Rognien, N. Tarocco, J. Tang, Modelling airborne transmission of SARS-CoV-2 using CARA: risk assessment for enclosed spaces, (2022). <https://doi.org/10.1098/rsfs.2021.0076>.
- [15] P. Vidmar, S. Petelin, Application of CFD Method for Risk Assessment In Road Tunnels, *Engineering Applications of Computational Fluid Mechanics.* 1 (2007) 273–287. <https://doi.org/10.1080/19942060.2007.11015199>.
- [16] M.L. Hosain, R.B. Fdhila, Literature Review of Accelerated CFD Simulation Methods towards Online Application, *Energy Procedia.* 75 (2015) 3307–3314. <https://doi.org/10.1016/j.egypro.2015.07.714>.
- [17] H.M. Zoka, M. Moshfeghi, H. Bordbar, P.A. Mirzaei, Y. Sheikhnejad, A cfd approach for risk assessment based on airborne pathogen transmission, *Atmosphere (Basel).* 12 (2021). <https://doi.org/10.3390/atmos12080986>.

- [18] N. Morozova, F.X. Trias, R. Capdevila, C.D. Pérez-Segarra, A. Oliva, On the feasibility of affordable high-fidelity CFD simulations for indoor environment design and control, *Build Environ.* 184 (2020) 107144.  
<https://doi.org/10.1016/j.buildenv.2020.107144>.
- [19] O. Fawwaz Alrebi, B. Obeidat, I. Atef Abdallah, E.F. Darwish, A. Amhamed, Airflow dynamics in an emergency department: A CFD simulation study to analyse COVID-19 dispersion, *Alexandria Engineering Journal.* (2021).  
<https://doi.org/10.1016/j.aej.2021.08.062>.
- [20] H. Liu, S. He, L. Shen, J. Hong, Simulation-based study of COVID-19 outbreak associated with air-conditioning in a restaurant, *Physics of Fluids.* 33 (2021) 23301.  
<https://doi.org/10.1063/5.0040188/1032688>.
- [21] K. Talaat, M. Abuhegazy, O.A. Mahfoze, O. Anderoglu, S. V. Poroseva, Simulation of aerosol transmission on a Boeing 737 airplane with intervention measures for COVID-19 mitigation, *Physics of Fluids.* 33 (2021) 33312.  
<https://doi.org/10.1063/5.0044720>.
- [22] V. Vuorinen, M. Aarnio, M. Alava, V. Alopaeus, N. Atanasova, M. Auvinen, N. Balasubramanian, H. Bordbar, P. Erästö, R. Grande, N. Hayward, A. Hellsten, S. Hostikka, J. Hokkanen, O. Kaario, A. Karvinen, I. Kivistö, M. Korhonen, R. Kosonen, J. Kuusela, S. Lestinen, E. Laurila, H.J. Nieminen, P. Peltonen, J. Pokki, A. Puisto, P. Råback, H. Salmenjoki, T. Sironen, M. Österberg, Modelling aerosol transport and virus exposure with numerical simulations in relation to SARS-CoV-2 transmission by inhalation indoors, *Saf Sci.* 130 (2020) 104866.  
<https://doi.org/10.1016/j.ssci.2020.104866>.
- [23] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Downloaded from [www.annualreviews.org](http://www.annualreviews.org) Access provided by 70.30.77.64 on 08/13/23. For personal use only, *Annu. Rev. Fluid Mech.* 2020. 52 (2019) 477–508.  
<https://doi.org/10.1146/annurev-fluid-010719>.
- [24] S.L. Brunton, Applying machine learning to study fluid mechanics, *Acta Mechanica Sinica.* 37 (2021) 1718–1726. <https://doi.org/10.1007/s10409-021-01143-6>.

- [25] R. Rai, M.K. Tiwari, D. Ivanov, A. Dolgui, Machine learning in manufacturing and industry 4.0 applications, *Int J Prod Res.* 59 (2021) 4773–4778.  
<https://doi.org/10.1080/00207543.2021.1956675>.
- [26] N. Da Silva, M. Oliveira, D. Cordeiro, Impacts, Potentials, and Trends of Machine Learning in Industry 4.0: An Overview, (2022).  
<https://doi.org/10.52591/LXAI202007136>.
- [27] Y. M. Banadaki, On the Use of Machine Learning for Additive Manufacturing Technology in Industry 4.0, *JOURNAL OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY.* 7 (2019).  
<https://doi.org/10.15640/JCSIT.V7N2A7>.
- [28] R. Rai, M.K. Tiwari, D. Ivanov, A. Dolgui, Machine learning in manufacturing and industry 4.0 applications, *Int J Prod Res.* 59 (2021) 4773–4778.  
<https://doi.org/10.1080/00207543.2021.1956675>.
- [29] M. Freire-Gormaly, A.M. Bilton, Design of photovoltaic powered reverse osmosis desalination systems considering membrane fouling caused by intermittent operation, *Renew Energy.* 135 (2019) 108–121.  
<https://doi.org/10.1016/j.renene.2018.11.065>.
- [30] A.M. Gilau, M.J. Small, Designing cost-effective seawater reverse osmosis system under optimal energy options, *Renew Energy.* 33 (2008) 617–630.  
<https://doi.org/10.1016/j.renene.2007.03.019>.
- [31] S. Lee, S. Myung, J. Hong, D. Har, Reverse osmosis desalination process optimized for maximum permeate production with renewable energy, *Desalination.* 398 (2016) 133–143. <https://doi.org/10.1016/j.desal.2016.07.018>.
- [32] M. Freire-Gormaly, Experimental Characterization of Membrane Fouling under Intermittent Operation and Its Application to the Optimization of Solar Photovoltaic Powered Reverse Osmosis Drinking Water Treatment Systems, PhD, University of Toronto, 2018.
- [33] D.R. Ferreira, J. Contributors, Using HPC infrastructures for deep learning applications in fusion research, *Plasma Physics and Controlled Fusion*, (2021), <https://doi.org/10.1088/1361-6587/ac0a3b>

- [34] V. Jadhao, J. Kadupitiya, Integrating Machine Learning with HPC-driven Simulations for Enhanced Student Learning, (2020), 2020 IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC), <https://doi.org/10.1109/EduHPC51895.2020.00009>
- [35] S.J. Lawson, M. Woodgate, R. Steijl, G.N. Barakos, High performance computing for challenging problems in computational fluid dynamics, *Progress in Aerospace Sciences*. 52 (2012) 19–29. <https://doi.org/10.1016/J.PAEROSCI.2012.03.004>.
- [36] L. Morawska, D.K. Milton, It Is Time to Address Airborne Transmission of Coronavirus Disease 2019 (COVID-19), *Clinical Infectious Diseases*. 71 (2020) 2311–2313. <https://doi.org/10.1093/cid/ciaa939>.
- [37] Y. Yan, X. Li, Y. Shang, J. Tu, Evaluation of airborne disease infection risks in an airliner cabin using the Lagrangian-based Wells-Riley approach, *Build Environ*. 121 (2017) 79–92. <https://doi.org/10.1016/j.buildenv.2017.05.013>.
- [38] M. Konstantinov, D. Schmeling, C. Wagner, Numerical simulation of the aerosol formation and spreading in an air-conditioned train compartment, *J Aerosol Sci*. 170 (2023) 106139. <https://doi.org/10.1016/J.JAEROSCI.2023.106139>.
- [39] Y. Sheikhejad, R. Aghamolaei, M. Fallahpour, H. Motamedi, M. Moshfeghi, P.A. Mirzaei, H. Bordbar, Airborne and aerosol pathogen transmission modeling of respiratory events in buildings: An overview of computational fluid dynamics, *Sustain Cities Soc*. 79 (2022). <https://doi.org/10.1016/j.scs.2022.103704>.
- [40] R. Vinuesa, S.L. Brunton, The Potential of Machine Learning to Enhance Computational Fluid Dynamics, (2021) 1–13. <http://arxiv.org/abs/2110.02085>.
- [41] K.E. Ringstad, K. Banasiak, Å. Ervik, A. Hafner, Machine learning and CFD for mapping and optimization of CO<sub>2</sub> ejectors, *Appl Therm Eng*. 199 (2021) 117604. <https://doi.org/10.1016/J.APPLTHERMALENG.2021.117604>.
- [42] M. Savarese, A. Cuoci, W. De Paepe, A. Parente, Machine learning clustering algorithms for the automatic generation of chemical reactor networks from CFD simulations, *Fuel*. 343 (2023) 127945. <https://doi.org/10.1016/J.FUEL.2023.127945>.

- [43] G. Gui, K.I. Roumeliotis, N.D. Tselikas, ChatGPT and Open-AI Models: A Preliminary Review, *Future Internet* 2023, Vol. 15, Page 192. 15 (2023) 192. <https://doi.org/10.3390/FI15060192>.
- [44] M.-A. Sicilia, N. Bessis, M. Trovati, H. Hassani, E.S. Silva, The Role of ChatGPT in Data Science: How AI-Assisted Conversational Interfaces Are Revolutionizing the Field, *Big Data and Cognitive Computing* 2023, Vol. 7, Page 62. 7 (2023) 62. <https://doi.org/10.3390/BDCC7020062>.
- [45] M.J. Skibniewski, H. Liu, Y. Lei, S.A. Prieto, E.T. Mengiste, B. García De Soto, Investigating the Use of ChatGPT for the Scheduling of Construction Projects, *Buildings* 2023, Vol. 13, Page 857. 13 (2023) 857. <https://doi.org/10.3390/BUILDINGS13040857>.
- [46] F.M. Mohee, A. Al-Mayah, A. Plumtree, Anchors for CFRP plates: State-of-the-art review and future potential, *Compos B Eng.* 90 (2016) 432–442. <https://doi.org/10.1016/j.compositesb.2016.01.011>.
- [47] Y. Zou, X. Zhao, Q. Chen, Comparison of STAR-CCM+ and ANSYS Fluent for simulating indoor airflows, *Build Simul.* 11 (2018) 165–174. <https://doi.org/https://doi.org/10.1007/s12273-017-0378-8>.
- [48] I. Aramendia, U. Fernandez-Gamiz, A. Lopez-Arraiza, C. Rey-Santano, V. Mielgo, F.J. Basterretxea, J. Sancho, M.A. Gomez-Solaetxe, Experimental and numerical modeling of aerosol delivery for preterm infants, *Int J Environ Res Public Health.* 15 (2018). <https://doi.org/10.3390/ijerph15030423>.
- [49] F.M. Mohee, A. Al-Mayah, Development of an innovative prestressing CFRP plate anchor: Numerical modelling and parametric study, *Compos Struct.* 177 (2017) 1–12. <https://doi.org/10.1016/j.compstruct.2016.12.039>.
- [50] M. Freire-Gormaly, The Pore Structure of Indiana Limestone and Pink Dolomite for the Modeling of Carbon Dioxide in Geologic Carbonate Rock Formations, MASc., University of Toronto, 2013. <http://hdl.handle.net/1807/42840>.
- [51] P. Altschuh, Y.C. Yabansu, J. Hötzer, M. Selzer, B. Nestler, S.R. Kalidindi, Data science approaches for microstructure quantification and feature identification in porous membranes, *J Memb Sci.* 540 (2017) 88–97. <https://doi.org/10.1016/j.memsci.2017.06.020>.

- [52] M. Freire-Gormaly, J.S. Ellis, A. Bazylak, H.L. MacLean, Comparing thresholding techniques for quantifying the dual porosity of Indiana Limestone and Pink Dolomite, Microporous and Mesoporous Materials. 207 (2015) 84–89. <https://doi.org/10.1016/j.micromeso.2015.01.002>.
- [53] F.M. Mohee, C. Miller, Climatology of thunderstorms for North Dakota, 2002–06, J Appl Meteorol Climatol. 49 (2010) 1881–1890. <https://doi.org/10.1175/2010JAMC2400.1>.
- [54] M. Leer, A. Kempf, Fast Flow Field Estimation for Various Applications with A Universally Applicable Machine Learning Concept, Flow Turbul Combust. 107 (2021) 175–200. <https://doi.org/10.1007/s10494-020-00234-x>.
- [55] F.M. Mohee, Development, Analysis and Testing of Innovative Mechanical Prestressing Anchors for CFRP Plates for Structural Rehabilitation and Retrofitting, University of Waterloo (UWSpace), 2017. <http://hdl.handle.net/10012/11207>.
- [56] C.J. Noakes, C.B. Beggs, P.A. Sleight, K.G. Kerr, Modelling the transmission of airborne infections in enclosed spaces, Epidemiol Infect. 134 (2006) 1082–1091. <https://doi.org/10.1017/S0950268806005875>.
- [57] J.A. Sleight, Applying the Wells-Riley equation to the risk of airborne infection in hospital environments: The importance of stochastic and proximity effects, (2008). <https://eprints.whiterose.ac.uk/> (accessed August 13, 2023).
- [58] A.J. Edwards, L. Benson, Z. Guo, M. López-García, C.J. Noakes, D. Peckham, M.F. King, A mathematical model for assessing transient airborne infection risks in a multi-zone hospital ward, Build Environ. 238 (2023) 110344. <https://doi.org/10.1016/J.BUILDENV.2023.110344>.
- [59] A. Karimipour, A. Amini, M. Nouri, A. D’Orazio, R. Sabetvand, M. Hekmatifar, A. Marjani, Q. vu Bach, Molecular dynamics performance for coronavirus simulation by C, N, O, and S atoms implementation dreiding force field: drug delivery atomic interaction in contact with metallic Fe, Al, and steel, Comput Part Mech. 8 (2021) 737–749. <https://doi.org/10.1007/s40571-020-00367-w>.
- [60] R. Mittal, R. Ni, J.-H. Seo, The flow physics of COVID-19, (2020). <https://doi.org/10.1017/jfm.2020.330>.

- [61] C.C. Wang, K.A. Prather, J. Sznitman, J.L. Jimenez, S.S. Lakdawala, Z. Tufekci, L.C. Marr, Airborne transmission of respiratory viruses, *Science* (1979). 373 (2021). <https://doi.org/10.1126/SCIENCE.ABD9149>.
- [62] M.E. Rosti, S. Olivieri, M. Cavaola, A. Seminara, & A. Mazzino, Fluid dynamics of COVID-19 airborne infection suggests urgent data for a scientific design of social distancing, (2020). <https://doi.org/10.1038/s41598-020-80078-7>.
- [63] S. Chaudhuri, S. Basu, A. Saha, Physics of Fluids ARTICLE scitation.org/journal/phf Analyzing the dominant SARS-CoV-2 transmission routes toward an ab initio disease spread model, Cite as: *Phys. Fluids*. 32 (2020) 123306. <https://doi.org/10.1063/5.0034032>.
- [64] R. Tellier, Y. Li, B.J. Cowling, J.W. Tang, Recognition of aerosol transmission of infectious agents: A commentary, *BMC Infect Dis*. 19 (2019) 1–9. <https://doi.org/10.1186/S12879-019-3707-Y/FIGURES/1>.
- [65] WHO-convened global study of origins of SARS-CoV-2: China Part, (2021). <https://www.who.int/publications/i/item/who-convened-global-study-of-origins-of-sars-cov-2-china-part> (accessed August 12, 2023).
- [66] A. Shadloo-Jahromi, O. Bavi, M. Hossein Heydari, M. Kharati-Koopae, Z. Avazzadeh, Dynamics of respiratory droplets carrying SARS-CoV-2 virus in closed atmosphere, *Results Phys*. 19 (2020) 103482. <https://doi.org/10.1016/j.rinp.2020.103482>.
- [67] M. Mesgarpour, J. Mohebbi, N. Abad, R. Alizadeh, S. Wongwises, M.H. Doranehgard, S. Ghaderi, N. Karimi, Prediction of the spread of Corona-virus carrying droplets in a bus-A computational based artificial intelligence approach, (2021). <https://doi.org/10.1016/j.jhazmat.2021.125358>.
- [68] J.P. Duguid, The size and the duration of air-carriage of respiratory droplets and droplet-nuclei, *J Hyg (Lond)*. 44 (1946) 471. <https://doi.org/10.1017/S0022172400019288>.
- [69] P. Mecnas, R.T. da Rosa Moreira Bastos, A.C. Rosário Vallinoto, D. Normando, Effects of temperature and humidity on the spread of COVID-19: A systematic review, *PLoS One*. 15 (2020). <https://doi.org/10.1371/JOURNAL.PONE.0238339>.

- [70] L. Morawska, Droplet fate in indoor environments, or can we prevent the spread of infection?, *Indoor Air*. 16 (2006) 335–347. <https://doi.org/10.1111/J.1600-0668.2006.00432.X>.
- [71] X. Yang, C. Ou, H. Yang, L. Liu, T. Song, M. Kang, H. Lin, J. Hang, Transmission of pathogen-laden expiratory droplets in a coach bus, *J Hazard Mater*. 397 (2020) 122609. <https://doi.org/10.1016/J.JHAZMAT.2020.122609>.
- [72] Y. Yan, X. Li, Y. Shang, J. Tu, Evaluation of airborne disease infection risks in an airliner cabin using the Lagrangian-based Wells-Riley approach, *Build Environ*. 121 (2017) 79–92. <https://doi.org/10.1016/J.BUILDENV.2017.05.013>.
- [73] F. Cui, X. Geng, O. Zervaki, D.D. Dionysiou, J. Katz, S.-J. Haig, M. Boufadel, Transport and Fate of Virus-Laden Particles in a Supermarket: Recommendations for Risk Reduction of COVID-19 Spreading, *Journal of Environmental Engineering*. 147 (2021). [https://doi.org/10.1061/\(ASCE\)EE.1943-7870.0001870](https://doi.org/10.1061/(ASCE)EE.1943-7870.0001870).
- [74] A. W. Date, Introduction to Computational Fluid Dynamics Nomenclature Symbols, 2012, <https://doi.org/10.1017/CBO9780511808975>
- [75] York University | Right the Future, (2023), <https://www.yorku.ca/> (accessed August 13, 2023).
- [76] G. Taguchi, M.S. Phadke, Quality Engineering through Design Optimization, in: *Quality Control, Robust Design, and the Taguchi Method*, Springer-Verlag, 1984: pp. 77–96. <http://link.springer.com/book/10.1007/978-1-4684-1472-1>.
- [77] H.-G. Beyer, B. Sendhoff, Robust optimization – A comprehensive survey, *Comput Methods Appl Mech Eng*. 196 (2007) 3190–3218. <https://doi.org/10.1016/j.cma.2007.03.003>.
- [78] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
- [79] D. Connolly, H. Lund, B.V. Mathiesen, M. Leahy, A review of computer tools for analysing the integration of renewable energy into various energy systems, *Appl Energy*. 87 (2010) 1059–1082. <https://doi.org/10.1016/j.apenergy.2009.09.026>.
- [80] W. Cao, Q. Liu, Y. Wang, I.M. Mujtaba, Modeling and simulation of VMD desalination process by ANN, *Comput Chem Eng*. 84 (2016) 96–103. <https://doi.org/10.1016/j.compchemeng.2015.08.019>.

- [81] L. Horrigan, M. Freire-Gormaly, Modelling the effects of ultrasonic sonification on reverse osmosis feed channel temperature, *Desalination*. 521 (2022) 115332. <https://doi.org/10.1016/j.desal.2021.115332>.
- [82] M. Getu, S. Mahadzir, M. Lee, Profit optimization for chemical process plant based on a probabilistic approach by incorporating material flow uncertainties, *Comput Chem Eng*. 59 (2013) 186–196. <https://doi.org/10.1016/j.compchemeng.2013.05.026>.
- [83] C.T. Mackay, D. Nowell, Informed machine learning methods for application in engineering: A review, *Proc Inst Mech Eng C J Mech Eng Sci*. (2023). <https://doi.org/10.1177/09544062231164575>.
- [84] A. Daw, R. Quinn Thomas, C.C. Carey, J.S. Read, A.P. Appling, A. Karpatne, Physics-Guided Architecture (PGA) of Neural Networks for Quantifying Uncertainty in Lake Temperature Modeling, (2020). <https://epubs.siam.org/terms-privacy> (accessed August 13, 2023).
- [85] R.H. Jhaveri, A. Revathi, K. Ramana, R. Raut, R.K. Dhanaraj, A Review on Machine Learning Strategies for Real-World Engineering Applications, (2022). <https://doi.org/10.1155/2022/1833507>.
- [86] S.L. Brunton, Applying machine learning to study fluid mechanics, *Acta Mechanica Sinica/Lixue Xuebao*. 37 (2021) 1718–1726. <https://doi.org/10.1007/S10409-021-01143-6/METRICS>.
- [87] IBM, About Linear Regression, (2023). <https://www.ibm.com/topics/linear-regression> (accessed August 14, 2023).
- [88] IBM, What is a Decision Tree, (2023). <https://www.ibm.com/topics/decision-trees> (accessed August 14, 2023).
- [89] IBM, What is Random Forest, (2023). <https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems> (accessed August 14, 2023).
- [90] SciKit, Support Vector Machines — scikit-learn 1.3.0 documentation, (2023). <https://scikit-learn.org/stable/modules/svm.html> (accessed August 14, 2023).
- [91] IBM, About SVM, (2023). <https://www.ibm.com/docs/en/spss-modeler/saas?topic=models-about-svm> (accessed August 14, 2023).

- [92] BMC, What's a Deep Neural Network? Deep Nets Explained, (2023).  
<https://www.bmc.com/blogs/deep-neural-network/> (Accessed August 14, 2023).
- [93] A. Marcato, G. Boccardo, D. Marchisio, A computational workflow to study particle transport and filtration in porous media: Coupling CFD and deep learning, *Chemical Engineering Journal*. 417 (2021) 128936.  
<https://doi.org/10.1016/J.CEJ.2021.128936>.
- [94] MATLAB, What Is a Convolutional Neural Network? | 3 things you need to know, (2023). <https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html> (accessed August 14, 2023).
- [95] IBM, What are Recurrent Neural Networks?, (2023),  
[https://www.ibm.com/topics/recurrent-neural-networks#:~:text=A%20recurrent%20neural%20network%20\(RNN,data%20or%20time%20series%20data](https://www.ibm.com/topics/recurrent-neural-networks#:~:text=A%20recurrent%20neural%20network%20(RNN,data%20or%20time%20series%20data) (accessed August 14, 2023).
- [96] A. Vidhya, What is LSTM? Introduction to Long Short-Term Memory, (2021).  
<https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/> (accessed August 14, 2023).
- [97] IBM, Generative adversarial networks explained, (2023),  
<https://developer.ibm.com/articles/generative-adversarial-networks-explained/> (accessed August 14, 2023).
- [98] J. Brownlee, Machine Learning Mastery, (2023).  
<https://machinelearningmastery.com/> (accessed August 13, 2023).
- [99] P.A. Mirzaei, M. Moshfeghi, H. Motamedi, Y. Sheikhejad, H. Bordbar, A simplified tempo-spatial model to predict airborne pathogen release risk in enclosed spaces: An Eulerian-Lagrangian CFD approach, *Build Environ*. 207 (2022) 108428.  
<https://doi.org/10.1016/j.buildenv.2021.108428>.
- [100] A. Kabanshi, H. Wigö, R. Ljung, P. Sörqvist, Experimental evaluation of an intermittent air supply system – Part 2: Occupant perception of thermal climate, *Build Environ*. 108 (2016) 99–109.  
<https://doi.org/10.1016/J.BUILDENV.2016.08.025>.
- [101] P.G. Saffman, The lift on a small sphere in a slow shear flow, *J Fluid Mech*. 22 (1965) 385–400. <https://doi.org/10.1017/S0022112065000824>.

- [102] CCDB, Documentation, (2023). <https://ccdb.alliancecan.ca/> (accessed August 12, 2023).
- [103] CCDB, Getting started Documentation, (2023).  
[https://docs.alliancecan.ca/wiki/Getting\\_started](https://docs.alliancecan.ca/wiki/Getting_started) (accessed August 12, 2023).
- [104] ANSYS, ANSYS FLUENT 12.0 UDF Manual - 7.1 Overview of Parallel ANSYS FLUENT, (2021).  
<https://www.afs.enea.it/project/neptunius/docs/fluent/html/udf/node213.htm>  
(accessed August 13, 2023).
- [105] CAN, Alliance Documentation, (2023),  
[https://docs.alliancecan.ca/wiki/Technical\\_documentation](https://docs.alliancecan.ca/wiki/Technical_documentation) (accessed August 27, 2023).
- [106] F. Mohee, A. Al-Mayah, A. Plumtree, Friction Characteristics of CFRP Plates in Contact with Copper Plates under High Contact Pressure, *Journal of Composites for Construction*. (2016) 4016022. [https://doi.org/10.1061/\(ASCE\)CC.1943-5614.0000673](https://doi.org/10.1061/(ASCE)CC.1943-5614.0000673).
- [107] F.M. Mohee, A. Al-Mayah, A. Plumtree, Friction Characteristics of CFRP Plates in Contact with Copper Plates under High Contact Pressure, *Journal of Composites for Construction*. 20 (2016) 1–10. [https://doi.org/10.1061/\(ASCE\)CC.1943-5614.0000673](https://doi.org/10.1061/(ASCE)CC.1943-5614.0000673).
- [108] F.M. Mohee, A. Al-Mayah, A. Plumtree, Development of a novel prestressing anchor for CFRP plates: Experimental investigations, *Compos Struct*. 176 (2017) 20–32. <https://doi.org/10.1016/j.compstruct.2017.05.011>.
- [109] M. Freire-Gormaly, A.M. Bilton, Impact of intermittent operation on reverse osmosis membrane fouling for brackish groundwater desalination systems, *J Memb Sci*. 583 (2019) 220–230. <https://doi.org/10.1016/j.memsci.2019.04.010>.
- [110] M. Freire-Gormaly, A.M. Bilton, Experimental quantification of the effect of intermittent operation on membrane performance of solar powered reverse osmosis desalination systems, *Desalination*. (2017).  
<https://doi.org/http://www.sciencedirect.com/science/article/pii/S0011916417311104>.

- [111] M. Freire-Gormaly, A.M. Bilton, An experimental system for characterization of membrane fouling of solar photovoltaic reverse osmosis systems under intermittent operation, *Desalination Water Treat.* 73 (2017) 54–63.  
<https://doi.org/10.5004/dwt.2017.20391>.
- [112] F.M. Mohee, A. Al-mayah, Effect of modulus of elasticity and thickness of the CFRP plate on the performance of a novel anchor for structural retrofitting and rehabilitation applications, *Eng Struct.* 153 (2017) 302–316.  
<https://doi.org/10.1016/j.engstruct.2017.09.057>.
- [113] F.M. Mohee, A. Al-Mayah, Effect of barrel, wedge material and thickness on composite plate anchor performance through analytical, finite element, experimental and 3D prototype investigations, *Eng Struct.* 175 (2018) 138–154.
- [114] E. Trigell, CFD-simulations of urea-water spray in an after-treatment system using Star-CCM+, 2018. <http://www.diva-portal.org/smash/get/diva2:1306981/FULLTEXT01.pdf> (accessed August 28, 2023).
- [115] M. Lisicki, W. Lubitz, G.W. Taylor, Optimal design and operation of Archimedes screw turbines using Bayesian optimization, *Appl Energy.* 183 (2016) 1404–1417.  
<https://doi.org/10.1016/j.apenergy.2016.09.084>.
- [116] M. Freire-Gormaly, J.S. Ellis, H.L. MacLean, A. Bazylak, Pore Structure Characterization of Indiana Limestone and Pink Dolomite from Pore Network Reconstructions, *Oil & Gas Science and Technology.* 71 (2015) 33.  
<https://doi.org/10.2516/ogst/2015004>.
- [117] Windows, MobaXterm free Xserver and tabbed SSH client for Windows, (2023).  
<https://mobaxterm.mobatek.net/> (accessed August 27, 2023).
- [118] M. Surendran, T.N.C. Anand, S. Bakshi, Experimental investigation of the evaporation behavior of urea-water-solution droplets exposed to a hot air stream, *AIChE Journal.* 66 (2020). <https://doi.org/10.1002/AIC.16845>.
- [119] L. Liu, J. Wei, Y. Li, A. Ooi, Evaporation and dispersion of respiratory droplets from coughing, *Indoor Air.* 27 (2017) 179–190. <https://doi.org/10.1111/INA.12297>.

- [120] X. Xie, Y. Li, A.T.Y. Chwang, P.L. Ho, W.H. Seto, How far droplets can move in indoor environments – revisiting the Wells evaporation–falling curve, *Indoor Air*. 17 (2007) 211–225. <https://doi.org/10.1111/J.1600-0668.2007.00469.X>.

## Appendix A – Bash code to open Fluent Cas File

```
#!/bin/bash

#SBATCH --time=00-30:00      # Time (DD-HH:MM)
#SBATCH --mem=16G           # Total Memory (set to 0 for all node memory)
#SBATCH --ntasks=4          # Number of cores
#SBATCH --nodes=1           # Do not change (multi-node not supported)
##SBATCH --exclusive        # Uncomment for scaling testing
##SBATCH --constraint=broadwell # Applicable to graham or cedar

module load StdEnv/2020 ansys/2021R2 # OR newer Ansys modules

if [ "$SLURM_NNODES" == 1 ]; then
    MEMPAR=0                # Set to 0 for SMP (shared memory parallel)
else
    MEMPAR=1                # Set to 1 for DMP (distributed memory parallel)
fi

rm -fv *_files/.lock
MWFILE=~/.mw/Application\ Data\Ansys/`basename $(find $EBROOTANSYS/v* -maxdepth 0
-type d)`/SolveHandlers.xml
sed -re "s/(.AnsysSolution>+)[a-zA-Z0-9]*(<\Distribute.)^1$MEMPAR\2/" -i "$MWFILE"
sed -re "s/(.Processors>+)[a-zA-Z0-9]*(<\MaxNumber.)^1$SLURM_NTASKS\2/" -i
"$MWFILE"
sed -i "s!UserConfigured=\"0\"!UserConfigured=\"1\"!g" "$MWFILE"

export KMP_AFFINITY=disabled
export I_MPI_HYDRA_BOOTSTRAP=ssh

# Get the journal and .cas.h5 file from the command line arguments
JOURNAL_FILE=$1
```

```
CAS_H5_FILE=$2

fluent 3ddp -g -i $JOURNAL_FILE -t$SLURM_NTASKS -cnf=$SLURM_NODELIST -ssh <<
EOF
/file/read-case $CAS_H5_FILE
/solve/initialize/compute-defaults/flow
/solve/iterate 100
/file/write-data solution.dat
EOF
```

## Appendix B – Bash Code to Open Workbench Project File

```
#!/bin/bash

#SBATCH --time=00-03:00      # Time (DD-HH:MM)
#SBATCH --mem=16G           # Total Memory (set to 0 for all node memory)
#SBATCH --ntasks=4          # Number of cores
#SBATCH --nodes=1           # Do not change (multi-node not supported)
##SBATCH --exclusive        # Uncomment for scaling testing
##SBATCH --constraint=broadwell # Applicable to graham or cedar

module load StdEnv/2020 ansys/2021R2 # OR newer Ansys modules

if [ "$SLURM_NNODES" == 1 ]; then
    MEMPAR=0                # Set to 0 for SMP (shared memory parallel)
else
    MEMPAR=1                # Set to 1 for DMP (distributed memory parallel)
fi

rm -fv *_files/.lock
MWFILE=~/.mw/Application\ Data/Ansys/`basename $(find $EBROOTANSYS/v* -maxdepth 0 -type d)/SolveHandlers.xml`
sed -re "s/(.AnsysSolution>+)[a-zA-Z0-9]*(<\/Distribute.)^1$MEMPAR\2/" -i "$MWFILE"
sed -re "s/(.Processors>+)[a-zA-Z0-9]*(<\/MaxNumber.)^1$SLURM_NTASKS\2/" -i "$MWFILE"
sed -i "s!UserConfigured=\"0\"!UserConfigured=\"1\"!g" "$MWFILE"

export KMP_AFFINITY=disabled
export I_MPI_HYDRA_BOOTSTRAP=ssh

runwb2 -B -E "Update();Save(Overwrite=True)" -F $1
```

## Appendix C – UDF Code

```
#include "udf.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <unistd.h>

#define PW 0.1 //Worth of a single particle
#define MAX_P_Diam (100*10^-6)
#define MIN_P_Diam (1*10^-6)

#define sig_A (0.1)
#define sig_B (0.85)
#define sig_K (0.6)
#define sig_centre (6)

char *fileNamePrefix = "UDM_DATA_A1_t";
char *folderName = "UDM_DATA_A1";

real save_interval = 1; //Interval between saves in seconds

static int file_num = 0; //current number of files saved
static int auto_save = 1; //Flag to toggle auto saving
int ZONE_ID = 14;

//Function Declarations
double scalingFactor(double particle_size, double max_size, double min_size);
void moveFileToFolder(char *fileName, char *folderName);
void encode_2d_to_1d(real **arr_2d, int rows, int cols, real *encoded);
void decode_1d_to_2d(real *encoded, int rows, int cols, real **decoded);
void free_2d_array(real **arr, int rows);
real** create_2d_array(int rows, int cols);

real** create_2d_array(int rows, int cols) {
    real **arr = (real **)malloc(rows * sizeof(real *));
    for (int i = 0; i < rows; i++) {
        arr[i] = (real *)malloc(cols * sizeof(real));
    }
    return arr;
}

// Function to encode a 2D array into a 1D array
void encode_2d_to_1d(real **arr_2d, int rows, int cols, real *encoded) {
```

```

for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        encoded[i * cols + j] = arr_2d[i][j];
    }
}
}

// Function to decode a 1D array into a 2D array
void decode_1d_to_2d(real *encoded, int rows, int cols, real **decoded) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            decoded[i][j] = encoded[i * cols + j];
        }
    }
}

void moveFileToFolder(char *fileName, char *folderName) {
    // Check if the folder exists, and create it if it doesn't
    struct stat st;
    if (stat(folderName, &st) == -1) {
        // Folder doesn't exist, create it
        if (mkdir(folderName, 0777) != 0) {
            fprintf(stderr, "Failed to create folder\n");
            return;
        }
    }

    // Build the new file path
    char newFilePath[256]; // Adjust the buffer size as needed
    snprintf(newFilePath, sizeof(newFilePath), "%s/%s", folderName, fileName);

    // Move the file
    if (rename(fileName, newFilePath) != 0) {
        perror("rename");
        exit(EXIT_FAILURE);
    }
}

double scalingFactor(double particle_size, double max_size, double min_size) {
    // Calculate the scaled particle size between 0 and 1
    double scaled_particle_size = (particle_size - min_size) / (max_size - min_size);

    // Calculate the scaling factor using the sigmoid function
    double scaling_factor = ((1.0 + sig_A) / (sig_B + exp(-sig_K * (scaled_particle_size - sig_centre))));

    return scaling_factor;
}

void free_2d_array(real **arr, int rows) {

```

```

    for (int i = 0; i < rows; i++) {
        free(arr[i]);
    }
    free(arr);
}

//HOOKS
DEFINE_EXECUTE_ON_LOADING(loading_func, libname)
{
    #if !RP_NODE
    /* either serial or host process is involved */
    Message("*****\n");
    Message("\n\n");

    if(!auto_save)
    {
        Message("Autosave is Disabled\n");
    }
    else
    {
        Message("Autosave is Enabled\n with an interval of %g s",save_interval);
    }

    Message("\n\n");
    Message("*****\n");
    #endif
}

DEFINE_INIT(my_init, domain)
{
    #if !RP_NODE
    file_num = 1;
    #endif

    #if !RP_HOST
    /* Initialize UDM values for all cells of each node */
    cell_t cell;
    Thread *thread;

    thread_loop_c(thread, domain)
    {
        begin_c_loop_all(cell,thread)
        {
            C_UDMI(cell, thread, 0) = 0.0; /* Initialize UDM value to zero */
            C_UDMI(cell, thread, 1) = 0.0;
        }
        end_c_loop_all(cell,thread)
    }
}

```

```

    }
    Message("\nInitilazed all UDMs on node %i\n",myid);
#endif
}

DEFINE_DPM_SCALAR_UPDATE(track_particles, cell, thread, initialize, particle)
{
    #if !RP_HOST

    real p_diam = 0;
    double SF = 0;

    p_diam = P_DIAM0(particle);
    SF = scalingFactor(p_diam, MAX_P_Diam, MIN_P_Diam);

    C_UDMI(cell, thread, 1) += PW; /* Increment UDM value by PW for each particle */
    C_UDMI(cell, thread, 0) += (PW * SF); /* Increment UDM value by PW*scaling factor for each particle */
    #endif
}

DEFINE_ON_DEMAND(toggle_autosave)
{
    #if !RP_NODE
    char toggle[10];
    auto_save = (auto_save==1) ? 0 : 1;
    switch (auto_save){
        case 0: Message("Autosave is now *Disabled*\n");
            break;
        case 1: Message("Autosave is now *Enabled*\n");
            break;
    }
    #endif
}

DEFINE_ON_DEMAND(print_udm_values)
{
    #if !RP_HOST
    Thread *thread;
    cell_t cell;
    real xc[ND_ND];

    Domain *domain; /* domain is declared as a variable */
    domain = Get_Domain(1); /* returns fluid domain pointer */
    //determine the number of useful cells for each node
    #endif

```

```

//determine file name
#ifdef !RP_NODE
char filename[100];
snprintf(filename, sizeof(filename), "%s_%d.txt", fileNamePrefix, file_num);
FILE *fp = NULL;
#endif

//Node Variables
#ifdef PARALLEL
int size;
int var_size = 5; //x, y, z, UDM_Value, PW Value
real ** data;
real *data_1D;
int pe;
#endif

//Find thread for current node
#ifdef !RP_HOST
thread = Lookup_Thread(domain,ZONE_ID);
#endif

//Open or create file on Host/Serial
#ifdef !RP_NODE
if ((fp = fopen(filename, "w"))==NULL)
{
    fprintf(fp, "*** t = %d ***\n",file_num);
}

else
{
}

#endif

//save the data
#ifdef RP_NODE
size=THREAD_N_ELEMENTS_INT(thread);
data = create_2d_array(size, var_size);
data_1D = (real *)malloc(size * var_size * sizeof(real));
//array = (real *)malloc(size * sizeof(real));

begin_c_loop_int(cell,thread)
{
    C_CENTROID(xc,cell,thread);
    //array[cell] = C_UDMI(cell, thread, 0);

```

```

    data[cell][0] = xc[0];
    data[cell][1] = xc[1];
    data[cell][2] = xc[2];
    data[cell][3] = C_UDMI(cell, thread, 0);
    data[cell][4] = C_UDMI(cell, thread, 1);
}
end_c_loop_int(cell,thread)

encode_2d_to_1d(data,size,var_size,data_1D);

/* Set pe to destination node */
/* If on node_0 send data to host */
/* Else send to node_0 because */
/* compute nodes connect to node_0 & node_0 to host */

pe = (I_AM_NODE_ZERO_P) ? node_host : node_zero;

PRF_CSEND_INT(pe, &size, 1, myid);
PRF_CSEND_REAL(pe, data_1D, size* var_size, myid);

//free(data_1D);/* free array on nodes after data sent */
//free_2d_array(data,size);

if (I_AM_NODE_ZERO_P)
compute_node_loop_not_zero (pe)
{
    PRF_CRECV_INT(pe, &size, 1, pe);
    real *data_1D = (real *)malloc(size * var_size * sizeof(real));
    PRF_CRECV_REAL(pe, data_1D, size* var_size, pe);

    PRF_CSEND_INT(node_host, &size, 1, myid);
    PRF_CSEND_REAL(node_host, data_1D, size* var_size, myid);
}
#endif

#if RP_HOST
compute_node_loop (pe)
{

    PRF_CRECV_INT(node_zero, &size, 1, node_zero);
    data_1D = (real *)malloc(size * var_size * sizeof(real));
    PRF_CRECV_REAL(node_zero, data_1D, size* var_size, node_zero);

    data = create_2d_array(size,var_size);
    decode_1d_to_2d(data_1D, size, var_size, data);

```

```

    for (int i=0; i<size; i++)
    {
        //fprintf(fp, "%g\n",array[i]);
        if (data[i][3] != 0)
        {
            fprintf(fp, "%g, %g, %g, %g, %g\n",data[i][0],data[i][1],data[i][2],data[i][3],data[i][4]);
        }
    }
}
#endif /* RP_HOST */

free(data_1D);
free_2d_array(data,size);

#if !RP_NODE /* SERIAL or HOST */
fclose(fp); /* Close the file that was only opened if on SERIAL or HOST */
moveFileToFolder(filename, folderName);
file_num++;
Message("Done\n");
#endif
}

DEFINE_EXECUTE_AT_END(fileSave)
{
    //every 10 flow iterations is 100dpm
    //every dpm iterations is 0.002s
    //thus, every 10 flow is 0.2s for flow
    const real dpm_secs_per_10_iterations = 0.2;
    const real flow_iterations = 10;
    static int iterations = 0;
    iterations++;

    int dpmTime = iterations * dpm_secs_per_10_iterations/flow_iterations;
    int dpmSaveFlag = dpmTime >= save_interval;

    if(dpmSaveFlag && auto_save)
    {
        iterations = 0;

        #if !RP_HOST
        Thread *thread;
        cell_t cell;
        real xc[ND_ND];

```

```

Domain *domain;      /* domain is declared as a variable */
domain = Get_Domain(1); /* returns fluid domain pointer */
//determine the number of useful cells for each node
#endif

//determine file name
#ifdef !RP_NODE
char filename[100];
snprintf(filename, sizeof(filename), "%s_%d.txt", fileNamePrefix, file_num);
FILE *fp = NULL;
#endif

//Node Variables
#ifdef PARALLEL
int size;
int var_size = 5; //x, y, z, UDM_Value, PW Value
real ** data;
real *data_1D;
int pe;
#endif

//Find thread for current node
#ifdef !RP_HOST
    thread = Lookup_Thread(domain,ZONE_ID);
#endif

//Open or create file on Host/Serial
#ifdef !RP_NODE
if ((fp = fopen(filename, "w"))==NULL)
{
    fprintf(fp, "*** t = %d ***\n",file_num);
}

else
{
}

#endif

//save the data
#ifdef RP_NODE
size=THREAD_N_ELEMENTS_INT(thread);
data = create_2d_array(size, var_size);
data_1D = (real *)malloc(size * var_size * sizeof(real));

```

```

//array = (real *)malloc(size * sizeof(real));

begin_c_loop_int(cell,thread)
{
    C_CENTROID(xc,cell,thread);
    //array[cell] = C_UDMI(cell, thread, 0);
    data[cell][0] = xc[0];
    data[cell][1] = xc[1];
    data[cell][2] = xc[2];
    data[cell][3] = C_UDMI(cell, thread, 0);
    data[cell][4] = C_UDMI(cell, thread, 1);
}
end_c_loop_int(cell,thread)

encode_2d_to_1d(data,size,var_size,data_1D);

/* Set pe to destination node */
/* If on node_0 send data to host */
/* Else send to node_0 because */
/* compute nodes connect to node_0 & node_0 to host */

pe = (I_AM_NODE_ZERO_P) ? node_host : node_zero;

PRF_CSEND_INT(pe, &size, 1, myid);
PRF_CSEND_REAL(pe, data_1D, size* var_size, myid);

//free(data_1D);/* free array on nodes after data sent */
//free_2d_array(data,size);

if (I_AM_NODE_ZERO_P)
compute_node_loop_not_zero (pe)
{
    PRF_CRECV_INT(pe, &size, 1, pe);
    real *data_1D = (real *)malloc(size * var_size * sizeof(real));
    PRF_CRECV_REAL(pe, data_1D, size* var_size, pe);

    PRF_CSEND_INT(node_host, &size, 1, myid);
    PRF_CSEND_REAL(node_host, data_1D, size* var_size, myid);
}
#endif

#if RP_HOST
compute_node_loop (pe)
{

```

```

PRF_CRECV_INT(node_zero, &size, 1, node_zero);
data_1D = (real *)malloc(size * var_size * sizeof(real));
PRF_CRECV_REAL(node_zero, data_1D, size* var_size, node_zero);

data = create_2d_array(size,var_size);
decode_1d_to_2d(data_1D, size, var_size, data);

for (int i=0; i<size; i++)
{
    //fprintf(fp, "%g\n",array[i]);
    if (data[i][3] != 0)
    {
        fprintf(fp, "%g, %g, %g, %g, %g\n",data[i][0],data[i][1],data[i][2],data[i][3],data[i][4]);
    }
}
}
#endif /* RP_HOST */

free(data_1D);
free_2d_array(data,size);

#if !RP_NODE /* SERIAL or HOST */
file_num++;
fclose(fp); /* Close the file that was only opened if on SERIAL or HOST */
moveFileToFolder(filename, folderName);
Message("Done\n");
#endif

}
}

```