2022

pymooCFD - A Multi-Objective Optimization Framework for CFD

George Martin Cunningham Love
University of Vermont

pymooCFD–A Multi-Objective Optimization Framework for CFD

A Thesis Presented

by

George Love

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Master of Science
Specializing in Mechanical Engineering

August, 2022

Defense Date: June 17, 2022
Thesis Examination Committee:

Yves Dubief, Ph.D., Advisor
Donna M. Rizzo, Ph.D., Chairperson
Jihong Ma, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of the Graduate College

Abstract

Modern computational resource have solidified the use of computer modeling as an integral part of the engineering design process. This is particularly impressive when it comes to high-dimensional models such as computational fluid dynamics (CFD) models. CFD models are now capable of producing results with a level of confidence that would previously have required physical experimentation. Simultaneously, the development of machine learning techniques and algorithms has increased exponentially in recent years. This acceleration is also due to the widespread availability of modern computational resources. Thus far, the cross-over between these fields has been mostly focused on computer models with low computational costs. However, this is slowly changing through the continued rapid development of both fields. The *pymooCFD* platform seeks to unite these fields of study by connecting a state-of-the-art library of optimization algorithms with industry leading CFD solvers. To begin with, this platform is important for testing the effectiveness of applying new optimization algorithms to CFD. Additionally, machine learning has been shown to help improve CFD models; this platform could serve to facilitate the development of better CFD models.

In this paper, the *pymooCFD* platform is applied to three different optimization problems. First, for validation purposes, the platform was used to conduct a well documented optimization problem, the reduction of drag around a circular cylinder through oscillating rotation around it's central axis. Second, the platform was applied to a Large Eddy Simulation (LES) to Reynolds-Average Navier-Stokes (RANS) model simplification. Lastly, the platform was applied to optimizing the direction, power and location of portable air purifiers in a room with six computer simulated persons (CSPs). The results show that the *pymooCFD* is a powerful tool for applying optimization algorithms to CFD. The validation of the platform was successful. A novel approach to CFD model simplification, called boundary condition calibration, is proposed. Finally, conclusions were drawn about optimal configuration of portable air purifiers within indoor spaces. These conclusions should serve to inform the experiments need to draw qualitative conclusion and create health advisories.

Code Repository: https://github.com/gmclove/pymooCFD

Dedicated to my mother for teaching me the value of good communication and for continuously helping me to develop better communication skills.

Acknowledgements

I would like to thank my family, friends and my advisor, Yves Dubief, for their support through the emotional roller coaster that has been this thesis. I would like to thank Professor Dubief in particular for providing me with the opportunity to pursue knowledge with a level of freedom I had never experienced before. I also appreciate the level of trust and generosity Yves has shown by allowing me to access his office as my own work space.

I would also like to acknowledge the YALES2 team, a team of researchers that develop the CFD solver using in most of this research. They are a welcoming and impressive community of researchers that has helped me through many a problem. To share an anecdote, at one of the YALES2 user meetings, I was the only non-French speaker present and the entire body of researchers switched to English to accommodate me.

This work would also not have been possible without the work of Dr. Julian Blank, the creator of *pymoo*, and the researchers at Michigan State University's Computational Optimization and Innovation (COIN) Laboratory that made the project possible. *Pymoo* is a python multi-objective optimization framework that is both generalizable and module, allowing users to apply state of the art optimization algorithms to any problem where evaluations can be automated. Julian Blank created *pymoo* in 2018 as part of his Ph.D. research. He focused on developing optimization algorithms for high-cost evaluation problems, such as CFD simulations. He will will soon be releasing these features for *pymoo*. I strongly believe these algorithms, and the *pymoo* interface in general, will become an integral part of uniting the gap between the highly specific optimization studies found in academic research and the

more practical industrial engineering design process. I was fortunate to be conducting my research in parallel with the development of *pymoo*, allowing me to utilize the program and hopefully make a meaningful contribution to the platform's future.

Finally, I would like to thank the Burlington, Vermont and University of Vermont communities. I have become deeply attached to this community and the many amazing people working to better the lives of their fellow community members. The Burlington and Vermont communities are a great examples of how a relatively small group of people can make a difference in the world.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Optimization algorithms attempt to find solutions, or operating conditions, that best satisfy a single or multiple objectives. For example, the profile of an airfoil (a wing) could be designed to minimize drag and maximize lift. A computational fluid dynamics (CFD) software is used to predict lift and drag for a given airfoil profile. While a computer-assisted design (CAD) software draws the profile which becomes the input to the CFD simulation. An optimization algorithm connects CFD and CAD software. It then systematically identifies profiles that best satisfy the lift and drag objectives. Such an approach, with the appropriate optimization algorithm, can be more efficient in identifying an optimal solution compared to running simulations of the vast range of possible profiles of the airfoil. Indeed, an optimization algorithm should, in theory, rapidly identify and focus resources on the set of profiles that are most likely to minimize drag and maximize lift while ignoring the rest of the possible profiles.

The goal of the algorithm is to minimize, maximize, or target a certain value for each objective by adjusting different parameters within the simulation. This set of parameters is adjusted within a given range for each iteration of the simulation. These

parameters can be the boundary conditions, the properties of the fluids, solids, the geometry of solids, or other modelling parameters. These parameters are typically real numbers, but could be binary, integer, or step function values. For example, a binary parameter could represent a valve being opened or closed.

In the last decade, the use of optimization algorithms in engineering has become commonplace as simulations become faster and more reliable. Optimization algorithms are being used as a design tool not only in research, but also in industry. In 2016, ANSYS, the industry-leading CFD software, released a platform for optimization on it's ANSYS Workbench platform. The release of this feature on a commercial software is an indication of this trend. [7]

1.1 Multi-Objective Optimization Algorithms

The engineering design process is continuously evolving with the emergence of new technologies. New technologies provide tools for improving this process, which in turn helps to create new, improved avenues for future technological development. This positive feedback loop has intensified the speed of scientific discovery and technological advancements throughout human history.

When developing new tools to improve the engineering design process, it is valuable to break down the process into smaller sets of interconnected problems, or goals. This can allow more specialized, precision tools to be developed and implemented in the design process. The continuous emergence of more specific fields of study in the scientific community can be viewed as an example of this development process. More

specific fields of scientific study lead to the development of more precise tools and knowledge.

In a design problem, once smaller design goals are well-defined, they can be grouped together based on the significance of their relationships to one another and their importance to the design as a whole. Sometimes these goals are competitive in nature. These "trade-off" relationships demand closer examination. By more closely examining the relationship between important, competing goals, insight can be gained into balancing their trade-offs in order to optimize the design. This approach to a design problem allows engineers to break down the design process into the framework of multi-objective optimization problems.

The ability to break down the engineering design process into multi-objective optimization problems is possible because the engineering design process almost always involves finding a balance between competing objectives. Minimizing cost is the most familiar example of an objective with trade-offs. Generally, optimization objectives can be grouped into cost objectives of a product and functional objectives of a product. Examining the trade-offs between these two groups of objectives, cost and benefit, is the subject of most multi-objective optimization studies.

In almost every design problem, a primary objective is to minimize cost. However, cost can be calculated in many ways. For certain systems, such as bridges, the cost is a function of material, construction and perhaps maintenance expenditures. Other mechanical systems, like HVAC systems, have an additional energy consumption cost throughout their lifetime, which is often referred to as the system's energy efficiency. Either way, in almost every design problem, one or more metrics for estimating cost are created to inform the design process.

Other objectives are usually specific to the function of the product. Metrics that measure the functionality of the product are always created during the design process. For example, a bridge must be capable of supporting a certain maximum amount of weight or an HVAC system should maintain a determined, minimal amount of oxygen in a room. Sometimes these functionality metrics have limits, minimum or maximum allowable values, but within that range there is room to explore potential "trade-offs" with other objectives.

These sets of functionality and cost metrics are typically created to inform the engineering design process, even when there is no intent of using a multi-objective optimization algorithm. Objective metrics are such an integral part of the design process, and yet sometimes, they are not thought of by engineers as having interconnected relationships that can be systematically explored. The multi-objective optimization framework offers a system to do just that. This is all possible because design problems are inherently optimization problems.

To summarize, most engineering systems are the result of compromises or trade-offs leading to the best solution based on multiple criteria that include technical constraints, such as the materials involved, the modes of operations, geometric limitations, etc., as well constraints outside of engineering like financial, aesthetics constraints. Multi-criteria optimization is a long-standing topic of research and applications in engineering [8].

In the last decade, the use of optimization algorithms in engineering has become commonplace as simulations become faster and more reliable. Optimization algorithms are being used as a design tool not only in research, but also in industry. In 2016, ANSYS, the industry-leading CFD software, released a platform for optimiza-

tion on it's ANSYS Workbench platform. The release of this feature on a commercial software is an indication of this trend. [7]

1.1.1 Pareto Optimum

Optimization algorithms work by mapping parameter, or search, space points to function, or objective, space points. In Figure 1.1, the search space is region S and the function space is region Z. This diagram represents a 3 parameter optimization problem with 2 objectives.

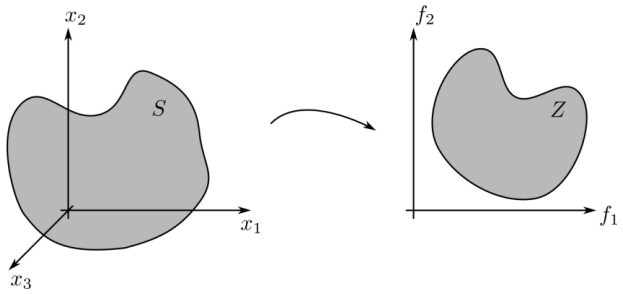


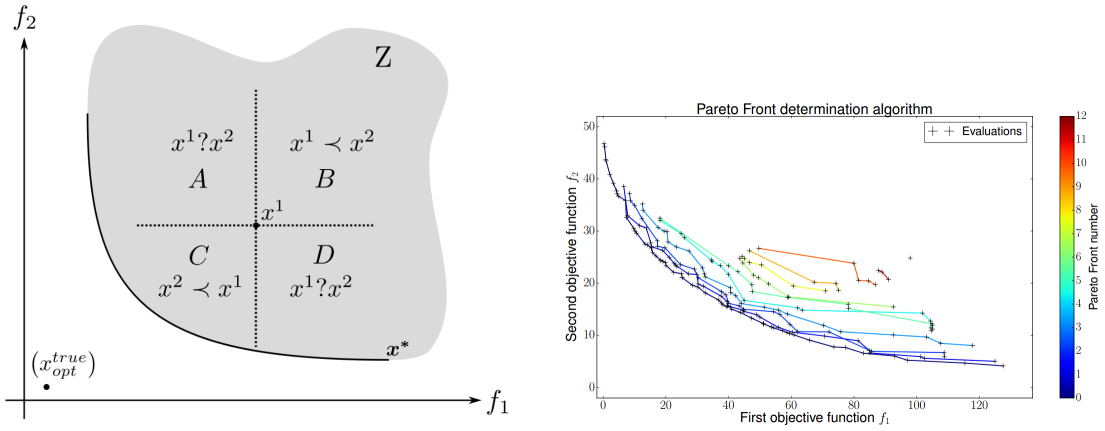
Figure 1.1: Search/Design/Parameter space, S, maps to objective space, Z. [1]

The mapping is determined by evaluating points in the parameter space to determine how they map to the objective space. The optimization algorithm, guided by the objectives, tries to find the most favorable regions of the parameter space with as few evaluations as possible. The ultimate goal of an optimization algorithm is to identify the most favorable points. The most favorable points can be connected to form regions. These favorable regions can be represented as *Pareto fronts* and *Pareto sets*. The Pareto sets and Pareto fronts exist in the parameter and objective spaces, respectively.

It is along the highest ranked Pareto front where the best trade-offs between

objectives are found. In other words, this is the space where the optimal set of solutions are found. Moving along the Pareto front allows the user to decide which objective(s) they want to favor. The Pareto front has one fewer dimension than the number of objectives. The easiest way to visualize this is with a two objective optimization problem, for which the Pareto front is a one dimensional line. Pareto front and Pareto set, unless otherwise specified, refers to the highest ranked Pareto front and Pareto set.

The images in Figure 1.2 below help to convey the concept of Pareto fronts and sets. Here there are two objective functions and the optimization algorithm's goal is to minimize both function. In Figure 1.2b the best trade-offs, or Pareto front, is the line colored black and ranked 0.



(a) Pareto front dominance. *(b) Two objective Pareto front visualization.*

Figure 1.2: Pareto front visualizations. [1]

The Pareto fronts are ranked using a method called non-dominated sorting. The highest ranked Pareto front contains only non-dominated solutions. Non-dominance of an objective point can be determined by drawing lines intersecting that point and parallel to the axes, as seen in Figure 1.9a. Now, regions of the objective space can

be defined using these lines as boundaries. One of these regions will be closer to your optimization goal. In the case of minimizing two objectives, this will be a 2D region to the lower left of the point in question. If any other points exist in this more favorable region, then these points "dominate" the point in question. A point that is not dominated by any other point is a non-dominated solution and is a part of the Pareto front. Solutions can be ranked by their dominance. Figure 1.2b uses rank 0 for the non-dominated set of solutions and counts up front there, ending with rank 12.

If an optimization algorithm has three objectives, the Pareto front is represented by a 2D surface. Though the 2D surface, also called a landscape, exists in a 3D objective space, it can be reduced to 2D in the sense that it has 2 directions of freedom to move along. This 2D surface provides a landscape of optimal solutions. Moving along this surface changes the balance of which objectives are favored. The Pareto front can have both local and global optimum. This is more likely to occur in a three objective problem, where the Pareto front is more complicated. A local optimum versus a global optimum along a 2D Pareto front is depicted in Figure 1.3 below.

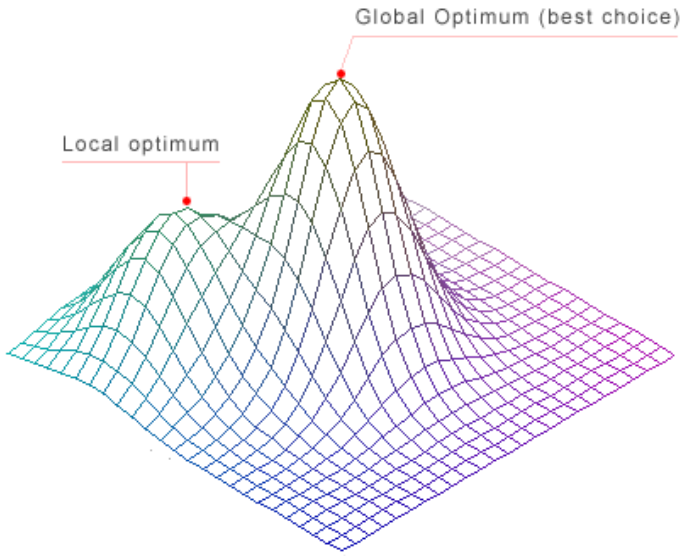


Figure 1.3: Three objective optimization - Local vs. Global Optima. [2]

The Pareto front is the most intuitive method of visualization for the results of an optimization algorithm, but there are many others. As one moves past three objectives to what are sometimes called "many-objective optimization" algorithms, visualization becomes more difficult and abstract.

1.1.2 Optimization Algorithm Types

Now that we have established the fundamentals of optimization algorithms, the different category of optimization algorithms can be discussed. Gradient descent optimization algorithms is one major categorization. The name gradient descent is used because it is common practice to manipulate your objective function(s) so that from the algorithm's perspective the goal is to minimize each function. In other words, the algorithm is searching for the global minimum. These algorithms operate by searching regions of the parameter space, mapping them to the objective space, and then look

for a downward slope in the objective space. This gradient search method, as it is sometimes called, has a notorious drawback; these algorithms can get stuck at a local minimum in non-convex Pareto fronts and fail to find the global minimum. Almost every gradient informed algorithm has a method to avoid this outcome. These methods involve evaluating more of the parameter space, which increases computational demand. Unless the space is convex, there is no guarantee gradient descent search algorithms won't get stuck on a local minimum.

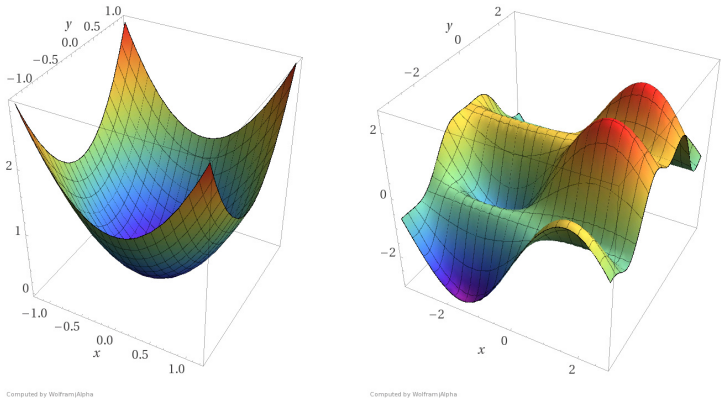


Figure 1.4: Convex (left) versus non-convex (right) objective space Pareto front. [3]

These algorithms are common place in most machine learning, and in particular neural network optimization. Neural network problems typically contain tens, if not hundreds, of parameters. Gradient descent algorithms are well suited for handling these large parameter spaces because they are able to focus their search only on regions of the parameter space that map to a downward slope in the objective space.

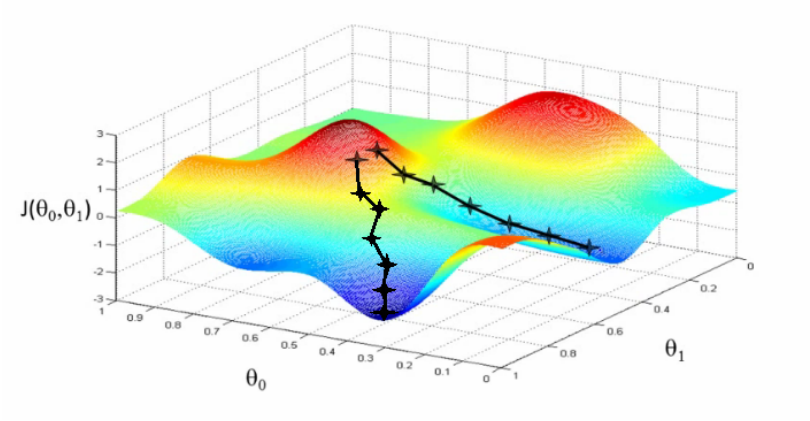


Figure 1.5: Gradient descent method with a single objective function, $J(\theta_0, \theta_1)$, and 2 parameters, θ_0 and θ_1. Here, the objective and parameter space are visualized using a single 3D graph. [4]

Figure 1.5 depicts an optimization problem with 1 objective, $J(\theta_0, \theta_1)$, and 2 parameters, θ_0 and θ_1. This figure shows how starting from different points close to each other would cause the algorithm to reach different optimum. However, the algorithm only has to evaluate points along that path to reach an optimum. This greatly reduces the number of evaluations needed to find the optimum, whether that be the local optimum in the case of non-convex Pareto fronts, or the global optimum in the case of a convex Pareto front.

Since CFD models are high-dimensional, complicated simulation predicting whether the Pareto front will be convex or non-convex can be extremely convoluted. Additionally, these algorithms are typically single objective, or, if they are multi-objective, they are not designed to converge around a well distributed Pareto front. For example, the Multiple Gradient Descent (MGD) algorithm converges to a single Pareto point. In recent years, gradient informed algorithms that identify well distributed Pareto fronts have been developed. There are now algorithms such as Pareto Front Stochastic Multiple Gradient (PF-SMG). This algorithm is designed to converge around a

well distributed Pareto front, but only for convex objective functions. [9]

Other algorithms, such as Pareto Multi-task Learning (PMTL) [10] and Exact Pareto Optimization (EPO) [11], are not guaranteed to converge around a uniform Pareto front additionally they require reference, or preference vectors. Gradient-informed algorithms are powerful in their ability to rapidly explore large parameter spaces. However, multi-objective gradient-informed algorithms that produce a well distributed Pareto front are a new development in the field and are not well tested. [12]

Another major category of optimization algorithms are genetic algorithms (GAs). This category of optimization algorithms can also be called evolutionary algorithms (EAs) or population-based algorithms, though the latter is a more encompassing category. Genetic algorithms are typically gradient-free and instead are designed using a framework provided by biology, the natural evolution of organisms. [12] Generally, GAs begin by generating a population of individuals with random genomes. An individual is defined by its genome. The genome of an individual is the set of variables, or parameters, that make the individual unique. These are the variables being adjusted in an attempt to find an optimal solution. In graphical terms, every individual is a point in the design space.

Every individual, or set of variables, is evaluated. For this study, each set of parameters is evaluated by a CFD simulation. The objectives are then extracted in CFD post-processing, which will be explained in detail in the sections below. Next, the genetic algorithm creates a new population of individuals to be evaluated. Each population of individuals created is called a generation. With each generation, the algorithm seeks to evaluate a population with individuals that are non-dominated. This allows the algorithm to converge around a well-defined Pareto front.

When working with high-cost evaluation problems, customizing the parallelization of evaluations based on the computation resource available is a significant factor in improving algorithm run time. Compared to gradient descent algorithms, genetic algorithms offer greater parallelization flexibility. When using a genetic algorithm, an entire population can be evaluated in parallel. The number of offspring, new individuals created each generation, can be adjusted based on computational resource available. Some gradient descent algorithm use batches of evaluations that could be run in parallel. However, these algorithms are not necessarily multi-object and are incapable of producing a well distributed non-convex Pareto front. Genetic algorithms are well suited for CFD simulations and the focus of this paper.

There are some key operations within genetic algorithms that can be adjusted to improve the algorithm's performance in certain situations. Sampling is an example of one of these processes, though it is not specific to genetic algorithms, as it is an operation required at the beginning of every optimization algorithm. The initial population must be selected, typically in a random fashion. There are different method of random sampling. The python multi-objective optimization package used in this study, pymoo, provides the option of using a completely random sampling, the Latin hypercube sampling or random permutation sampling. For more specifics on pymoo visit www.pymoo.org or refer to [13]. Biased sampling can also be used to speed up convergence if the user has a priori knowledge of the problem.

After the initial population is evaluated, by CFD simulations in our case, the next population to be evaluated must be selected. With genetic algorithms, this survival selection operation returns the individuals that will be the parents of the next generation. These selected parents are mated using the crossover operation,

where the genomes of parents are combined to create new individuals. Finally, a mutation operation is applied to genes in the new individuals' genomes. Often a Gaussian distribution around the gene's previous value is used to select the mutated gene value.

There are many genetic algorithm schemes, but they all use the power of evolution to seek an optimal solution. Implementing the age-old tools of sexual reproduction, random mutation, and natural selection.

1.2 Computational Fluid Dynamics

Since the 1980s, computational fluid dynamics (CFD) has grown to become a prevalent tool in the engineering design process. The revolutionary computing power of the 1980s brought about the first modern implementation and applications of CFD.

The democratization of CFD has been accelerated in recent years by (1) increased accessibility to computing power through cloud computing in particular, (2) computational resources that enable the simulation of practical engineering problems and (3) the pay-off from decades of investment in CFD modeling research. [14] In addition, the increased speed of more precise and accurate models has opened the door for real-time control system applications. [15]

One possible application of CFD informed real-time control systems that has been discussed among COVID researchers is essentially a Roomba® for air purification. This robot moves around a room detecting contamination, predicting it's spread through the air and moving throughout the room filtering out the contaminants. [16] Another similar application would be to use drones to back trace a gas contamination leak

to its source the same way male moths follow plumes of pheromones emitted by females [17]. The theory of transport of scalar in turbulent flows does not contain the type of spatio-temporal information to back-trace the origin of a leak based on boundary conditions. Such a task can be achieved at the scale of a chemical plant, for instance, using multiple CFD runs to explore the parameter space of leak location and local climate in combination of with machine learning. The objective would be to identify in real-time the most likely location of the leak, given the location where the leak plume has been identified and the meteorological conditions prior to the time of detection. Although this approach would require a supercomputer, it is within reach using current resources with the help of low-order models informed by prior high-fidelity simulations, for instance. [18]

It is reasonable to assume this trend towards CFD as an integral part of the engineering design process will continue. Computer power and access to this power has increased at an astonishing rate over the 50 years since the invention of the integrated circuit in the late 1950s and early 1960s. This can be seen in recent studies of Moore's Law. [19] [20]

The bigger picture is important to keep in mind, but it is also important to understand the current limitations of optimization algorithms applied to CFD. As suggested above, the main limitation is in regard to computational power. Optimization algorithms rely on running iterations of the simulation and learning from the results. This means that it must be practical to run often hundreds of simulations before meaningful results can be obtained.

Better modeling is one way to overcome these computational limitations. In modeling the precision and accuracy of a simulation, its ability to reproduce the behavior

of the real world, is called the model's fidelity. The higher the fidelity of a computational simulation, the more computational power is required. In fluid dynamics, for example, Some of the most computationally expensive phenomena to simulate is high-Reynolds number turbulence. The Reynolds number is a non-dimensional number that quantifies the energy of a flow. It is derived from a velocity scale of the flow U, the largest length scale, L, and the fluid viscosity, here we will use the kinematic viscosity, nu:

$$Re = \frac{UL}{\nu} \tag{1.1}$$

Turbulence is driven by many coherent scales, with length and time scales spanning a range that increased with the Reynolds number. The ratio of the smallest to the largest scale is estimated as

$$\frac{\eta}{L} \sim Re^{-3/4} \tag{1.2}$$

. For the turbulent boundary layer over the wing of a commercial jet aircraft at cruise speed, the largest scales are expected to be of the order of centimeters (boundary layer thickness), whereas the smallest scales are of the order of microns.

The highest fidelity CFD models that exist are called Direct Numerical Simulations (DNS). These models "directly" solve the Navier-Stokes equations, which are the mathematical expression of the conservation of mass and the conservation of momentum of Newtonian fluids. In other words, all the scales are resolved in time and space. This is a powerful too to study turbulence, albeit limited in Reynolds numbers even with today's supercomputers [21].

A closely related high fidelity model is called Large Eddy Simulation (LES). LES simulations only solve the full Navier-Stokes equations up until a certain length scale

of "eddies" is reached. The energetic contribution of the scales below this cutoff scale is modeled under the assumptions that such scales are universal enough that the theory of turbulence can predict their energy content [22]. LES can provide of around tenfold to hundredfold in certain cases and are particularly useful in applications where multiphase phenomena, mixing of chemical species or temperature, or vibrations are important. A perfect example is combustion, which is the main focus of the development of the high-fidelity code used in this work, YALES2 [23].

The lowest fidelity model is RANS, Reynolds average Navier Stokes, named after the transport equations it solves. The flow, species, phase, energy transport equations are averaged in time to derive a system of equations for the local averages, or resolved variables, of velocity, concentration, phase and energy. RANS equations are derived using the Reynolds decomposition, that states that any flow variable in a statistically steady state flow is the sum of its local mean and fluctuation, e.g. for the velocity:

$$u(x, y, z, t) = \bar{u}(x, y, z) + u'(x, y, z, t) \tag{1.3}$$

In the process of averaging, new terms arise in the equations which trigger a closure problem: these terms, typically products of fluctuations can only be resolved through a semi-empirical model based on the resolved flow variables, or new transport equations. The latter approach creates more new terms that must be closed. For the RANS simulations of momentum only, the simplest models add one equation to the momentum transport equation, whereas the most advanced may add 5 or more. [24] Whereas RANS models can accurately predict lift and drag on a body in steady state flow conditions, their ability to predict the turbulent dispersion of a contaminant or the turbulent transport of heat remains poor in complex flows. [25]

Currently, the only practical applications of optimization algorithms to CFD is through lower fidelity models. These models have a narrower range of applications in which they perform adequately by producing physically reliable results. However, these low fidelity models are the most persistent in industry because of their low computational cost. In addition, the most popular of these models such as, Reynolds average Navier-Stokes (RANS) and large eddy simulation (LES), have been around since the 1970s and the applications in which they perform adequately are well documented. These applications tend to be high Reynolds number flows.

1.3 Optimization Applied to Modeling

As discussed so far, applying optimization algorithms to models that are verified and validated to adjust sets of physical parameters is a powerful tool. However, with the CFD models and computational resources currently available, there exists limitations to this work. Only low fidelity models can be used in an optimization study. Luckily, optimization algorithms provide a tool for developing new CFD models which can achieve higher fidelity results with less computational power. This application of optimization algorithms is an example of the positive feedback loop between improving the engineering design process and the emergence of new technology.

Optimization algorithms along with other machine learning, such as artificial neural networks (ANN), provide a method for improving and developing models. Optimization algorithms can be used to study the effects of different combinations of modeling parameters in order to improve the model. This method of model development falls into the category of *data-driven modeling*.

Data-driven modeling can be approached in many ways. In [26], one of the most informative papers to this study, data-driven models are represented using the following equation.

$$\widetilde{M} = M(w; P(w); c(\theta); \delta(\theta, \eta); \epsilon_\theta) \tag{1.4}$$

where the variables are:

- M: model
- w: independent, averaged variables
- P: algebraic or differential operators
- c: set of target parameters
- θ: data
- δ: discrepancy
- η: features
- ϵ_θ: uncertainty

Data-driven modeling is driven by the discrepancy term, δ, which refers to the quantification of the difference between the current model prediction, $\widetilde{M}$, and the high fidelity data, θ. There are many approaches to formulating the δ function. Some δ functions are in terms of specific features of the model, η. Specifying features can be computationally beneficial when working with large sets of candidates.

The discrepancy term in the framework of a multi-objective optimization problem would be expressed as one or more "functional" objectives. Additional functional

objectives could include a quantification uncertainty, ϵ_θ, though including an uncertainty evaluation for every optimization algorithm evaluation is typically impractical. As discussed later in Section 1.4.1, **Verification & Validation**, there are less computationally intensive approaches to ensure that acceptable levels of uncertainty are maintained. Uncertainty is important to consider when creating your model functionality/fidelity objectives because there is an inherently competitive nature between a model's computational cost and its predictive uncertainty.

The "cost" objective of a model which competes with the model's functionality/-fidelity is minimizing computational power. If the same amount of computational power is dedicated to each evaluation, this can be measured simply as the time it takes to complete the simulation. If an adaptive distributed computing system is used, a slightly more complicated metric would be used that factors in the changing computational resources of each evaluation.

The results of the optimization study will map the relationship between the set of target parameters, c, and the two or more competing objectives. The optimization algorithm will search for the best trade-offs between cost and functionality, converging around the Pareto Front.

1.4 Methodology

The computation time of an optimization study with a genetic algorithm can be approximated using the following equation:

$$T_{opt\ study} \approx T_{ind\ sim} \frac{N_{pop}}{N_{parallel\ sims}} N_{gen} \tag{1.5}$$

where, $T_{opt\ study}$ is the optimization study total time, $T_{ind\ sim}$ is the computational time of individual simulation, N_{pop} is the number of individuals in a population, $N_{parallel\ sims}$ is the number of parallel simulations, and N_{gen} is the number of generations.

Ideally, the entire generation can be run in parallel, $N_{parallel\ sims} = N_{pop}$. This means optimization study total time can be approximated as:

$$T_{opt\ study} \approx T_{ind\ sim} N_{gen} \tag{1.6}$$

1.4.1 Verification and Validation

The verification and validation (V&V) is an essential step in any CFD applications. The interest to codify V&V grew in the 90s from researchers like Oberkampf [27] and is regaining renewed interest with recent progress on uncertainty quantification [28]. The verification step ensures that the equations of the physics are solved adequately. A specific goal is to demonstrate that the numerical methods used for time integration and computation of spatial derivatives or integrals have the expected theoretical discretization errors. This step is mostly a debugging step.

The validation step should show that the choice of numerical methods, models (including the form of governing equations – incompressible, low-Mach variable progress variable, compressible, plasma, etc.) is adequate for the purpose of the study. The validation of CFD typically consists of a comparison between the prediction of the code and experimental data, or through a code-to-code comparison. There is a level of subjectivity in the choice of flow for which the validation is carried out, as well as in the analysis of the comparison. Flows, even in the same category, may be vastly different with sometimes small changes of boundary conditions. A code may perform well in flows where one physical phenomenon is not dominant, and therefore passes validation against such flows, but fail in other flows that are dominated by that physical phenomenon. The choice of the validation is sometimes constrained by availability of benchmark, which may be just the former flows. In that case, the code is validated but fails in the latter flows and the user may not be able to identify/assess/measure the failure. The other source of subjectivity is the acceptable percentage of error that constitutes an acceptable error. When comparing with experiments, the CFD users must account for measurements' uncertainties, which are not always reported. There is a wide range of possible approaches which are dependent on how the study will be used to inform the product design. This section will address how the V&V of CFD models can be applied in the context of an optimization study.

Several possible V&V procedures will be discussed here. The V&V of a CFD optimization problem presents added challenges due to the variability of the models being evaluated. The variability of the models create added degrees of uncertainty that must be parsed out during the V&V procedure.

The first step is typically to create a test case as close to the center of the design

space as possible and decide on an appropriate V&V procedure for this case. If the optimization study is being applied to a simulation that already exists, a base case, then it is often already close to the center of the design space as users will likely be optimizing a model who design limitations exist above and below the current model.

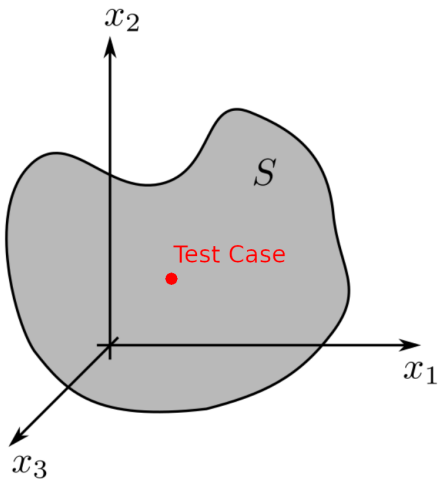


Figure 1.6: Search/Design/Parameter Space. The test case will be considered the case closes to the center of the design space. [1]

The most rigorous approach would be to use the same test case model V&V procedure on models with parameter sets across the entire design space. This is not a practical approach in many optimization studies due to the size of some design spaces. In addition, such a rigorous V&V procedure may not be necessary. As with any CFD models, the rigor of the V&V procedure is dependent on how the simulations will inform the product design process. In this section, I will present different possible approaches to V&V of a CFD optimization study. A wide range of V&V procedures and the code used to conduct them will be presented. The hope is that the user will then be able to decide on an appropriate set of V&V procedures which will provide the desired level of confident in their optimization study results.

Mesh Sensitivity Studies

Conducting a mesh sensitivity study, also called a computational grid convergence study, is an important part of the CFD verification process. A mesh sensitivity study involves shrinking, or refining, the mesh's element sizes until the results produced by the simulation converge. This indicates that the mesh is at a size that properly captures the fluid flow. The coarsest mesh with the fewest number of elements that still produces the converged results is chosen. This mesh is ideal because it produces results unaffected by the mesh's size, and it requires the least amount of computational power.

In some CFD optimization studies, a single mesh sensitivity study of the test case may be inadequate for verification of all possible CFD simulations within the design space. Any combination of parameter values within the given limitations could be evaluated using a CFD simulation. Therefore, it is important to consider which sets of parameters will produce fluid flow at the smallest scale. Capturing these small scale flow fields would require refining the mesh to give accurate results.

One solution to this problem is to use adaptive meshing, which is a mesh that changes size throughout the simulation to properly capture the fluid flow. In most adaptive mesh schemes a threshold metric is set and the mesh gets smaller when this threshold crossed. Often, the threshold is a measure of a value's gradient between cells. When this gradient between cells crosses the set threshold, these "marked" cells are divided or fused together, making the mesh coarser or more refined. However, selecting a threshold that works across the design space without failing to capture flow characteristics, or without producing a mesh size that is too computationally demanding, may prove more trouble than it's worth.

The first step is to see if the parameter space chosen for the optimization study will present mesh sensitivity problems. NSGA-II, and most genetic algorithms, use the first generation to evaluate sets of design parameters distributed across the entire design space. This distributed sampling of the design space provides a set of simulations with which to conduct a mesh sensitivity studies on. The results of these mesh studies provides verification of mesh convergence across the entire design space.

In *pymooCFD/core/cfdCase.py* there is an attribute named *mesh_study*. This method provides a framework for automating the procedure described above. The code works by scaling the case's mesh using an array of scale factors provided by the user. The *mesh_study* object is run by calling it's *run* method, *cfdCase.mesh_study.run()*. This *run* method finishes by calling its own *plot* method, *cfdCase.mesh_study.plot()* which plots the values of the optimization study's objectives as a function of the number of elements in the mesh. These plots can then be evaluated by the user to decide what level of mesh refinement will provide the appropriate convergence of the objectives' values. Then the coarsest mesh that provides convergence for any possible set of parameters can be selected and used for all the simulations. This solution is best suited for simulations that are fast and simple. A fast simulation is preferred because using a mesh more refined than necessary in some areas of the parameter space will increase computation time. A simple model allows the user to better select what set of parameters will require the smallest mesh without missing a set of parameters that could create an unexpected small scale fluid phenomenon.

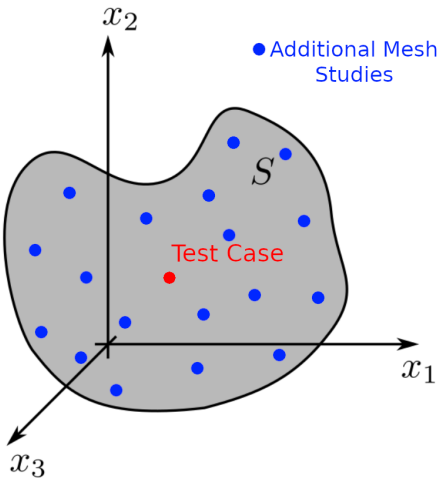


Figure 1.7: Search/Design/Parameter Space. Optimization study mesh convergence verification methodology. [1]

Using the *mesh_study* attribute of each *cfdCase* to conduct mesh studies on the entire first generation may be a more rigorous approach than necessary. Less computationally demanding approaches will be adequate in most cases.

Another, less rigorous, approach would be to conduct mesh sensitivity studies on CFD models with sets of parameters that are along the limits of the design space. These mesh sensitivity studies along the limits of the design space, in addition to a mesh study of a test case (at the center of the design space), may provide an appropriate level of confidence in the verification of CFD models throughout the entire design space.

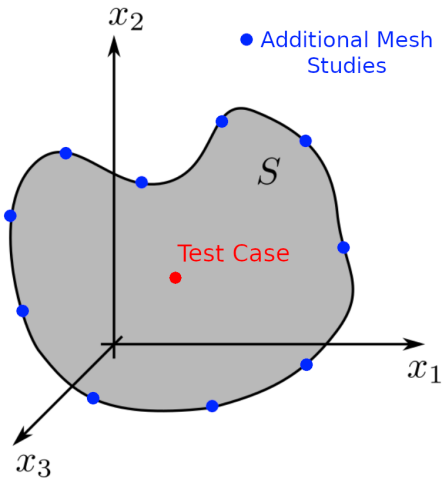


Figure 1.8: Search/Design/Parameter Space. Optimization study mesh convergence verification methodology. [1]

These can be conducted using the *OptRun* class method *run_bnd_cases*, found in the *pymooCFD.core.optRun* module. Calling this method during optimization study pre-processing can be a critical step in evaluating the appropriate parameter limitations. If the keyword argument *doMeshStudy=True* is passed to this method, it will conduct a mesh study on each of the boundary cases. Other arguments include the number of points along each limitation's edge, *n_pts=2* is the default. This default value will only generate corner cases.

Pareto Front

There are a range of V&V procedures that establish varying levels of confidence in the models generated by the optimization study. The simplest, but least confident method, is to conduct the V&V process on a test case in the center of the design space and assume that the variability in the model creates acceptable levels of uncertainty in the V&V process. Another option is to create models that exist at the corner

of the design space and conduct V&V on each of these cases. Keeping in mind this will produce $2^{N_{var}}$ cases that must be verified and validated, where N_{var} is the number of parameters. Even more confidence can be established by verifying and validating additional models along the boundaries of the design space. The most rigorous approach to the V&V procedure of a CFD optimization problem is to verify and validate models created using a distributed sampling of parameter sets from across the entire design space. In this way, you can achieve the same level of V&V across the entire design regime.

Another piece of the V&V process of a CFD optimization study is the V&V of the Pareto set/front. This is the set of solutions that will be used to inform design decisions. Therefore, it is extremely important that it is properly verified and validated. In some cases where the user has a high degree of confidence in their base case model, or as higher-fidelity simulations become viable options for optimization problems, this may be the only V&V needed. If solutions along the Pareto front satisfy the V&V procedure, then one could assume that the models evaluated by the optimization algorithm which led to the realization of a Pareto front also satisfy the V&V producer. The assumption is that the evaluations proceeding the Pareto front evaluations appropriately informed the realization of the Pareto front.

The use of multi-objective optimization algorithms with CFD simulations requires at minimum some verification and validation of the base case before proceeding with an optimization study. This should include a mesh convergence/sensitivity study for verification, as well as an evaluation of the model's fidelity for validation. As mentioned previously, the iterative nature of these algorithms requires the use of low-fidelity simulations. It is important to consider the possibility that flow fields

not captured by the model create a degree of uncertainty in the objective metrics. Experimental or high-fidelity simulation results should be compared to the low-fidelity simulation results to validate the results. This should indicate if there is anything that is not captured by this model that is significant to the results of the simulation.

1.4.2 Design Space Selection

Selecting and then fully verifying and validating an optimization study design space is the most tedious and subjective part of the CFD optimization study procedure. Evaluation of the first generation, which contains CFD cases distributed across the entire design space, or even the first few generations, often becomes an iterative process. During this iterative process, potential limitations of the computational model to produce verified and validated results are revealed. Additionally, the effects of the design parameters on the objectives are explored. The effects of the limitations placed on the design space must be examined critically. In many cases, there will be physical limitations or other product design factors that will limit the design space. If these external limitations encompass the entire design space, then this gives your design space selection definitive boundaries to begin with. However, in some cases, the design space does not have definitive boundaries with respect to the upper or lower boundary of one or more parameter. Putting a limitation on these "open-ended" parts of the design space can be challenging when working with problems complicated enough to warrant an optimization study. One solution is to guess and check by running a few generations and seeing if the optimization algorithm is restricted by the arbitrary limitation you have established. If an arbitrary limitation on the design space is restricting the objective space, then it should be adjusted as long as the

parameter value means the model is still physically and computationally valid.

Eventually, if the optimization algorithm is trying to minimize all objectives, as it is with all *pymoo* algorithms, then there should no longer be a sharp corner in the lower left of the objective space. A beveled edge to the Pareto front indicates that the design space is capable of exploring the limits of objective space. Here it can find a fully realized Pareto front which gives the best trade-offs between objectives. If a sharp corner remains, then the necessity for a full multi-objective optimization study may need to be re-evaluated.

The relationship between parameter and objectives, as well as the competitive nature between objectives, may be more straight forward than originally thought. A Pareto front with a gradient of 0 and infinity exclusively indicates that the limit of the objective's competitiveness, the true Pareto front, is not being evaluated.

As the lower or upper limitation of a parameter is adjusted, it may be prudent to restrict the opposing limit of that same parameter. Exploring a design space that is only as large as you need it to be creates fewer problems when constructing a computational scheme that is valid across the entire design space.

To summarize, since creating a computational scheme that is valid over a range of parameters is challenging and has inherent limitations, then the selection of the design space is fundamentally tied to these limitations. Therefore, unless a computational scheme is extremely rigorous or adaptive, then selection of the design space typically involves adjusting the space's limitations, running a generation or two, then checking if the computational scheme is still producing valid results. Adjustments to the computational scheme can then be made if needed. At first, to save time, these adjustments could be made using a graphic user interface instead of automating using

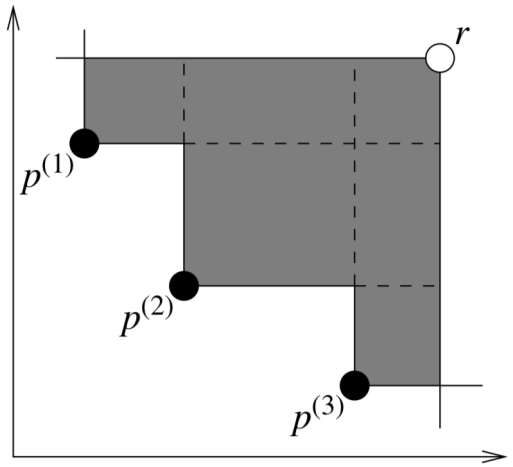
code. Once the scope of the adjustment to the computational scheme is understood, and the impact of this new parameter limit on the objective space is understood, then coding of the new computational scheme should commence. After this, the first few generations can be run once more and the objective space can be re-examined to establish whether the Pareto front realization is still being limited by an arbitrary design limitation.

1.4.3 Analysis

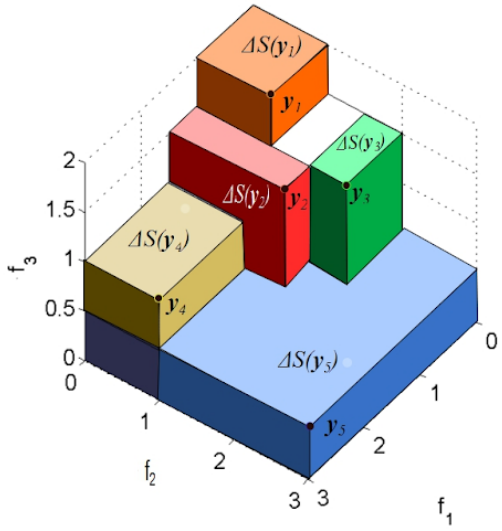
Analysis of optimization studies is performed using a Jupyter Notebook file, *optPost-Proc.ipynb*, located in the top level of the optimization study folder. By placing this notebook at the top level the notebook file can access all the data generated by the optimization study. The notebook should be launched from your *pymooCFD* Anaconda environment. It is important to note if you are using a terminal to launch the notebook. You must also be in a file location that allows access to your *anaconda3* folder, a.k.a. your "home" directory, where your environment's python packages are stored. The notebook begins by loading the checkpoint files. Optimization algorithm checkpoints are generated twice every generation, once after the optimization algorithm selects the next population and once upon completion of the generation. This algorithm object contains the attribute history, *optRun.algorithm.history* or *optRun.algorithm.result().history*, which is a list of deep copies of the object itself saved every generation. If the checkpoint is from the middle of a generation and has no final objective population, the python notebook sets the algorithm equal to the last completed generation, *algorithm = algorithm.history[-1]*. From here, the analysis of the optimization study becomes increasingly subjective.

However, the code found in *optPostProc.ipynb* can provide helpful tools for any optimization study analysis. Any analysis should begin with an analysis of the first generation to ensure the design space has been properly selected and ensure the preliminary results warrant further investigation. The first generation, which is typically a distribution across the entire design space, is used to graph every parameter and objective against each other. These plots show the relationship of each parameter to each objective while, other parameters are also changing. This means any strong correlation indicated by the plots are independent of the other parameters'variability within the design space.

Next in the notebook file are the optimization algorithm convergence statistics, which inform the user how close to convergence the Pareto front is. Unless the optimization problem can be solved using mathematical analytics, the fully converged Pareto front is unknown. Optimization algorithm test problem often have a known Pareto front, while most real world problem do not. Instead, for real world problems, the convergence has to measured based on the progress being made by the optimization algorithm. A metric called hypervolume is used, which is a measurement of the space between the Pareto front points and a reference point. The dimensions of this hypervolume space are equivalent to the dimensions of the objective space. Figure 1.9, below, has a visualization further explaining hypervolume.



(a) Objective space showing 2-D hypervolume measurement. Each axis represents an objective metric, r is the reference point and p are the non-dominated Pareto front points. [13]



(b) Objective space showing 3-D hypervolume measurement. f_1, f_2, and f_3 are objective metrics, the reference point is coordinate $(0, 0, 0)$, and y_1 through y_5 represent non-dominated Pareto front points. The ΔS volumes are summed to get the final hypervolume measurement. [29]

Figure 1.9: Hypervolume visualizations. [1]

Finally, beginning with scatter plots, various visualization options can be found in the remainder of the Jupyter Notebook. The visualization tools provided by *pymoo* are wrappers around matplotlib, a popular visualization python package used by engineers. The wrapper provides convenient default keywords arguments and other features which make visualization of optimization studies more convenient while still allowing access to all the keywords options available in matplotlib. The *pymoo* Scatter class changes the type of scatter plots generated by the *pymoo* depending upon the dimensionality of the objective or design space being plotted. In spaces with more than 3 dimensions, pairwise scatter plots are used. The other visualization methods, such as parallel coordinate plots, heatmaps, petal diagrams, etc. are mostly suitable

for optimization studies with high dimensional spaces that must be analyzed. It is helpful to refer directly to *pymoo.org/visualization* for the latest updates on the visualization and other analysis tools offered.

1.4.4 Model Simplification Through Optimization

Computational model simplification has two main objectives: maximizing fidelity while minimizing computational cost. The best trade-offs between these objectives can be found by adjusting certain modeling parameters. Ideally, these modeling parameters should be selected by an expert in fluid dynamics and preferably by an expert in specifically computational fluid dynamics. However, this study will explore a method to modeling simplification through optimization that may allow engineers without expert knowledge in the specifics of a computational fluid dynamic model to achieve model simplification. In other words, this study will propose a methodology where someone with a basic knowledge of fluid dynamics could achieve model simplification. What remains to be explored is the generalizability of such models. Model generalizability should be considered the third major object in model simplification in future studies.

1.5 Platform Development

Computer-aided design (CAD) has become an integral part of the engineering design process. Geometric computer design tools, sometimes called "traditional CAD", are an industry standard. Computational analysis of these geometric designs, often through numerical analysis such as CFD, are also becoming increasingly commonplace. This has lead to the development of tools that analyze data from many different numerical analyses to inform engineering decisions. These types of software have been dubbed by some as “computer-aided engineering”. Many CAD programs have embedded tools for engineering analysis such as CFD and solid mechanics simulations. Company's are motivated to create a vertical integration of software in order to offer a one-license-fits-all solution where institutions and individuals are able to purchase then learn to use a set of software created by a single company. There are many ways this solution is more convenient than using software from a host of different companies. However, one often trades convenience for flexibility. In research, flexibility is essential to solving and exploring unforeseen problems. Cost is another barrier that looms over researchers. To achieve flexibility at a low cost a lot of open-source software was implemented for this project.

There are some distinct disadvantages with the current ANSYS optimization tool. The software is proprietary and therefore closed source. This makes it's use as a research tool extremely limited as there are very few adjustments to the algorithms offered. ANSYS markets these as tools used for the early design process which is appropriate considering their limitations. [7]

The solution is to create an open source application that allows for the adjustment

of all the optimization algorithm's parameters. The challenge is to create this tool in a manner that offers a comprehensive user interface that would allow an engineer with limited coding experience to operate with ease. Luckily there are many open source Computational Fluid Dynamic solvers, optimization algorithm libraries, computer aided design tools and fluid simulation post-processors available. The last piece of the puzzle is developing an interface between these open source softwares with the option to use proprietary software that many pre-existing models are build with.

In the interest of keeping the software more comprehensive and user friendly, python API tools were chosen whenever possible. A multi-objective optimization algorithm python package called *pymoo* is the core of the program. *Pymoo* was created by Julian Blank of Michigan State University Computational Optimization and Innovation Laboratory (COIN). Blank is a PhD graduate of Michigan State University and has a B.Sc. in Business Information Systems and a M.Sc. in Computer Science. *Pymoo* offers a comprehensive framework for setting up optimization problems for thirteen different optimization algorithms. In addition, Blank and his team are committed to expanding *pymoo* as new algorithms are developed. Blank and the COIN team are also specifically involved in researching optimization algorithms that are best suited for computationally expensive evaluations. Blank and another member of the COIN lab, Deb, published several paper on this subject recently [30] [31]. Most recently, [32] was published as the culmination of Blank's PhD program. Unfortunately, there was not time to fully integrate this extension to the *pymoo* interface into *pymooCFD*.

The two CFD solvers that will be used are ANSYS Fluent and YALES2. YALES2 was developed at CORIA, a French combustion laboratory located in Rouen, Nor-

mandie Université. The motivation for YALES2's development was solving two-phase flows, specifically atomization and combustion on large and complex meshes. However, since it's original conception the software has been expanded to include many high fidelity fluid solvers. YALES2 was built around Message Passing Interface (MPI) a protocol used to communicate across computer nodes making it well suited for running simulations on high performance computing clusters (HPCs). For more information visit www.coria-cfd.fr.

The other fluid solver used is Fluent. This is an ANSYS product and likely the most widely used CFD software in the world. [33]

The mesher that will be used is called GMSH and was developed by Christophe Geuzaine and Jean-Francois Remacle starting in 1997. OpenCascade is an open-source CAD python command line program that is baked into GMSH and used for constructing the geometry before meshing. More information can be found at https://gmsh.info/.

In this study the focus will be on multi-objective, genetic optimization algorithms. Since CFD simulations are a computationally expensive evaluation, genetic algorithms are chosen because of their parallel evaluation nature. Genetic algorithms evaluate an entire generation, or population of individuals, at the same time. These results then inform the next generation of evaluations. Multi-objective optimization algorithms are chosen because most engineering design problems that would benefit from an optimization study having multiple competing objectives.

The specific optimization algorithm that is used in this study is the Non-Dominate Sorting Genetic Algorithm II. [34] Non-dominated Sorting Genetic Algorithm II (NSGA-II) is one of the most prominent multi-objective optimization algorithms used

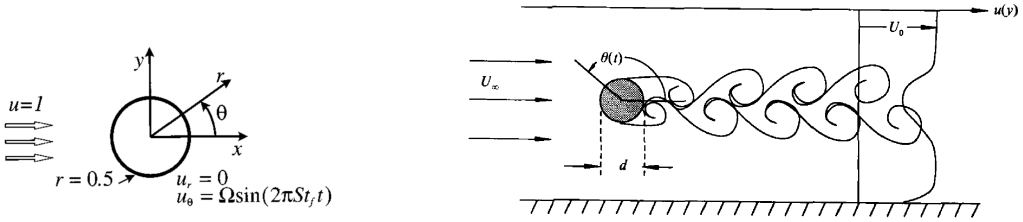
by engineers. It was proposed by Deb et al. [34] and has been widely used in CFD optimization studies since its creation in 2000. This algorithm will be the focus of this thesis because its widespread use in other CFD studies can help to provide a baseline for comparison. An example of one such study is [35].

Chapter 2

Cases

2.1 Oscillating Cylinder

The *pymooCFD* platform was verified using a CFD optimization study commonly found in literature. There are many studies surrounding the optimization of flow control techniques to reduce the vortex shedding around a cylinder in cross flow. These types of studies have some potential practical applications, such as increasing the longevity of cylindrical building materials that are subject to vortex shedding. One well documented flow control technique is the oscillatory rotation of a cylinder around its central axis.



(a) Diagram of oscillatory rotation of circular cylinder around central axis. [5]

(b) Diagram of entire domain. Figure 1 from [36].

Figure 2.1: Validating Strouhal number.

The flow control theory behind this technique states that there is something called the lock-on region, where the frequency of vortex shedding behind a stationary cylinder is about equal to the forcing frequency. The frequency of vortex shedding, f, can be described using the dimensionless Strouhal number.

$$St = \frac{fL}{U} = \frac{fD_{cyl}}{U_\infty}$$

In this case, the characteristic length scale is the cylinder diameter, $D_{cyl} = 1m$ and the velocity is the bulk velocity, $U_\infty = 1\frac{m}{s}$. The lock-on region can be described symbolically as:

$$St_n \approx St_f$$

Where St_n is the Strouhal number of a stationary cylinder and St_f is the Strouhal number of the forcing frequency.

2.1.1 Previous Work

For an object in a cross flow, as the Reynolds number increases, the drag on the object is increasingly contributed to by a phenomenon known as vortex shedding.

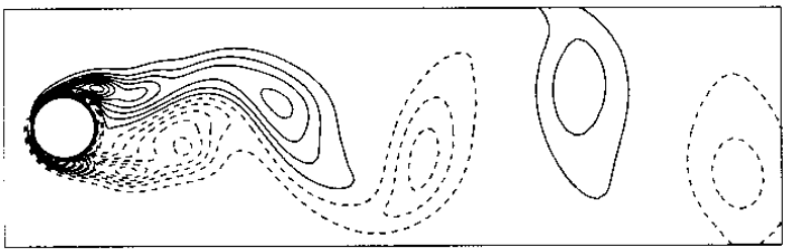


Figure 2.2: Vortex shedding around cylinder in cross flow. [5]

There have been many proposed "flow control" mechanisms for reducing the drag caused by vortex shedding. One of the most thoroughly studied of these mechanisms is the oscillation of a cylinder around it's central axis in cross flow. This oscillation, when applied at the correct amplitude and frequency, can cause a reduction in vortex shedding and therefore a reduction in the drag force. The relationship between Reynolds number, the amplitude and frequency of the oscillation, and drag has been explored in studies dating back to 1978 and likely even earlier. [37] [38] [36] [39]

Many studies of oscillating cylinders to reduce drag, especially early on, are approaches with the goal of exploring the fundamentals of the flow phenomenon that cause the reduction in drag. These studies have contributed to an optimal control theory. This theory, that has been confirmed by experiments and simulations, states that there exist synchronized, or phase-locked, regions where the vortex shedding can be greatly reduced. This is because as the fluid passes over the surface of the cylinder, a momentum gradient forms, creating a shear layer and eventually vortex shedding. The movement of the wall under the fluid disturbs the formation of the shear layer, which would normally propagate into large vortices. However, there exist only certain combinations of the forcing frequencies and amplitudes which create the desired drag reduction. This has been observed by both [36] and [40]. [40] also contends that there

exists other harmonics that are odd multiples of the fundamental frequency. Additionally, [40] states that as the forcing frequency is increased, the amplitude must also be increases for the flow to "feel" the forcing. While the latter is supported strongly by the present study, there is little evidence found to support the former claim.

The energy cost associated with this forcing will also be qualified in this study. There are relatively few studies on flow control optimization of a cylinder in cross flow that use an energy cost metric. Two studies that do use an energy metric are [41] and [42]. Neither study defines the material and geometric properties necessary to calculate the full energy cost of forcing. Instead, they quantify the forcing cost using the resistive force of the fluid on the surface of the cylinder, also called skin friction. The fact that none of the studies found define the specific material and geometric properties needed to get a full energy cost analysis indicates that there are very few practical applications of this flow control method.

Low Reynolds number flows were chosen for two reasons. One: after a Reynolds number of approximately Re=300 the three-dimensional effects of the wake become significant. [43] A 3-D simulation would constitute a significant increase in computational cost. Two: according to previous studies, the trade-off between a reduction in drag and the resistive torque of the fluid is more significant at lower Reynolds numbers. At high Reynolds numbers, the use of forcing to reduce the drag on the cylinder is more easily justified. [41]

The thorough documentation of physical and numerical experiments on oscillating cylinder in cross flows makes it an ideal case for verifying the *pymooCFD* platform. The following sections will describe more novel applications of *pymooCFD*.

2.1.2 Coefficient of Drag

The coefficient of drag is a dimensionless number used to quantity the force of drag on a particular surface area.

$$C_D = \frac{2F_D}{\rho U_\infty^2 A} \tag{2.1}$$

Where U_∞ is the bulk flow velocity, ρ is the fluid's density, A is the cross-sectional area perpendicular to the bulk flow and F_D is the force of drag on the cylinder.

The force of drag, F_D, on the cylinder is calculated by accounting for the surface forces on the object, which can be broken down into two contributing components. Let's begin with the simpler of the two to calculate, the contribution of pressure force, $f_{p,i}$. The subscript i is the Einstein/index notation, representing an arbitrary number of dimensions.

$$f_{p,i} = \frac{1}{\rho} \int -p dA_i \tag{2.2}$$

Here the pressure, p, is divided by density, ρ, because the model is incompressible. Therefore, it is simpler to use force per unit volume, or force density; this is indicated by the lower case f.

The second contribution to the surface forces are the frictional forces, f_f. In Equation 2.3 below τ_{ij} represents the viscous shear stress tensor, μ represents the dynamic viscosity and x_j represents the length of the surface perpendicular to the fluid's velocity.

$$f_{f,i} = \frac{1}{\rho} \int \tau_{ij} dA_j = \int \nu \frac{\partial u_i}{\partial x_j} dA_j \tag{2.3}$$

where ν is the kinematic viscosity, $\nu = \mu/\rho$. This yields the same units of force per unit volume that are used above. These frictional forces can also be called the viscous

shear forces. The net surface forces on the cylinder can now be calculated.

$$f_{surf,i} = f_{f,i} - f_{p,i}$$

In this case, the force of drag is the x-component, $i = 1$, of the surface forces. A time average is taken to get the mean force.

$$\overline{f}_D = \overline{f}_{surf,1} = \frac{1}{t_f - t_i} \sum_{t=t_i}^{t_f} f_{surf,1}$$

Where $t_i = 100$ seconds and $t_f = 200$ seconds. This time period was used because a maximum 100 seconds is needed to "wash out" initial conditions and achieve quasi-steady state flow. Then a 100-second time period is needed to achieve a well converged time averaged flow.

$$\overline{f}_D = \frac{1}{200 - 100} \sum_{t=100}^{200} f_{surf,1} \tag{2.4}$$

Finally, Equation 2.4 above can be plugged back into the coefficient of drag equation, Equation 2.1.

$$\overline{C}_D = \frac{2\overline{f}_D}{U_\infty^2 D_{cyl}}$$

Add in known values across all simulations. $D_{cyl} = 1[m]$ and $U_\infty = 1\left[\frac{m}{s}\right]$.

$$\overline{C}_D = 2\overline{f}_D$$

This can then be used to get a percent reduction in the coefficient of drag using the

known coefficient of drag for a stationary cylinder, $\overline{C}_{D,stat}$.

$$\overline{C}_D\%_{reduction} = \frac{\overline{C}_{D,opt} - \overline{C}_{D,stat}}{\overline{C}_{D,stat}} * 100 \tag{2.5}$$

2.1.3 Energy Consumption Metric

In addition to repeating a single objective optimization study similar to those conducted by most of the previous literature, a new multi-objective optimization study is conducted by adding a second objective to the optimization problem. This second objective is the energy consumption, or energy cost, metric. [44] [6]

The energy consumption metric will be quantified using the resistive torque applied by the fluid on the surface of the cylinder. It is very important to note that the inertia force is not accounted for. The net torque equation can be written as follows:

$$\Sigma\tau = I\alpha = \tau_{applied} - \tau_{fluid-resistance} \longrightarrow \tau_{applied} = I\alpha + \tau_{fluid-resistance}$$

The moment of inertia is application specific. With the goal of keeping this study as general as possible, the contribution of the inertial forces are not accounted for. The power metric used in [42] and [41] also do not account for inertia in their power metrics. These two studies of power metrics use an indirect calculation method of calculating torque applied by the fluid on the cylinder's surface. The study conducted here uses the direct measurement of resistive torque based on the stress tensors at each node on the cylinder surface. This measurement technique makes fewer assumptions and therefore should be more accurate.

2.1.4 Verification & Validation

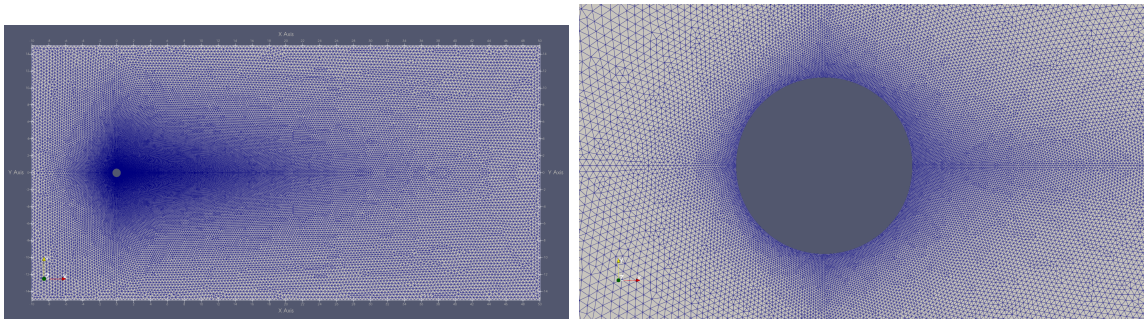


Figure 2.3: Oscillating cylinder mesh at mesh size factor of 1.0

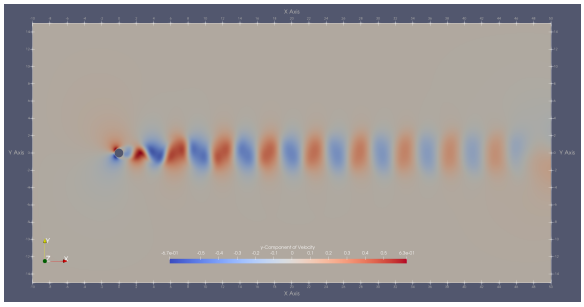


Figure 2.4: Vortex shedding in the no oscillation case at $t = 200$ seconds.

The first mesh study conducted was the case of a circular cylinder in cross flow with no oscillation. The lack of variability made this the easiest of the cases to validate with data from previous work.

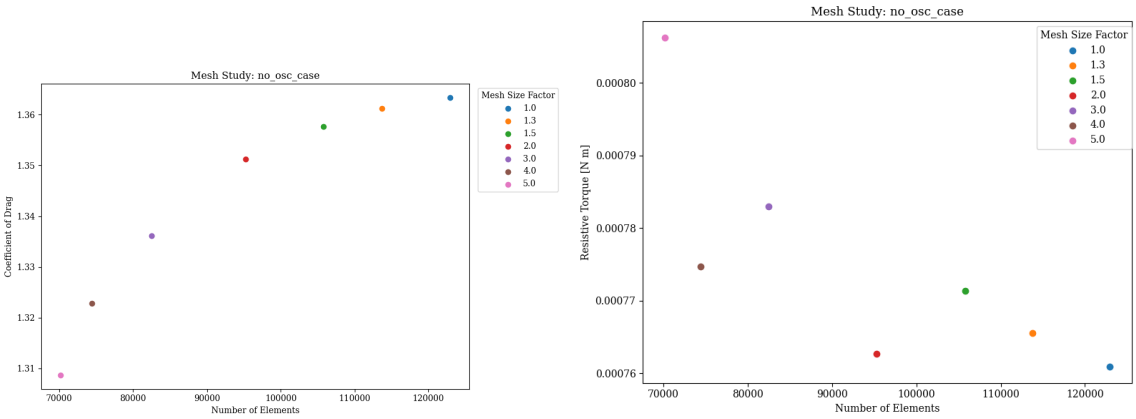


Figure 2.5: No oscillation case mesh study, Re=100.

Both the coefficient of drag and the resistive torque metrics show good mesh convergence at mesh size factor 1.0.

The coefficient of drag calculated by the simulation with a mesh size factor of 1.0 is tabulated in Table 2.1, $\overline{C}_D = 1.3633$. In the column next to this value are the coefficients of drag from previous work.

Re	C_{Drag}		Citation
	Present Work	Previous Work	
100	1.3633	1.16	[45]
		1.33	[46]
		1.431	[47]
		1.3528	[40]
		1.3500	[48]

Table 2.1: Tabulated coefficient of drag values for circular cylinder in cross flow at Re=100.

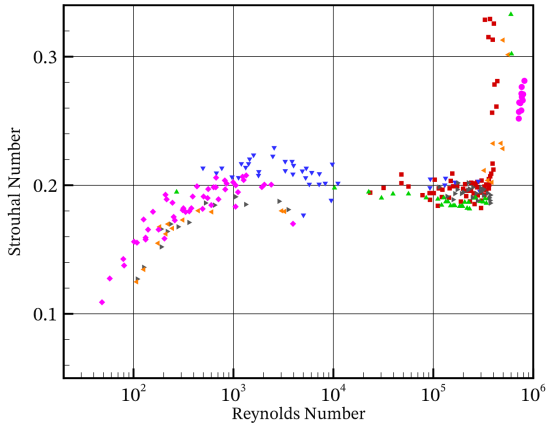
The coefficient of drag was the metric used to validate the model of a circular

cylinder in cross flow at a Reynolds number of 100. This process was less straightforward than expected. There are innumerable numerical studies of this scenario. This factor, along with the stochastic nature of the solution, are bound to lead to some discrepancy in results. However, there seems to be many examples of poor verification of the model which lead to larger than normal discrepancies. The study by [45] is included in Table 2.1 as an example of one such study. The mesh size seen in this study is far from being a uniform size at the boundary of the cylinder. In addition, there seems to have been no mesh sensitivity study. Care should be taken when validating this problem. The vast amount of previous literature on the subject is mostly beneficial, but can also lead to difficulty establishing which studies are properly verified and validated.

The Strouhal number was also found to be $St = 0.150$ for the no oscillation case simulated in this work. This is in agreement with the values found in previous studies.

Circular cylinder	
Author	St
Calhoun [16]	0.175
Silva et al. [18]	0.16
Xu and Wang [20]	0.171
Wang et al. [21]	0.17
Present study	0.152

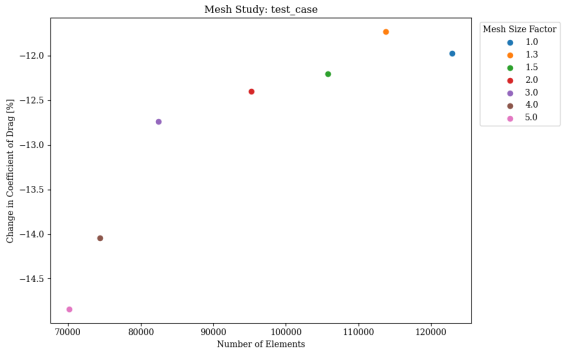
(a) Table 1 from [49]. Comparing the Strouhal number of circular cylinders in cross flow at a Reynolds number of 100 found by different studies.



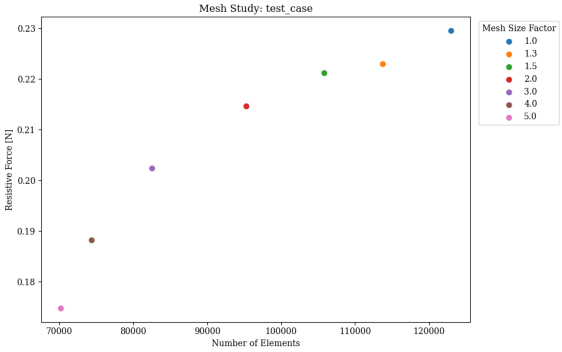
(b) Figure 5.23 from [50]. Comparing the Strouhal number of a circular cylinder in crossflow over a range of Reynolds numbers.

Figure 2.6: Validating Strouhal number.

Next, a test case in the center of the initial design space was created. A mesh sensitivity study was conducted.



(a) Mesh study convergence of change in coefficient of drag



(b) Mesh study convergence of resistive force.

Figure 2.7: Test case mesh study. Amplitude of oscillations, $\omega_{amp} = 5.05$, frequency of oscillations, $f = 0.6$, at $Re = 100$.

These results also show good mesh convergence at a mesh factor of 1.0 for both coefficient of drag and resistive torque.

Studies on oscillating a cylinder around its central axis to reduce drag force are used for the validation of these results. Below is a table from [6] showing the reduction of drag by oscillation using a control parameter (λ).

$$\omega(t) = -\lambda C_L(t) \Longrightarrow \lambda = -\frac{\omega(t)}{C_L(t)}$$

At $Re = 100$, $\omega_{amp} = 5.05$, $f = 0.6$ and a mesh size factor of 1.0; $\overline{C}_D = 1.200$.

Re	60	80	100	150	200
$\lambda = 0.00$	1.412	1.369	1.347	1.328	1.337
$\lambda = 0.25$	1.388	1.336	1.305	1.269	1.259
$\lambda = 0.37$	1.377	1.322	1.289	1.248	1.233
$\lambda = 0.50$	1.367	1.310	1.276	1.231	1.211
$\lambda = 0.63$	1.358	1.300	1.266	1.219	1.196
$\lambda = 0.75$	1.351	1.295	1.260	1.212	1.186
λ_c	1.350	1.293	1.256	1.207	1.179

Figure 2.8: Table VI from [6]. Reynolds number, control parameter (λ) and coefficient of drag from an oscillating cylinder are tabulated here.

Since this study using an active control to determine the oscillation pattern of the cylinder, an exact comparison is not possible. However, if we take the root-mean-square of the coefficient of lift for the stationary case and the root-mean-square of the oscillation's angular velocity we get:

$$\frac{RMS[5.05sin(2\pi 0.6t)]}{RMS[C_{L,stat}]} = 14.9456$$

This approximation of the lambda value would fall beyond tabulated lambda values, which stop at $\lambda = \lambda_c = 1$. Still, the trend in the coefficient of drag at $Re = 100$ suggests an asymptotic relationship at around 1.2. If nothing else, this study shows that our results are comparable to another numerical simulation of an oscillating

cylinder.

Re	Ω	St_f	$M \times N$	Δr_b	$\Delta \theta_b$	Δt	St_{vs}	$\bar{C}_D$	C'_L
			241×241	0.0179	1.42°	0.02	0.4	1.231	0.299
			301×301	0.0068	1.13°	0.02	0.4	1.236	0.304
100	2.0	0.4					(0.0)	(0.41)	(1.68)
			241×241	0.0179	1.42°	0.01	0.4	1.231	0.299
							(0.0)	(0.00)	(0.00)
			301×301	0.0068	1.13°	0.02	0.6	0.614	0.194
			241×241	0.0179	1.42°	0.02	0.6	0.670	0.135
							(0.0)	(9.12)	(30.4)
1000	2.0	0.6	361×361	0.0026	0.94°	0.02	0.6	0.613	0.201
							(0.0)	(0.16)	(3.61)
			301×301	0.0068	1.13°	0.01	0.6	0.615	0.202
							(0.0)	(0.16)	(4.12)

Figure 2.9: Table I from [5]. A mesh study at a consistent amplitude and frequency of oscillation. Convergence of mean coefficient of drag at around $\overline{C}_D = 1.231$

To further validate this case, the results of [5] are used. The amplitude and frequency, $\omega_{amp} = 2.0$ and $St_f = 0.4$, were used to run another test case. However, the coefficients of drag do not match. $\overline{C}_{D,prev} = 1.231$ and $\overline{C}_{D,pres} = 1.338$. This discrepancy could be due to the use of DNS over LES. The model used by [5] is a DNS, and the small scale of turbulence this model solves for may be the source of the discrepancy.

Finally, [41] simulated a rotating cylinder at $\omega_{amp} = 2.0$ and $St_f = 1.0$ in a $Re = 150$ flow. The authors tabulated a coefficient of drag of $\overline{C}_D = 1.47$. When this simulation was performed using the present model, the result was $\overline{C}_{D,pres} = 1.339$. This result again show a discrepancy in results and full validation can not be achieved. However, [41] uses the viscous vortex method, not an LES. LES simulations are known to have more reliable results.

The boundary cases were all found to have good mesh convergence at a mesh size

factor of 1.0. More figures of the mesh studies can be found in the **Appendix>Mesh Studies>Oscillating Cylinder**.

2.2 Breathing Jet Simplification

2.2.1 Background and Motivation

Modeling the distribution of respiratory droplet aerosols [from [human] breathing] is fundamental to understanding the transmission of airborne infectious diseases and to establish effective mitigation techniques. The highest risk scenarios for aerosol transmission of infectious diseases occurs in indoor spaces. Compared to outdoor spaces, indoor spaces have a finite volume which can trap the aerosols inside the space until they are exhausted through ventilation. In addition, indoor spaces contain flow fields with much less energy compared to outdoor spaces. This means that higher concentrations of respiratory aerosols will persist for longer in indoor spaces than outdoor spaces. The longer high concentrations of aerosols persist in a space, the more likely transmission will occur.

The most desirable scenarios to model are those with the highest risks of aerosol transmission. An example of a common high risk scenario would be a crowded room with poor air ventilation and no air purification. Ideally, the regular breathing, the inhales and exhales, of each individual would be modeled. In many such scenarios, multiple individuals would be vocalizing regularly, and there may be the occasional cough or sneeze for some individuals. Modeling such a complex scenario has many challenges, but if this could be achieved, then it would be extremely informative to developing ideal mitigation techniques. A validated model of such a space would, for example, allow for the rapid testing of ventilation/purification placement and strength in a range of different scenarios. Such a model could not only be used to mitigate

the airborne transmission of infectious diseases, but also to better understand how to improve indoor air quality in general. Applications beyond mitigating the transmission of airborne disease include low cost ventilation of cooking spaces, classrooms and in general poorly ventilated indoor spaces. Lowering the cost of ventilation is a balancing act between the forcing energy cost, climate control and the ultimate goal of improving indoor air quality.

The crux of a creating a CFD model for this problem lies in the computational cost of simulating the dispersal of respiratory droplets as they exit the lungs in a processes called atomization. Atomization is the processes of a bulk liquid breaking into small droplets, which are called aerosols. In the case of infectious diseases, the disease lives in the mucus membrane that coats the inside of the respiratory tract. This mucus is atomized by the movement of the membrane (as the lungs contract) and the high velocity air that passes over the membrane when the subject is vocalizing, coughing, sneezing or simply breathing. High fidelity atomization modeling is achieved using computationally intense CFD solvers.

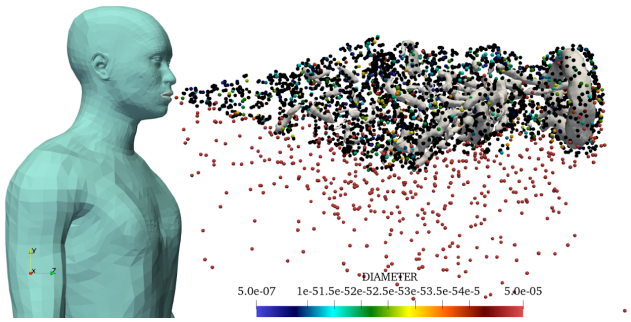


Figure 2.10: Extremely high fidelity simulation which uses a LES model of a zero net mass flux free jet with aerosols modeled as particles with mass and manikin geometry.

Other studies have evaluated the fidelity of simpler, lower cost models of human

breathing in indoor spaces. The human models are called computer simulated persons (CSPs). [51] developed a model of a small, poorly ventilated room containing one human sleeping on a cot in the corner. The boundary conditions of the room are measured and a fairly accurate model is achieved. However, this scenario is far from the high risk of infection scenario described above. The room is a fairly stable region with low Reynolds number flows that are less computationally expensive to simulate. In scenarios with minimum ventilation, $Re < 100$, a single person's boundary conditions, such as wall temperature, may have a more significant effect on the steady-state flow pattern than a room with flow dominated by ventilation and multiple people breathing.

[52] created a model of two individuals breathing in the vicinity of one another. This study was unable to provide a high level of agreement between the experimental and computational results. Instead, this study explores the significance of different CFD simulation parameters. The conclusion was that the grid, or mesh, sizing plays a significant role in the results of the simulation and care should be taken to try to achieve mesh independent results. Other significant factors were related to boundary conditions. The model was very sensitive to the convective heat outputs, cross-sectional exhalation area, and the pulmonary ventilation rate.

Over 20 years separates this study from the present one. In that time, it has become more practical to use higher fidelity models, as discussed in the **Introduction** section above. There have also been incremental improvements in the development of similar lower fidelity, RANS, models. With a lack of significant improvement to RANS models through traditional CFD modeling, other data-driven approaches have been explored with increasing interest in recent year. For a comprehensive review of

these recent studies, see "Turbulence Modeling in the Age of Data" [26].

The data-driven approach proposed in this study is based on the concept that, ideally, these simulations would use what could be called a "spacial" multi-fidelity model. Different spaces in the discretized domain would use different schemes with different levels of fidelity. In this case, it would be ideal to use a high fidelity scheme for the area around the mouth jet, while a lower fidelity scheme could be used for the rest of the room. This is ideal because the majority of the mesh would be making low fidelity, simpler, faster calculation that still produce accurate results. This is possible because of the lack of turbulence in these relatively low Reynolds number flow areas far from the breathing jet. However, around the mouth's breathing jet, it would be ideal to use a high fidelity scheme that fully captures the turbulence present in this relatively higher Reynolds number flow. The separate schemes would likely require different time stepping, and the exchange of flow information across the low to high-fidelity boundary would be extremely complicated. As far as this author knows, there has not been any success in developing such a model. Instead, multi-fidelity models tend to involve an iterative comparison between high-fidelity and low-fidelity results, where corrections are made to the low-fidelity simulation based on the discrepancies between the two models.

Whenever a separate model is used to inform the results of the primary model, the separate model is sometimes called a surrogate model. Both multi-fidelity and surrogate model are relatively new terms to the field of CFD. There seems to be no precise definition of either and they are sometimes used interchangeably. As far as the author can gather, a multi-fidelity simulation is a low-fidelity model which is informed by a higher-fidelity model; this higher-fidelity model can be called a surrogate model

or vice-versa. [53] [54]

The approach of this study is to consider the boundary conditions that are highly sensitive to the results of simulation as potential parameters. These parameters can be varied in order to develop a model that produces desirable, accurate results but does not necessarily seek to represent the boundary conditions actually found in the physical world. In other words, if RANS models are incapable of capturing accurate results given realistic boundary conditions, why not adjust these boundary conditions to whatever conditions produce the most accurate results. The obvious potential problem will be generalizability. If we fit low fidelity boundary conditions to achieve results comparable to high fidelity results, will these "fitted" conditions hold true as the scenarios being modeled change? If we want the multi-fidelity simulation to produce accurate results, how many high-fidelity simulations must be conducted to inform it? Previous work has highlighted that one big problem with data-driven models such as this one. That is, they are able to perform well when interpolating between the results of the high-fidelity data, but perform poorly when trying to extrapolate beyond the high-fidelity data set. It is expected that this approach will be no different. However, by utilizing the flexibility and convenience of the *pymooCFD* platform and the generalizability of the multi-fidelity modeling approach itself, perhaps this approach can allow for the rapid development of simplified models. This is in contrast to the often extremely problem-specific multi-fidelity simplification approaches that have been used in previous work.

2.2.2 Modeling Simplifications

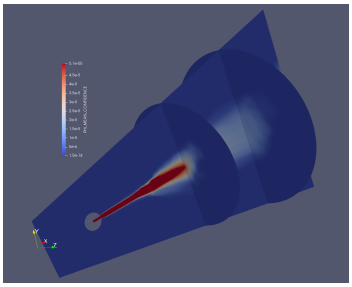


Figure 2.11: High fidelity, LES, zero net mass flux free jet simulation. Profiles of passive scalar injected at mouth inlet boundary.

There are many simplifications that must be made to the current high fidelity model to reduce computational cost and make the model practical for developing mitigation techniques. The goal, as always with model simplification, is to maintain the highest level of fidelity possible while reducing the computational cost of solving the model. The most significant simplification that must be made, so the model is practical to solve, would be using of the Reynolds-Average Navier-Stokes (RANS) equations. This is in contrast to the high fidelity simulation which solves the full Navier-Stokes equations.

The flow field close to the mouth is complex and turbulent due to the free-stream jet's pulsatile nature, as well as the high velocity and temperature of the jet compared to the rest of the domain. Eventually, the high energy complex turbulent flow field found near the mouth becomes a lower energy, steady flow field further from the mouth. A RANS model is incapable of fully capturing the complex turbulent flow field that is produced near the mouth by periodic breathing, coughing, sneezing or vocalizing. However, the flow field far from the mouth is something a RANS model

is capable of capturing.

In addition, there are high computational costs associated with periodic boundary conditions. Therefore, using a lower fidelity RANS model of a periodic flow is counter to the goal of reducing the computation cost of the model. For this reason, formulating a model which represents the periodic flow that is breathing, coughing, etc. as a steady flow would be highly beneficial. There is precedent for this in previous work such as [51].

So further simplification of the breathing pattern used by the high fidelity simulation is needed. Let's begin by discussing how the breathing pattern is modeled in the high-fidelity simulation. The high fidelity simulation used a periodic boundary condition, which modeled each breath as having the same volume of fluid exhaled as inhaled. In fluid dynamics terms, this can be called a zero net mass flux (ZNMF) free stream jet. (When working with an incompressible fluid zero mass flux, it is the same as a zero volume flux.) To model breathing, the inhales are "sharper" than the exhales. This means inhales peak at a higher speed but take place over a shorter amount of time than exhales. This allows the condition of zero mass flux to be maintained. However, this periodic boundary condition creates instabilities in the flow field near the mouth that are computationally intense to resolve. With the goal of creating a model that is less computationally intense, and with the understanding that focusing on long term, time-independent flow patterns is acceptable, the simplification to a steady-state model would be beneficial. In order to achieve a time-independent solution, the periodic boundary condition must be simplified to a steady stream jet.

At this point in the simplification process, the ability to properly model the near-field of a ZNMF jet is not possible. However, there is still hope of creating a model

where a steady jet creates a flow field far from the head that closely matches the flow field of a periodic jet far from the head. We know this is possible because the far-field of a ZNMF jet has been shown to be similar to that of a steady jet. [55]

Another simplification that can be made is geometric. The significance of the manikin geometry to the dispersal of respiratory aerosols inside an indoor space is minimal. Therefore, a sphere with a cylindrical channel cut out of it is proposed. This is similar to the geometric simplification used in [56].

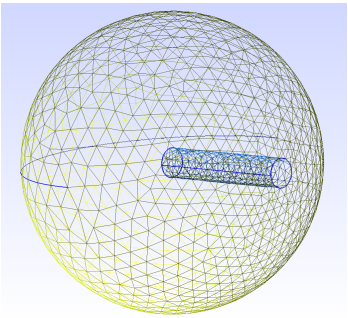


Figure 2.12: Geometric simplification of head and mouth generated by Gmsh. The mesh seen here is for easy visualization of geometry and is not the mesh used in the simulation.

A co-flow, $U_{co-flow} = 0.005\frac{m}{s}$, in the positive x-direction is used as the boundary condition for the cone's wall and for the back-wall, behind the jet. This boundary condition prevents back-flow, which would occur if the boundary condition was a pressure outlet. Another possible boundary condition would be a slip wall; however, this boundary condition could generate back flow at the inlet behind the sphere. Therefore, the more numerically stable option is a co-flow.

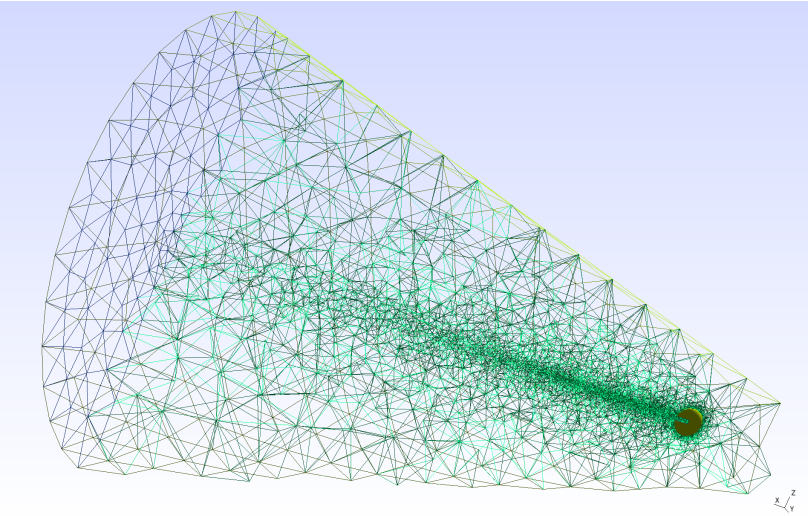


Figure 2.13: 3D conical domain used by LES simulation. Again, the mesh seen here is for easy visualization of geometry and is not the mesh used in the simulation.

Next, an irregular breathing pattern was generated to model a human being breathing over a long period of time. An irregular pattern was chosen over a regular breathing pattern because not only is it generally a more realistic model, but the regular interval breathing pattern caused an undesirable flow phenomenon. The vortex rings generated by the breathing pattern deviated from the center-line in one particular direction. This biasing of the flow field in one direction was likely caused by a perturbation from irregular mesh element sizes inside the sphere's cylindrical passageway. A perturbation combined with a regular frequency in the flow field has been known to cause large deviations from what is excepted to be symmetrical flow fields. Over time, this deviation is often followed by a biasing of the flow field in the opposition direction, sometimes called bi-fluctuation. This could eventually create the desired symmetrical flow field. However, these types of phenomena can have all kinds of different potential modes with varying time-scales. More about these types of flow phenomena can be found in [57], [58] and [59]. More investigation is

needed before definitive conclusions can be drawn about the behavior of this particular phenomenon. The important part is that these deviations from a symmetrical flow, though typically not stable, can have modes with large time-scales. Simulating these large timescale modes is computationally expensive.

To avoid having to conduct this time-consuming investigation, the modeling of breathing velocity was improved to simulate irregular breathing. This irregular frequency was enough to prevent a biasing of the jet flow field in one direction for long periods of time. Hence, an axi-symmetric mean flow field could be generated with much less computational power.

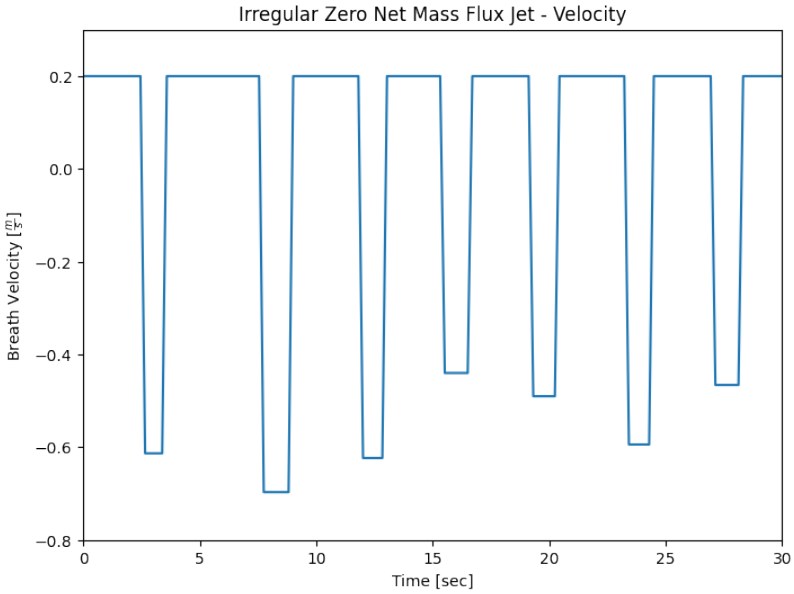


Figure 2.14: Irregular zero net mass flux jet velocity versus time. Models the irregular breathing of a human being.

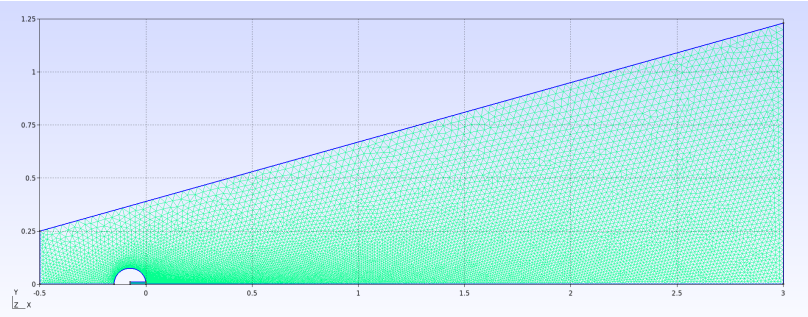


Figure 2.15: Axial-symmetric domain used by RANS simulations.

The peak velocity of the jet is fixed at 0.2 m/s. The duration of exhale has a mean of two seconds and a standard deviation of 0.25 seconds. The inhale has a mean duration of 1 second with a standard deviation of 0.2 seconds. The peak speed of the inhale is determined by the ratio of the exhalation duration over the inhale duration, upholding the net-zero mass flux boundary condition.

2.2.3 Methodology

The simplification described in the previous section leaves the problem of modeling a high fidelity pulsatile jet as a steady RANS jet with the caveat that only the far-field can be modelled properly. To accomplish this high fidelity to RANS simplification, a novel approach is proposed. Using the *pymooCFD* platform, boundary condition parameters of the steady RANS jet can be adjusted with the objective of minimizing the difference between the pulsatile jet's far-field and the steady jet's far-field.

A high-fidelity simulation, or high-quality (HQ) simulation, was conducted using a 3D mesh generated in Gmsh and the YALES2 incompressible solver. The details of this simulation will be laid out in the next section. In order to quantify the difference between flow fields on two different meshes, an interpolation onto a "universal" grid

had to be conducted using the data from the RANS and YALES2 simulations. The YALES2 simulation was first interpolated onto a 3D Cartesian grid with a resolution of 200 cells per meter. The grid is 1x1x1 meter block, making it 200x200x200 nodes. Next, this grid was radially averaged and interpolated once again onto a Cartesian 2D grid with a resolution of 200 cells per meter; however this grid was 1x0.5 meters, making it a 200x100 node grid. This is the "universal" grid, and each RANS simulation is interpolated onto this grid in post-processing. Finally, the high-fidelity simulation flow field, which has been reduced to a 2D radially averaged grid, is compared to the low-fidelity simulations flow field, which has been interpolated onto the same "universal" grid. The absolute value of the difference between the quantity of interest at each grid point is found, and the mean across all grid points is calculated.

x^n : quantity of interest at each node $\qquad$ N : total number of nodes

$$\Delta_{avg} = \frac{\sum_{n=0}^{N} |x_{hq}^n - x_{lq}^n|}{N}$$

The quantities that will be used to evaluate the difference between the high and low quality simulations are magnitude of velocity and mass fraction of a passive scalar. A passive scalar refers to a quantity that is injected into the flow and transported by the flow, but has no effect on the flow's behavior. In this case, a passive scalar is injected at a mass fraction of 100% into the domain from the mouth inlet. The passive scalar is then dissipated into the domain by the flow, and the mass fraction distribute provides a rough estimate of the behavior of respiratory aerosols.

A linear interpolation scheme is used for the LES 3D interpolation and a cubic scheme is used for the RANS 2D interpolation.

2.2.4 Verification and Validation

A mesh study is conducted on the YALES2 LES simulation of the 3D free jet. The evolution of the jet's velocity profile along the y-axis at different constant x locations are graphed below.

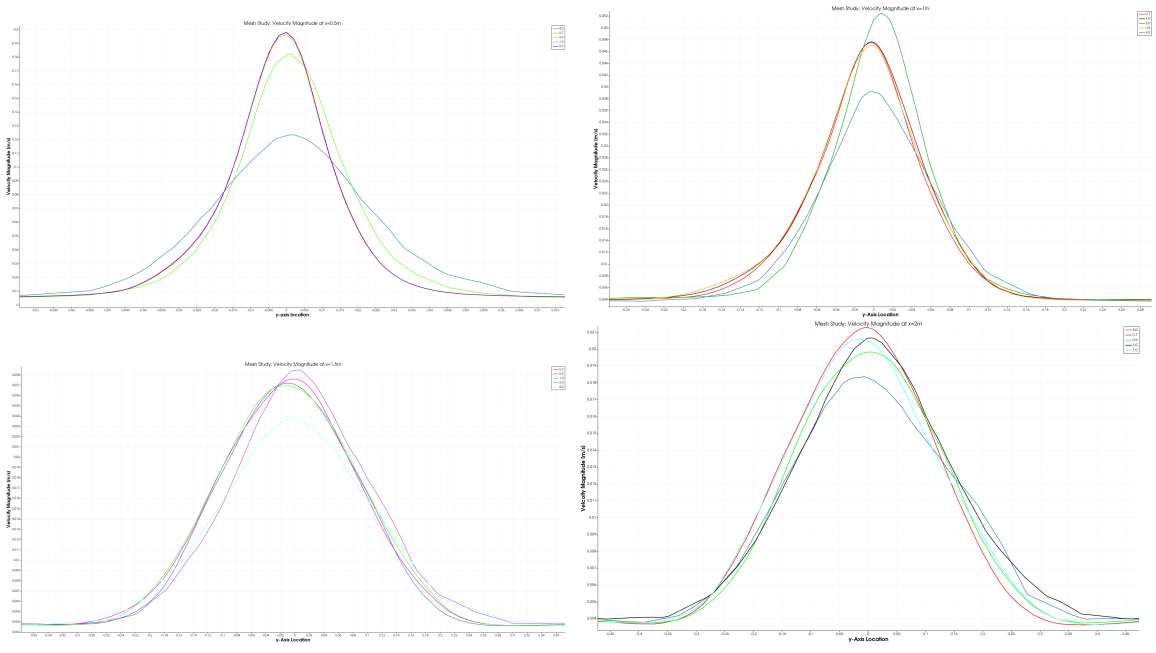


Figure 2.16: Evolution of the velocity magnitude profile. The legend indicates the mesh size factor used to scale the mesh size.

These graphs show a convergence of the profile at around a mesh size factor of 1.0. The GMSH mesh scheme multiplies the global mesh size factor, indicated in the legend, by the target mesh element sizes. The mesh is also controlled by transfinite conditions, where the number of nodes per curve is specified. With flexible transfinite turned on, changing the mesh size factor generates a somewhat uniform mesh scaling. This is an extremely useful feature of GMSH that allows for mesh studies to be conducted on complicated meshing schemes with relative ease. For more details on

the meshing scheme, reference Section 5.2.4, **Appendix>Code**, below and visit *gmsh.info*.

Though a mesh size factor of 1.0 should be sufficient to capture the LES free jet, the simulation with a mesh size factor of 0.7 was allowed to run for longer, creating more symmetrical results. This is most prominently seen in the profiles furthest from the jet's origin. In y-axis velocity profile at $x = 1.5m$ and $x = 2.0m$ the simulation with a mesh size factor of 0.7 best captures the symmetry that should theoretically exist in a mean free jet profile.

A mesh study was also conducted on the RANS jet test case located in the center of the parameter space.

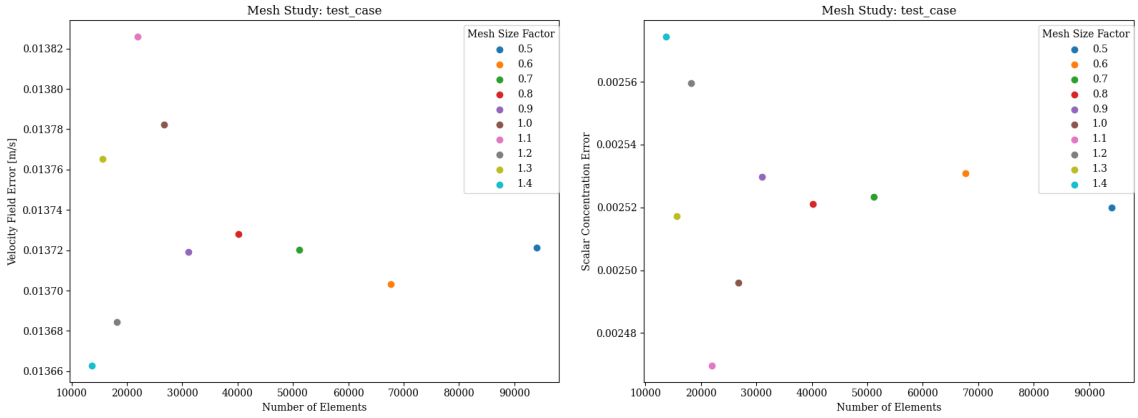


Figure 2.17: Test case mesh study. Breath velocity, $u_{breath} = 0.35\left[\frac{m}{s}\right]$, diameter of the circular mouth opening, $D_{mouth} = 0.021[m]$.

Addition mesh studies were conducted on the parameter space boundary cases.

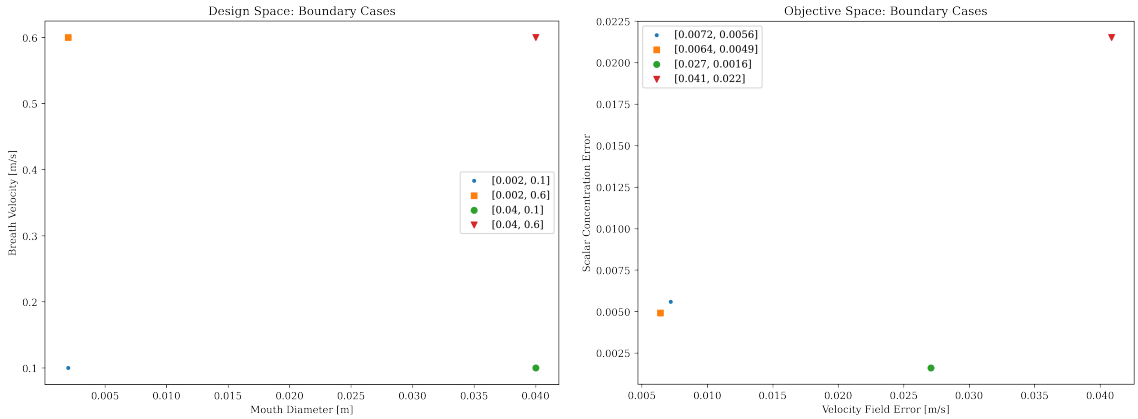


Figure 2.18: Parameter space boundary cases mapped in design and objective spaces.

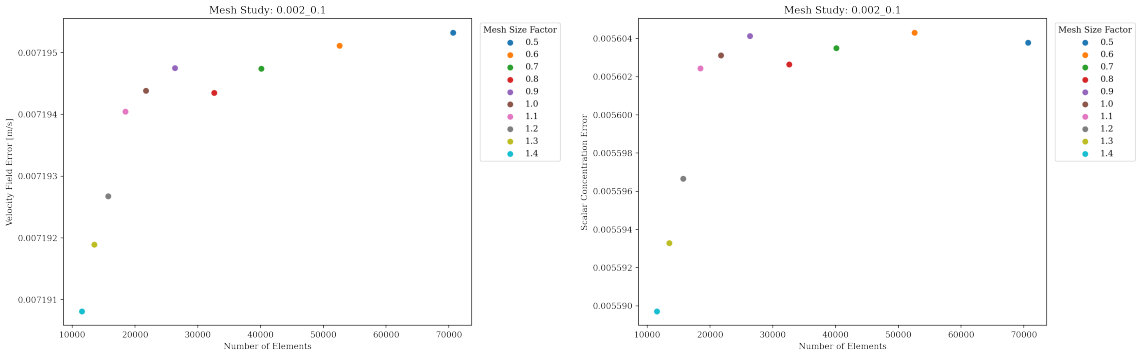


Figure 2.19: Boundary case mesh study. Breath velocity, $u_{breath} = 0.1\,[\frac{m}{s}]$, diameter of the circular mouth opening, $D_{mouth} = 0.002[m]$.

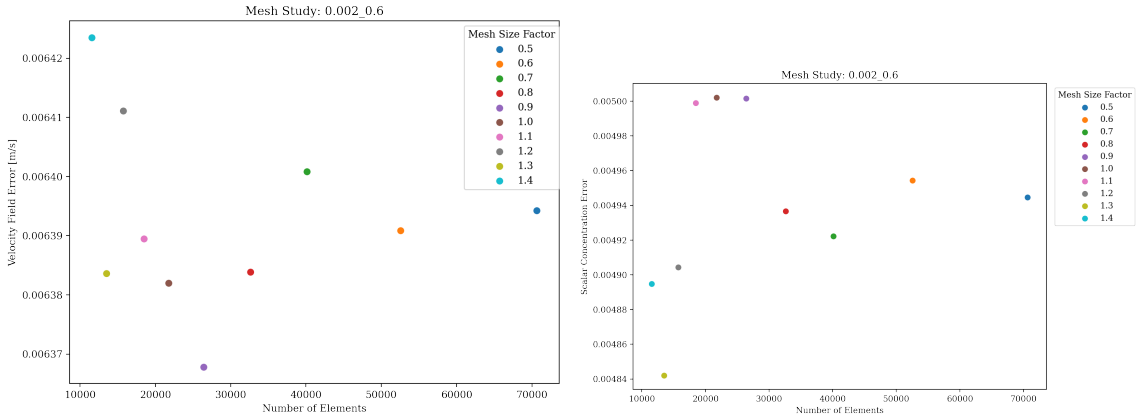


Figure 2.20: Boundary case mesh study. Breath velocity, $u_{breath} = 0.6\,[\frac{m}{s}]$, diameter of the circular mouth opening, $D_{mouth} = 0.002[m]$.

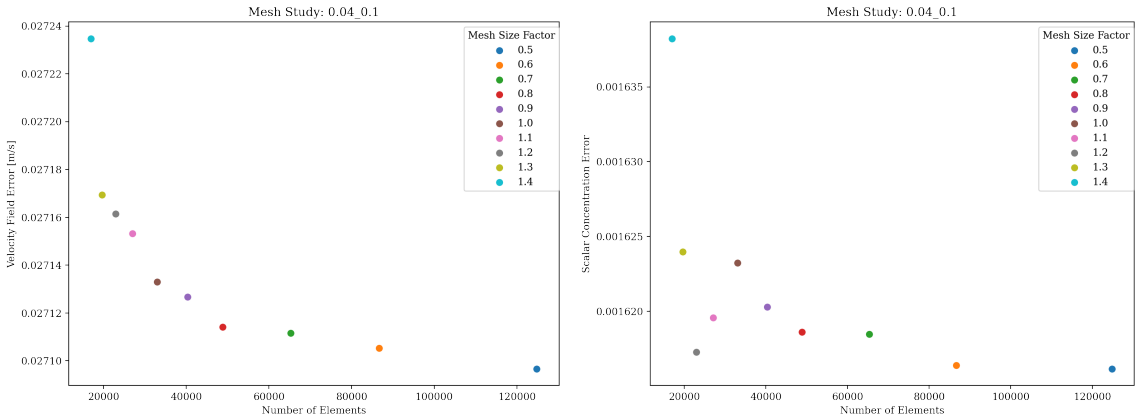


Figure 2.21: Boundary case mesh study. Breath velocity, $u_{breath} = 0.1\,[\frac{m}{s}]$, diameter of the circular mouth opening, $D_{mouth} = 0.04[m]$.

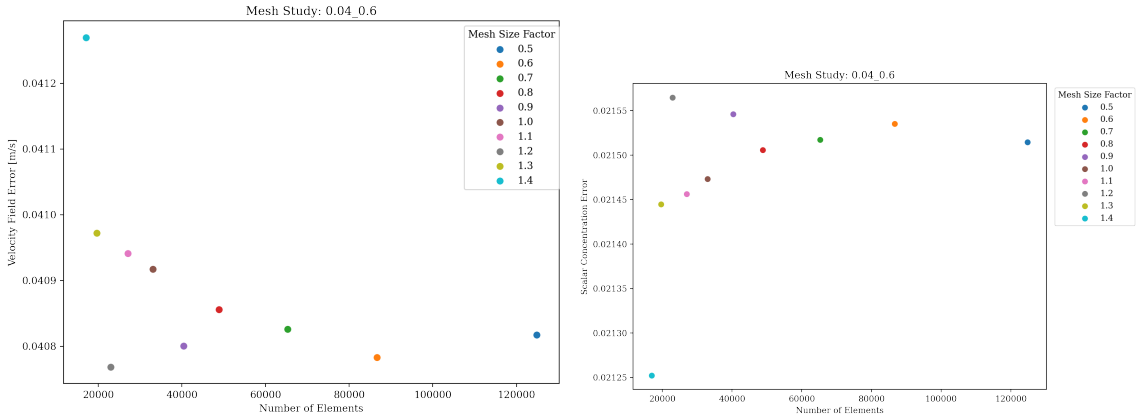


Figure 2.22: Boundary case mesh study. Breath velocity, $u_{breath} = 0.6\,[\frac{m}{s}]$*, diameter of the circular mouth opening,* $D_{mouth} = 0.04[m]$*.*

2.3 Air Purifier Configuration

2.3.1 Background & Motivation

Over the past 100 years, our understanding of the effects of indoor air quality on human health has evolved dramatically, and our buildings have come to reflect this better understanding. Modern commercial HVAC system typically use a minimum of 4 ACH in all spaces. Residential indoor spaces tend to have lower standards. In the USA, many states have a law (IRC-2006) that requires residential spaces to have an ACH of 0.35 for the entire home, which can be met with windows. Additionally, there is usually a requirement for bathroom and kitchen ventilation if there are no windows. Modern homes typically exceed these requirements. Many older buildings in use today have been retrofitted with HVAC systems. However, depending on when this retrofit took place, the modern standards for ACH may not have existed. Older buildings are therefore the most likely to have poor indoor air quality.

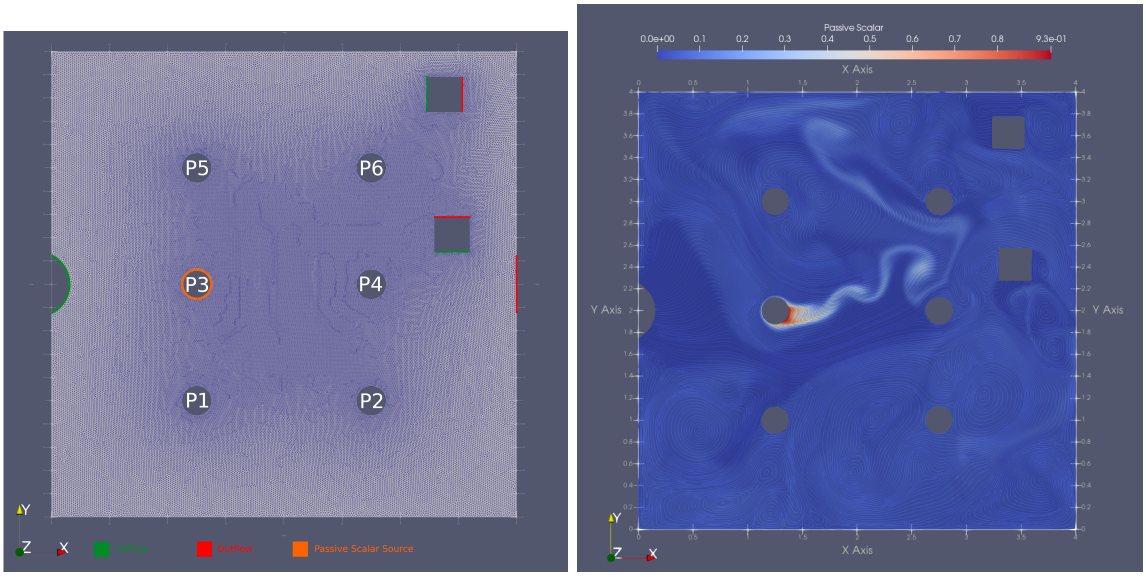
The COVID-19 pandemic has led to a heightened awareness by the public of indoor air quality standards. Portable air purifiers have become an important tool in combating the spread of COVID-19 and improving indoor air quality by offering an easy retrofit to any indoor space. This is an affordable and effort-less option for improving indoor air quality compared to the cost of modifying or re-configuring an existing HVAC system. Though it is important to note that air purifier are one part of maintaining high indoor air quality, fresh outdoor air exchange is still needed to stop the build up of carbon dioxide. [60] For more information on indoor ventilation standards, visit the Home Ventilating Institute's website, HVI.org. [61]

The motivation behind this optimization study are the questions that come next. How can portable air purifiers be best configured within an indoor space to reduce the exposure of the occupants to contaminants? How many air purifiers should be used? Where should I place them? How powerful should they be? What direction should they face? This study will attempt to answer these questions. Other researchers have conducted similar studies. [62] use numerical simulations as well to explore the effect of placing an air purifier in a room. This study was specific to a music room; the simulations used 3D geometry with aerosols modeled as particles with mass using an Eulerian-Lagrangian framework. The computational cost of this model was 384 CPU hours per 11 minutes of simulation time. Each scenario tested runs until 36 minutes are simulated. Therefore, more than 1152 CPU hours would be required per simulation. For reference an entire cluster on the Vermont Advanced Computing Center, the HPC used for this work, has a total of 3144 CPUs. So, running a single simulation on an entire cluster is estimated to take about 2 hours and 45 minutes. Looking back at Equation 1.5 and even Equation 1.6, which assumes the ideal computational resources are available, it is apparent that an optimization study run time using this CFD model is impractical and could take months.

This study looked at the effect of an air purifier in three different location within the music room. Alarmingly, one air purifier location creates increased exposure for the occupant of interest compared to the case with no air purifier at all. This indicated that not only does optimal air purifier placement reduce energy and material costs, but improper air purifier placement can lead to a scenario worse than using no air purifier at all. [63]

2.3.2 Modeling Simplifications

The most significant simplification made in modeling a room with air purifiers is, of course, the reduction of 3D to 2D. This simplification brings the simulation run time to between 1-2 hours, making it feasible for an optimization study. Similar 3D models took up to 8 hours per simulation. The use of a 2D simulation to represent a 3D space makes this study qualitative, not quantitative. The only situation where a 2D simulation can produce reliable quantitative results is when the third dimension is significantly larger or smaller than the other 2 dimensions, such as airfoil or global weather simulations. The 2D simplification in CFD is known to under-predict momentum diffusion because there is one fewer dimension for the momentum diffusion to occur in. This shot coming is compensated for by other modeling decisions discussed below.



(a) Mesh and boundary conditions. *(b)*

Figure 2.23: 2D room with 2 air purifiers.

The model consists of 6 subjects represented by evenly spaced circles throughout the domain. The room ventilation inflow is represented as a semicircle to simulate the diffusers found at room ventilation inlets. A ventilation inlet with flat geometry would produce a free jet into the room. This would fail to capture the flow pattern caused by the vent diffuser. In addition, the vent's semicircle geometry increases the momentum diffusion of the vent inflow, which is under-predicted by 2D simulations. The subject closest to the room inflow was chosen as the infected individual. The surface of this circle is the source for 100% mass fraction of the passive scalar.

Another scalar is used to add vertical velocity in both directions to increase mixing. This is essentially the same as imposing a temperature gradient between the lower and upper walls. A scalar with values between -0.5 and 0.5 is used. The scalar has one source at the lower wall at a value of 0.5 and another source at the top wall at a value of -0.5. At each node, this scalar is multiplied by a buoyancy factor of 0.002 and added to the y-component of the velocity.

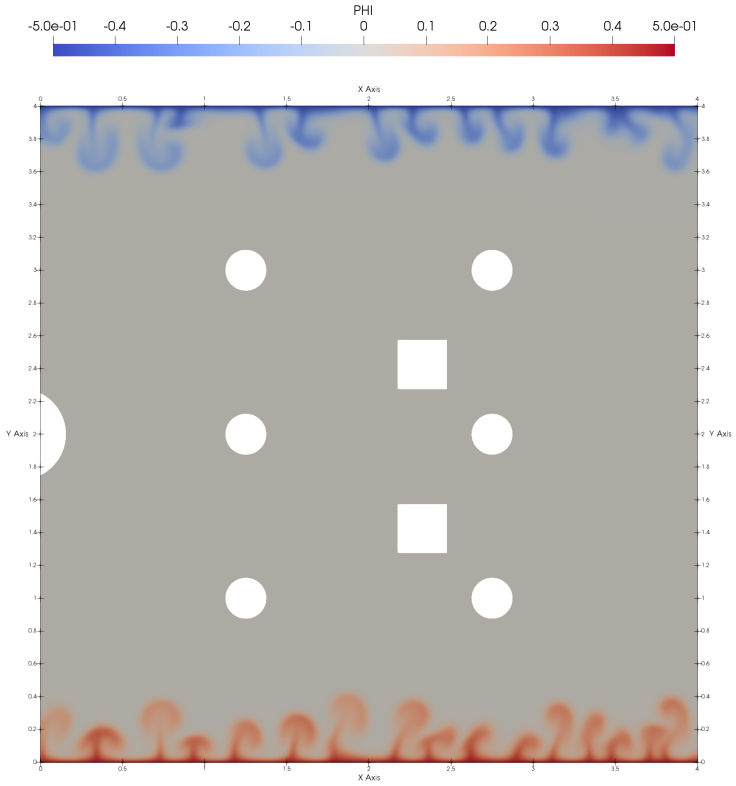


Figure 2.24: 2D room with air purifiers at simulation time of 250 seconds. Capturing scalar used to mimic buoyancy forces and increase mixing.

The air passes through the air purifier outlet and inlet at the same speed, acting as a momentum source. Any passive scalar that enters the air purifier is destroyed completely, with no contaminants returning to the domain at the air purifier outlet. Therefore, these air purifiers are assumed to operate at 100% efficiency. This assumption is actually realistic because high efficiency particulate air (HEPA) filters operate at 99.97% efficiency for particles 0.3 microns in diameter. Since 0.3 microns is the most penetrating particle size, HEPA filters operate at higher efficiency for any other size particles. [64]

2.3.3 Methodology

The middle left subject was chosen as the infected individual to study what is likely the worst case scenario. This is thought to be the worst case scenario because an infected individual is in front of a room's ventilation inflow. This would likely cause the infected respiratory droplets to have maximum distributed throughout the room.

The simulation is run with the infected subject producing passive scalar the entire time. The first half hour, 1800 seconds, of the simulation is discarded. The average exposure of the other 5 subjects from 1800 seconds to 7200 seconds is calculated. Then these averages are averaged together. This produces a metric representing the mean exposure of the subjects. Minimizing this metric is the first objective.

$$\text{mean subject exposure} = \frac{\sum_{subj.} \frac{\sum_{t_i}^{t_f} \frac{\sum_{surf.nodes} z}{N_{surf.nodes}}}{t_f - t_i}}{N_{subj.}}$$

Another competing metric was chosen, minimize air change per hour (ACH). This metric is commonly used when designing indoor ventilation and can be calculated in different ways. For this study, ACH will be how many room volumes worth of fluid passes through the room outlet or through an air purifier.

$$ACH = \frac{\sum_{APs} \int u_{in} \rho_{air} dA_{in} + \sum_{outlets} \int u_{out} \rho_{air} dA_{out}}{V_{room} \rho_{air}} \tag{2.6}$$

In this case of incompressible flow, a single outlet and only 2 spacial dimensions, the equation becomes similar.

$$ACH_{2D} = \frac{\sum_{APs} u_{in} L_{in} + u_{out} L_{out}}{A_{room}} \tag{2.7}$$

Since higher ACH generally means lower average exposure of the subject, the addition of this air change per hour objective should create a Pareto front where the best configuration of air purifier(s) is found at a range of ACH values. To adjust the ACH of the room ventilation is kept constant and the ACH of the air purifier(s) is adjusted. This is because the study's motivation is to inform the use of air purifiers as an inexpensive retrofit to indoor spaces. Therefore, by only adjusting the air purifier ACH, we better model a room with poor ventilation being augmented by air purifiers. The aim is to help inform what strength air purifier should be chosen for a given space or, vice versa, what can be accomplished with air purifier(s) of a certain strength.

One challenge faced when setting up this optimization problem was restricting the air purifiers x, y location parameters. The air purifier(s) can only be placed in between the circles representing the people and not too close to the outside walls. The outer limits are easily restricted with the upper and lower bound variables used by *pymoo*. Restricting these coordinate parameters internally so that they can not be placed on top of or too close to a person was more challenging.

The solution was to use the *Repair* class that is part of the *pymoo* framework. This class is used to "repair" each population's parameters after they are generated, but before they are evaluated. A function was created within a custom Repair class. This function finds coordinates that are too close to the circles, then randomly changes that coordinate to one side of that person or the other.

Another solution would have been to create an integer parameter that placed the air purifier in a selected set of positions corresponding to an integer value. This custom *Repair* class method was chosen to provide as many air purifier locations as possible for the optimization algorithm to explore. The hope is that this will generate

air purifier configurations that are non-intuitive but produce desirable results.

2.3.4 Verification & Validation

The modeling simplification described above create a model that will only be capable of producing qualitative results, not quantitative. Therefore, validation of the model is not possible.

Chapter 3

Results & Discussion

3.1 Oscillating Cylinder

3.1.1 Pre-Process

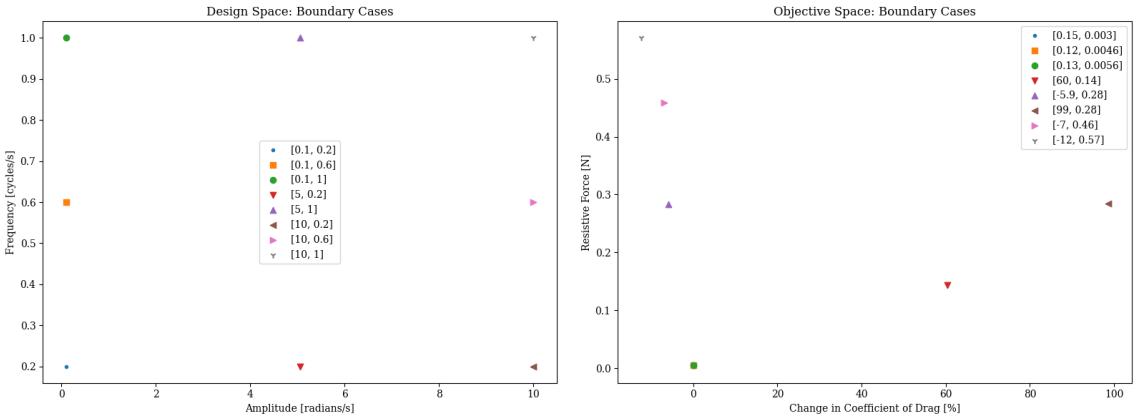


Figure 3.1: Oscillating Cylinder Boundary Cases - Parameter and Objective Spaces

The boundary cases show that the parameter space limitation will be capable of capturing the desired Pareto front. By closely examining the mapping from parameter

to objective points, correlated by color and shape, you can start to see that parameters with frequencies and amplitudes near the low limit of the parameter space will yield results that are undesirable. The boundary cases indicate that these lower end amplitudes and frequencies will not produce a reduction in the coefficient of drag.

The low amplitude cases map to a cluster around $[0, 0]$ in the objective space. Low amplitude means very low velocity rotation is applied; so it makes sense that these cases will produce almost no change compared to the stationary cylinder case. The low frequency of oscillation cases map to increases in the coefficient of drag and a non-negligible amount of resistive force. Therefore, these results will also be undesirable. However, these cases are still informative. The red upside-down triangle, center bottom of the parameter space with $A_\omega = 5$ and $f_\omega = 0.2$ shows potential. There is an increase in both drag and resistive force but it does significantly better than the brown left-pointing triangle in the lower right corner, $A_\omega = 10$ and $f_\omega = 0.2$. This indicates there are potentially predictable trends in coefficient of drag when increasing the amplitude at a fixed frequency. Further conclusions can be draw once the entire first generation is run and mapped to the objective space.

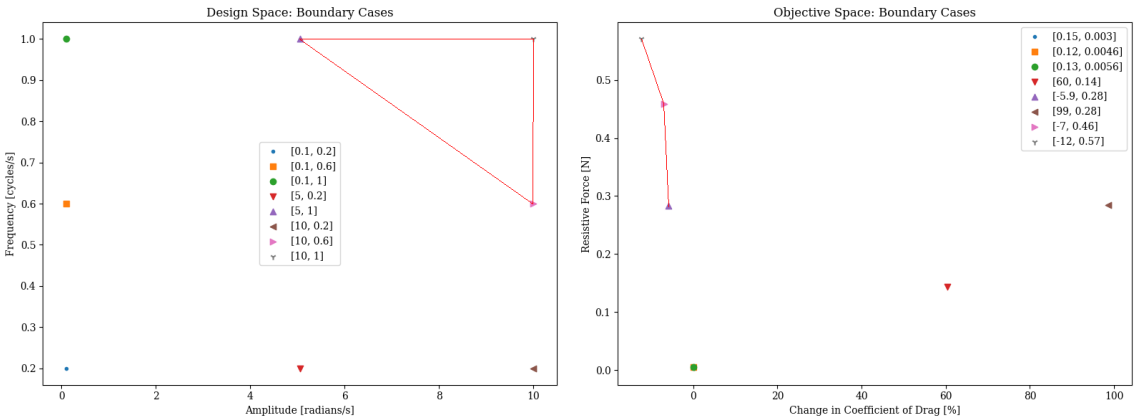
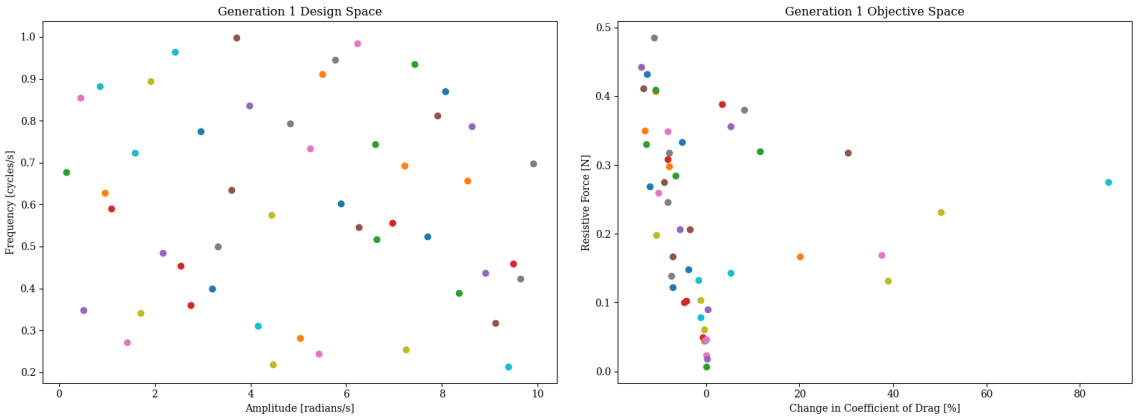


Figure 3.2: Oscillating Cylinder Optimization Study - Boundary Cases

In Figure 3.2, above, the boundary cases which produce the most desirable results in the objective space are connected into what could be called the boundary cases' Pareto front. One can assume that the most desirable portions of parameter space will lie in between these three points that form the boundary cases' Pareto set. However, this problem is relatively straightforward compared to some optimization studies. Some relationships here are fairly linear, making it relatively easy to imagine how parameter will map to objectives. This type of analysis may not be possible in other optimization studies.

At this point in the optimization study it can be beneficial to expand or contract the parameter space before moving onto the next step in optimization study pre-processing, mapping generation 1. Here, it would be pretty safe to increase the lower limit of the amplitude and frequency so that every case run along these boundaries are not moot. However, since this case is an example meant to evaluate the performance of *pymooCFD*, we will pretend that either (a) this was not spotted, (b) that we *do* want to capture this portion of the Pareto front or (c) that we want to be thorough and make our parameter space decision after mapping generation 1.



(a) *(b)*

Figure 3.3: Oscillating Cylinder Optimization Study - Generation 1 Objective and Design Space.

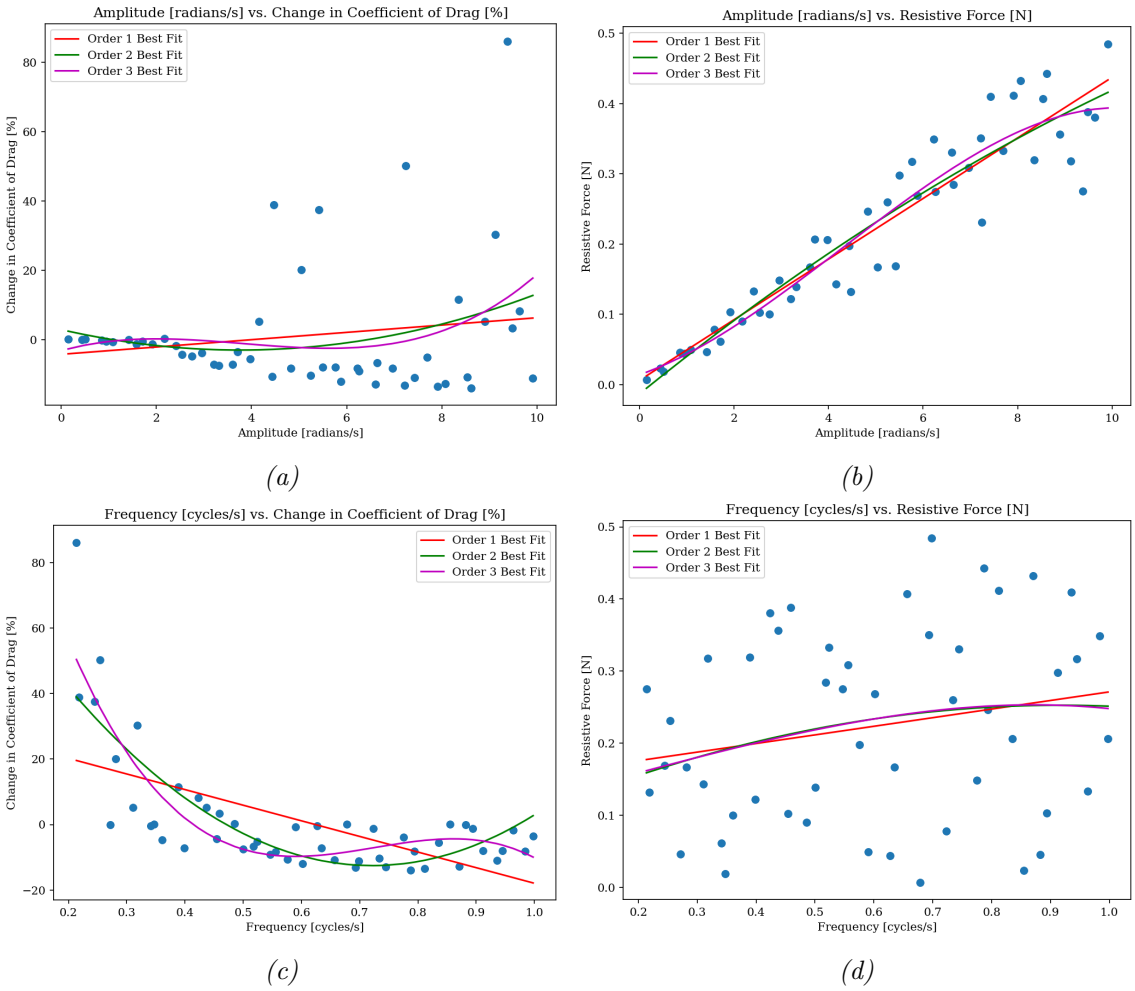


Figure 3.4: Oscillating Cylinder Optimization Study - Parameters versus objectives for generation 1; which is a Latin hypercube sampling of the entire parameter space. First through third order best fit lines plotted as well.

When evaluating the mapping generation 1 graphs, it is important to remember that the other parameters not labeled on the individual graphs are also varying. This means any strong correlations found on these graphs are strong enough to exist while the other parameters are being varied simultaneously.

The most striking relationship is the strong linear correlation between amplitude and resistive force. This is to be expected as the resistive force should be proportional

to the integral of the absolute value of the sinusoidal curve that maps the cylinder's oscillation.

$$F_{res} \propto \frac{1}{T} \int_0^T |A_\omega sin(2\pi f_\omega)| dt$$

In Figure 5.9d, frequency and resistive force seem to have a direct relationship, similar to amplitude and resistive force; this too was to be expected. However, the correlation between frequency and resistive force is far weaker than that of amplitude and resistive force. This tells us the amplitude is the dominant factor in changing the resistive force.

If the outliers are eliminated in Figure 5.9a you can also see a strong correlation between an increased amplitude and a reduction in drag. Again, this makes sense because, in general, the more energy consumed by the flow control technique, the more effective it should be. This is the trade-off we set out to capture in our Pareto front. The existence of the outliers, one of which increases the drag by over 80%, tells us the frequency is still important to the reduction of drag.

Finally, there is Figure 5.9c, frequency versus change in drag. The third order best fit line in Figure 5.9c shows a promising dip in the coefficient of drag between $f_\omega = 0.5$ and $f_\omega = 0.6$. This is the kind of inflection points that indicates a higher order relationship worth exploring.

A single objective optimization study was also conducted with minimizing coefficient of drag as the only objective.

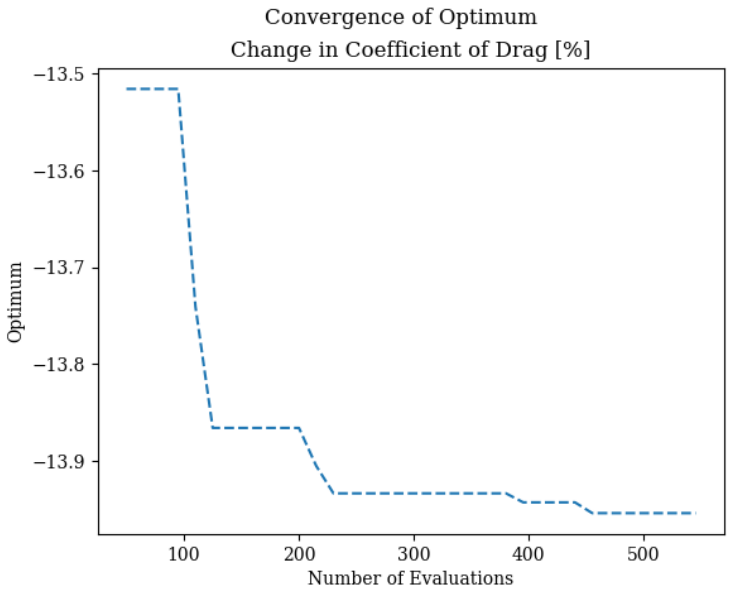


Figure 3.5: Oscillating Cylinder SOO Study - Convergence of optimum.

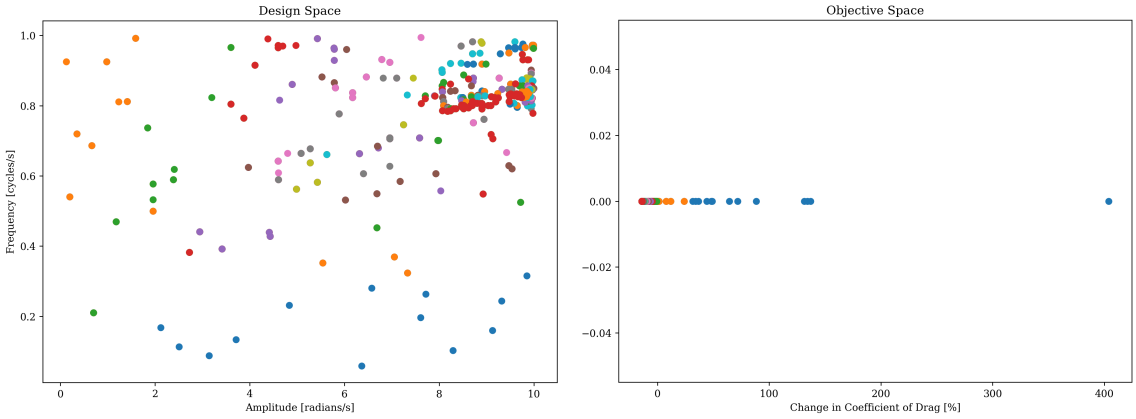


Figure 3.6: Oscillating Cylinder SOO Study - Entire Design and Objective Spaces - Generations 1 through 35

The results converge around the top right corner of the objective space because there is no energy consumption metric to establish a trade-off with. The goal here was to have results comparable to other oscillating cylinder single objective optimizations studies. However, the study conducted here uses a much larger parameter space com-

pared to previous studies. Therefore, the optimization algorithm converged around the highest energy cost solution, in the top right corner. This solution was far beyond the parameter spaces used in any previous studies found.

3.1.2 Post-Process

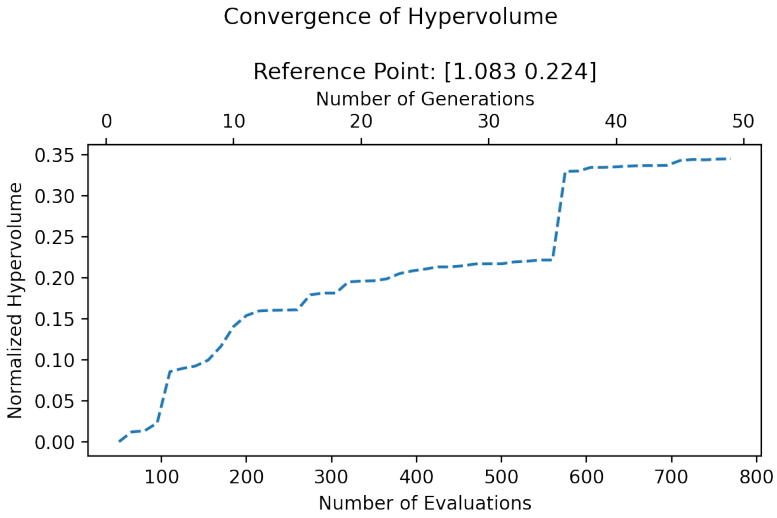


Figure 3.7: Oscillating Cylinder Optimization Study - Convergence of normalized hypervolume.

The optimization run shows good convergence at around 600 evaluations. With a population size of 50 and 15 offspring per generation:

$$N_{gen} = \frac{600 - 50}{15} + 1 = 37.67$$

Now this can be verified by plotting ranges of generations onto the parameter and objective spaces.

Below is the entire optimization study design and objective spaces.

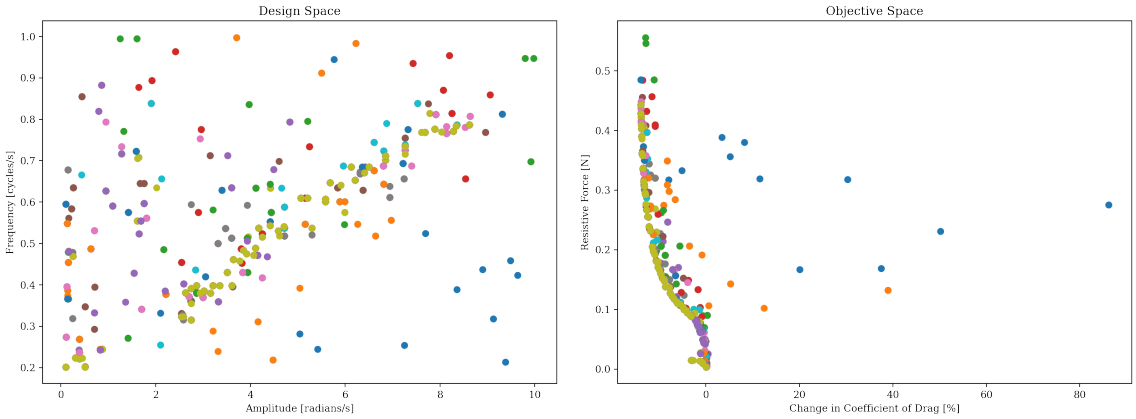
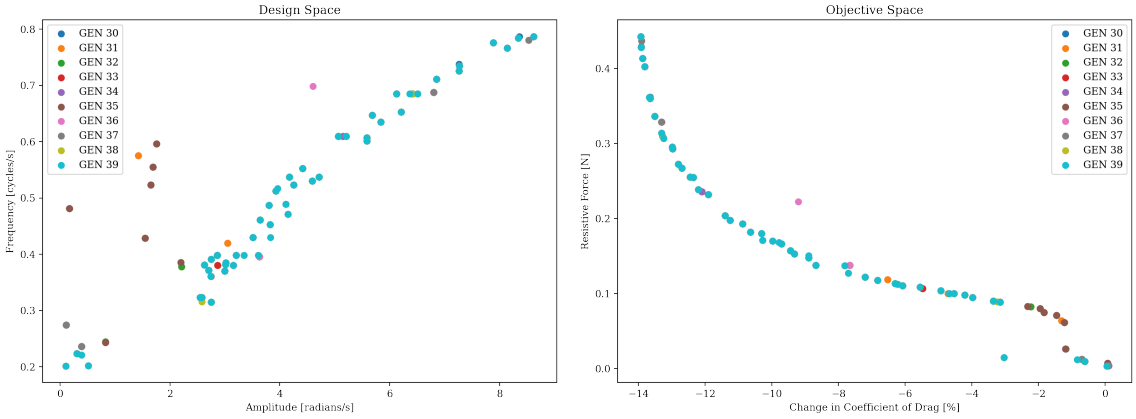


Figure 3.8: Oscillating Cylinder Optimization Study - Entire Design and Objective Spaces - Generations 1 through 50

In Figure 3.8, above, the graphs show the first generation produced the outliers which were discussed above. These outliers are discarded in the following generation.

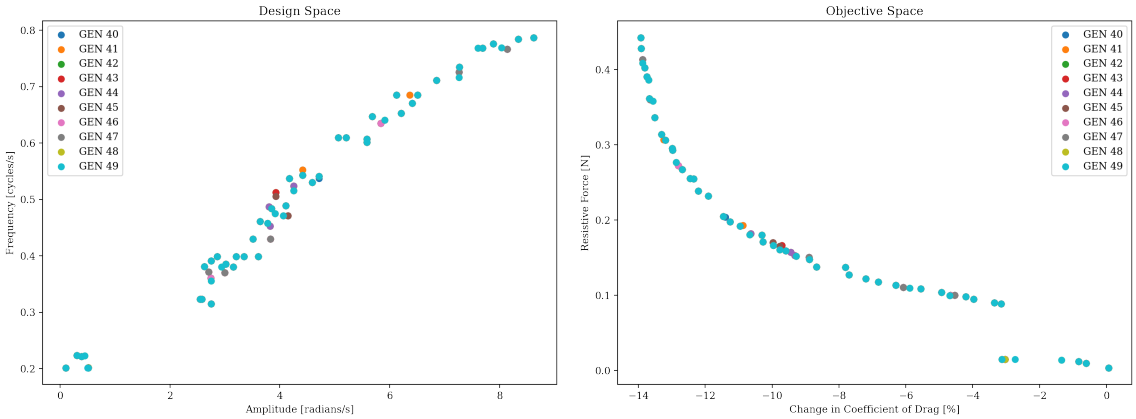
These figures show that the objective space converges around a Pareto front at about Generation 38. This is in agreement with the number of generations before convergence calculated using Figure 3.7, convergence of the hypervolume.



(a) *(b)*

Figure 3.9: Oscillating Cylinder Optimization Study - Generations 30 to 40 - Objective and Design Spaces.

Little progress was made in finding more optimum after Generation 38. This can be seen in the Figure 3.10, below, this can be seen.



(a) *(b)*

Figure 3.10: Oscillating Cylinder Optimization Study - Generations 40 to 50 - Objective and Design Spaces.

The final generation, generation 50, is mapped below.

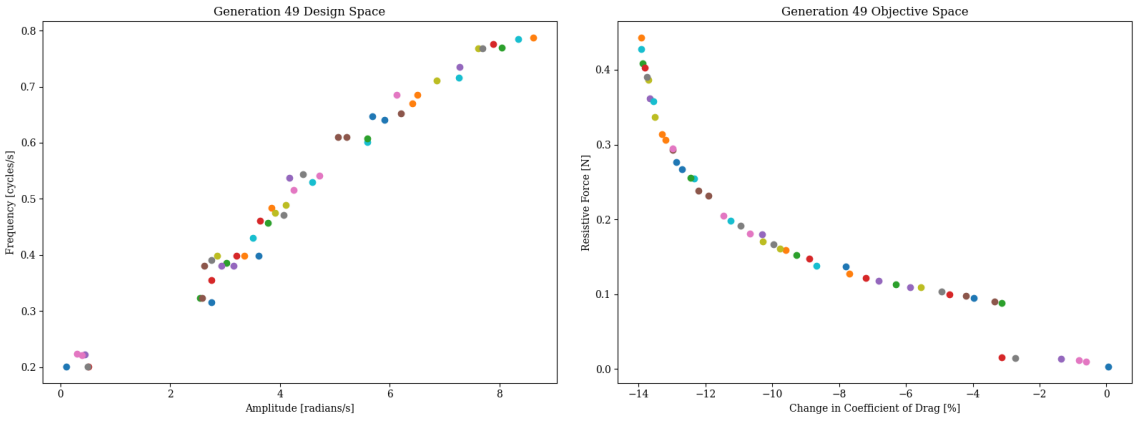
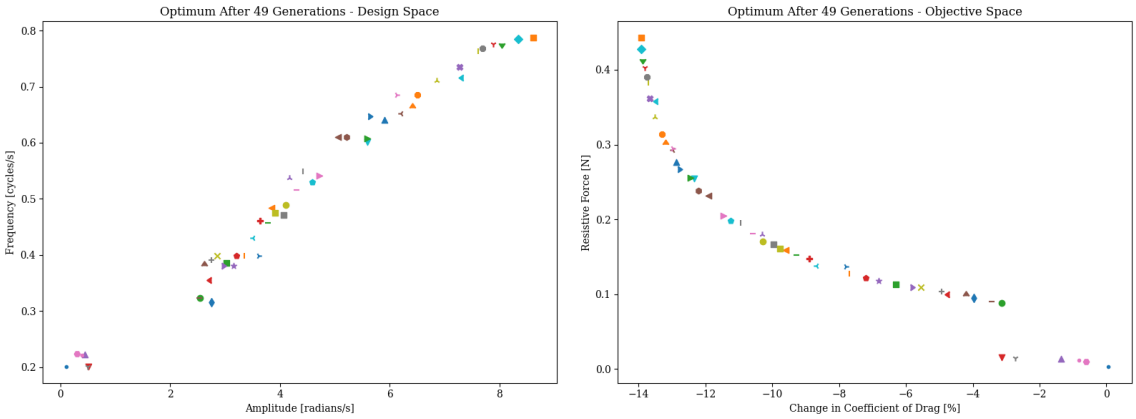


Figure 3.11: Oscillating Cylinder Optimization Study - Final Generation, Generation 50

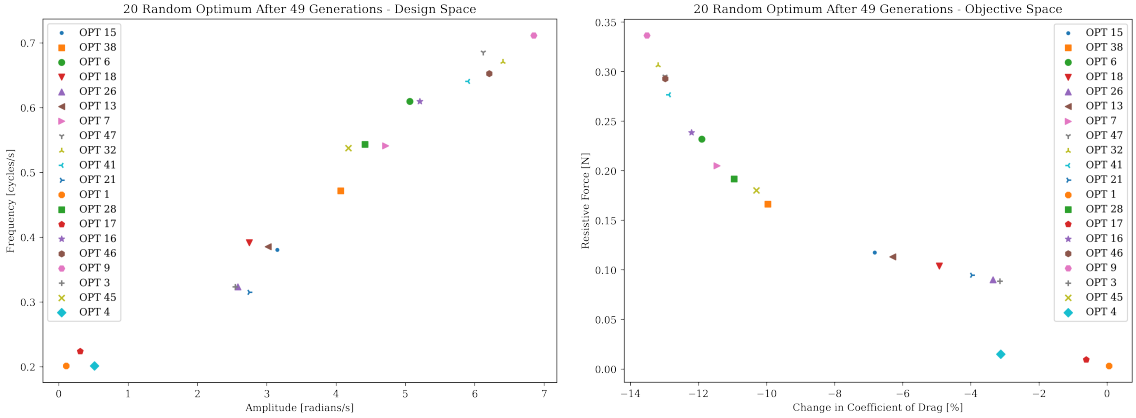
The optimum, or Pareto front, for this generation consists of 50 individuals. Since the population size is 50 the final generation is our Pareto front.



(a) *(b)*

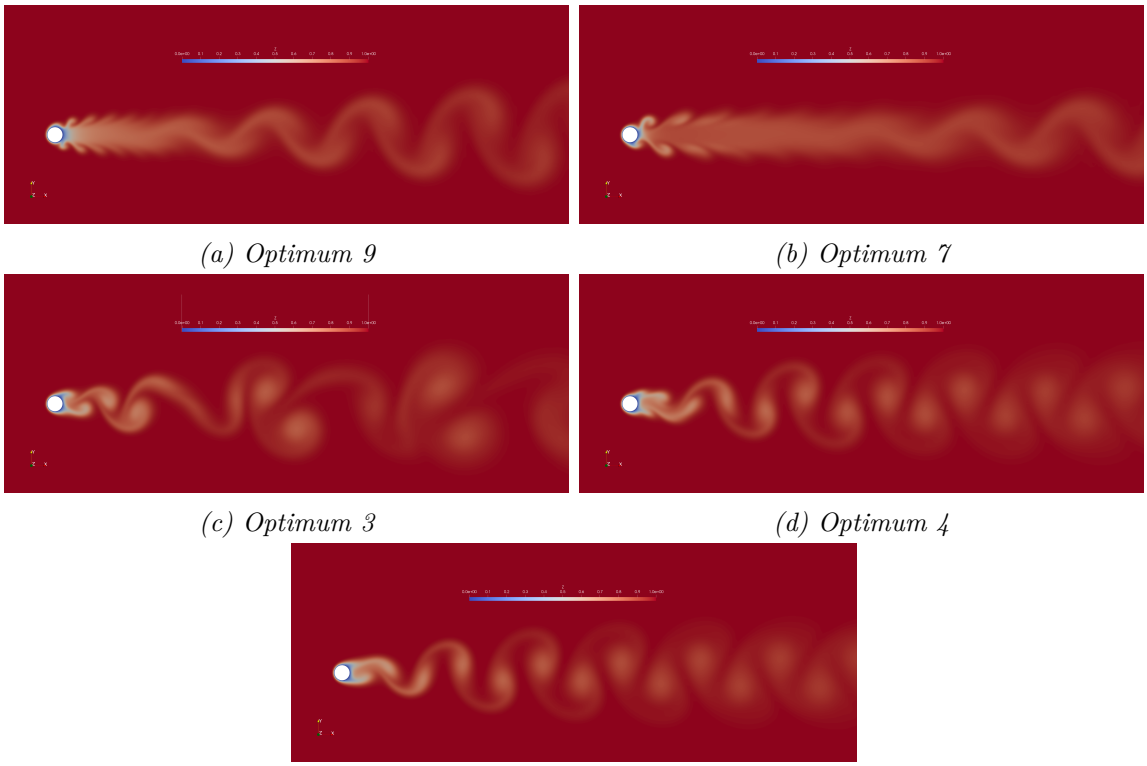
Figure 3.12: Oscillating Cylinder Optimization Study - Final Generation Optimum

For easy visualization, 20 optimum are selected at random and labeled. The mapping from design to objective space can now be clearly seen. In addition, the numbered labels correspond to files found in the *pareto_front* directory within each optimization run directory. A set of optimum from across the Pareto front is selected and the vortex shedding is visualized.



(a) *(b)*

Figure 3.13: Oscillating Cylinder Optimization Study - Final Generation 20 Optimum



(a) Optimum 9

(b) Optimum 7

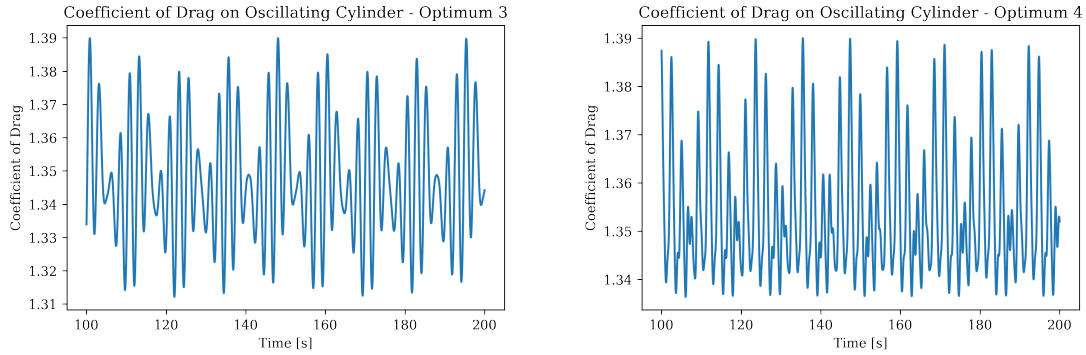
(c) Optimum 3

(d) Optimum 4

(e) No Oscillation

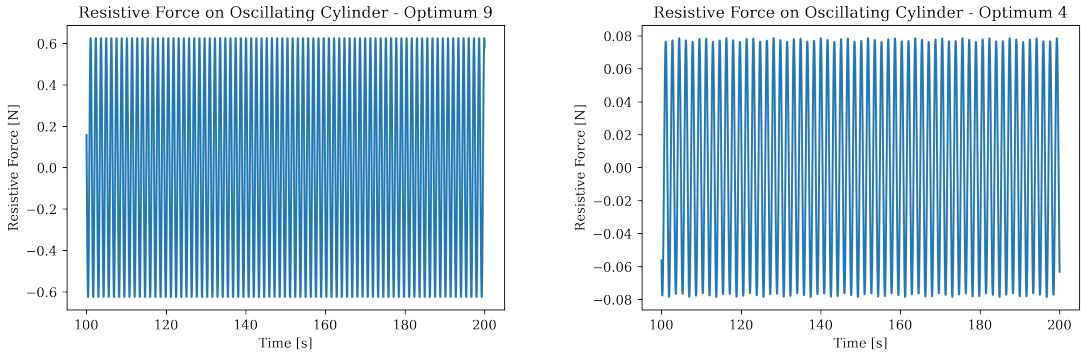
Figure 3.14: Oscillating Cylinder Optimization Study - Optimum After 50 Generations - Vortex Shedding. Z is a passive scalar.

The decision to evaluate the lower left corner of the design space, where the frequency and amplitude of the cylinder's oscillations approach 0, has resulted in a discontinuity in the Pareto front. This portion of the design space minimizes the resistive force, but is incapable of creating a significant reduction in drag.



(a) *(b)*

Figure 3.15: Oscillating Cylinder Optimization - Optimum 3 and 4 - Coefficient of drag over time.



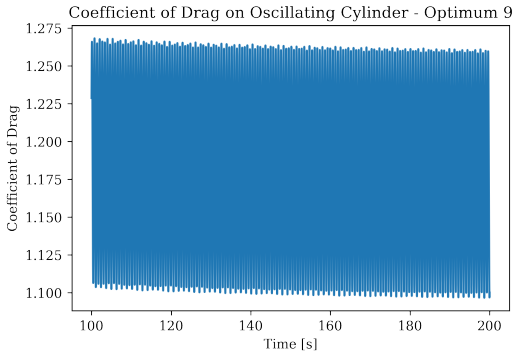
(a) *(b)*

Figure 3.16: Oscillating Cylinder Optimization - Optimum 3 and 4 - Resistive force over time.

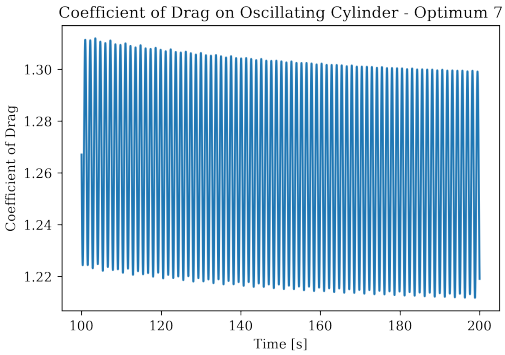
Optimum 3 and 4 were visualized because they are two cases on opposite sides of the Pareto front discontinuity. While the Pareto front with respect to the coefficient of drag is continuous, a discontinuity with respect to the resistive force exists. Figure 3.9, above, shows that this discontinuity was not discovered by the optimization algorithm until after Generation 35. Since the discontinuity occurs at the far end of the Pareto front, where the coefficient of drag reduction is <4%, the discontinuity is

of less consequence to the overall results of the study.

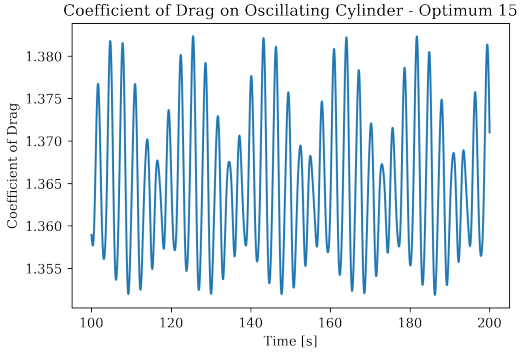
Some other Pareto front cases are shown below. They are ordered based on their location along the Pareto front left to right.



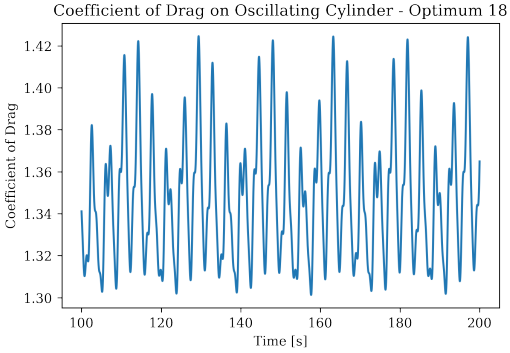
(a) Optimum 9



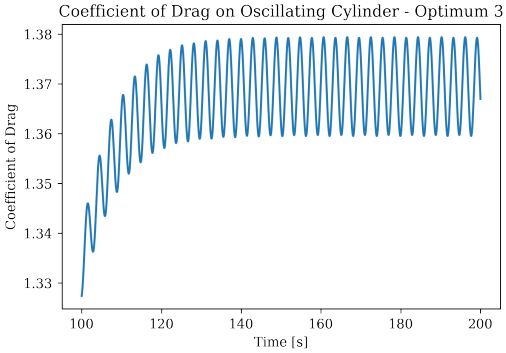
(b) Optimum 7



(c) Optimum 15



(d) Optimum 18



(e) No Oscillation

Figure 3.17: Optimum After 50 Generations - Vortex Shedding. Z is a passive scalar.

3.1.3 Follow-Up

An optimization study was also conducted at a Reynolds number of 500.

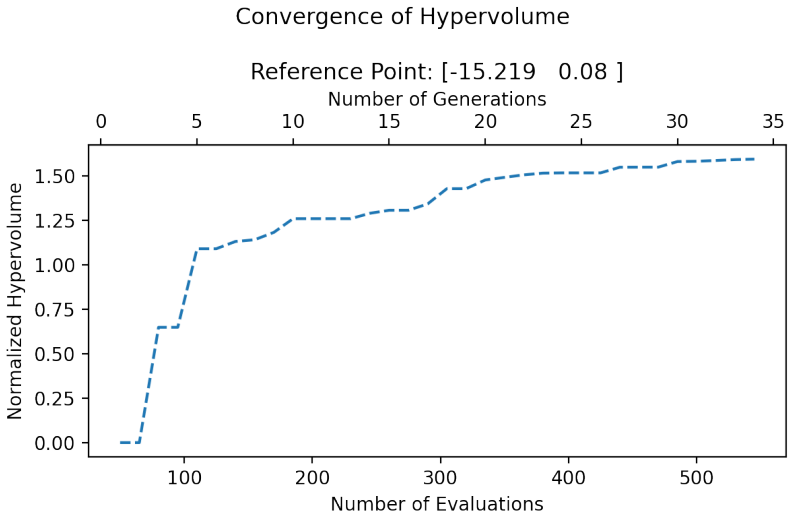


Figure 3.18: Oscillating Cylinder Optimization Study - Re=500 - Convergence of normalized hypervolume.

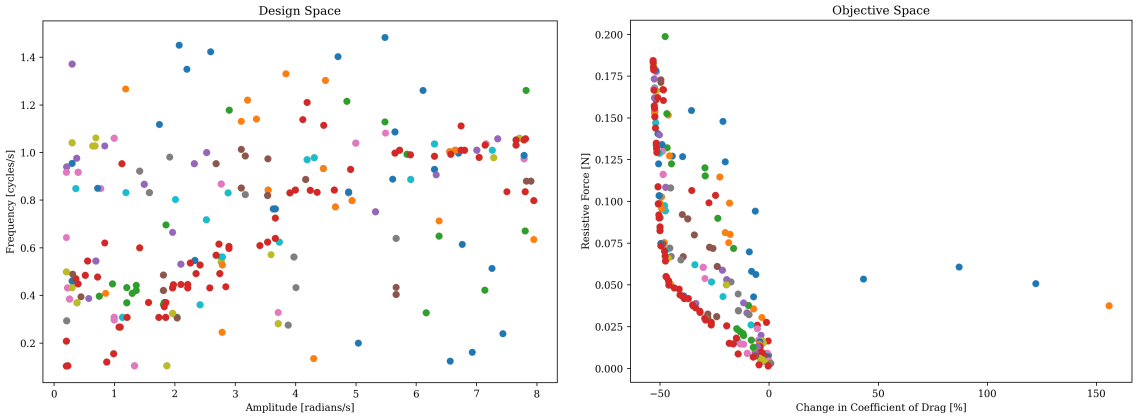
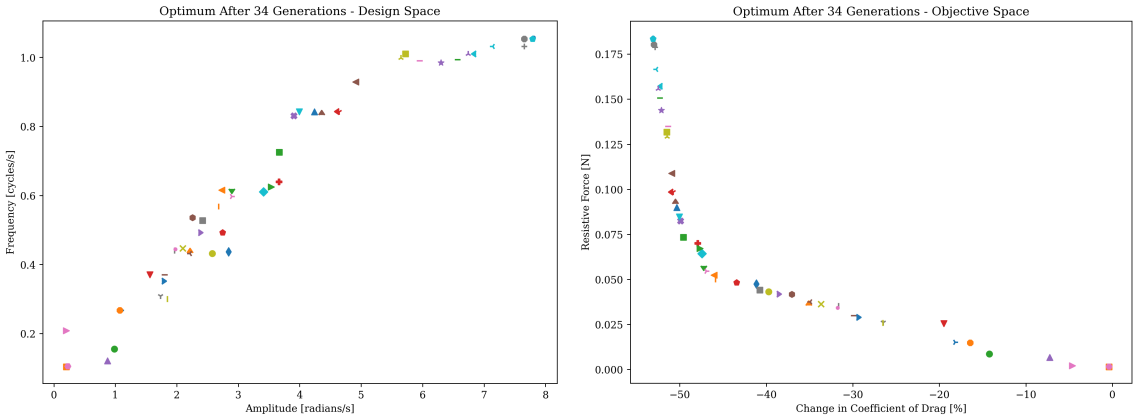


Figure 3.19: Oscillating Cylinder Optimization Study - Re=500 - Entire Design and Objective Spaces - Generations 1 through 35



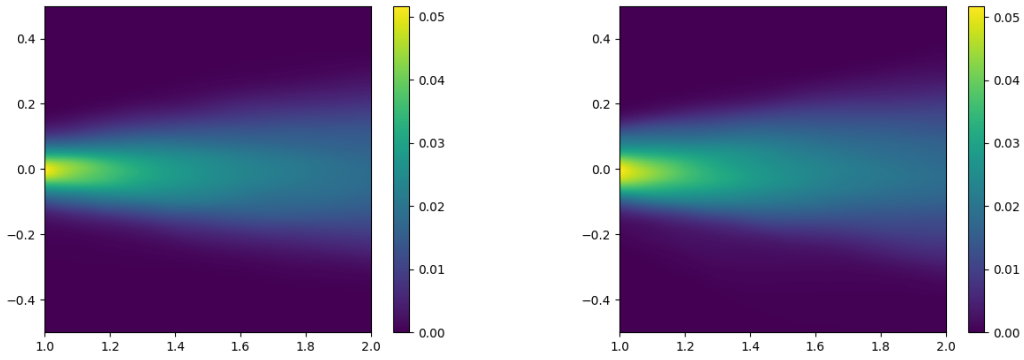
(a) *(b)*

Figure 3.20: Oscillating Cylinder Optimization Study - Re=500 - Final Generation Optimum

3.2 RANS Jet Simplification

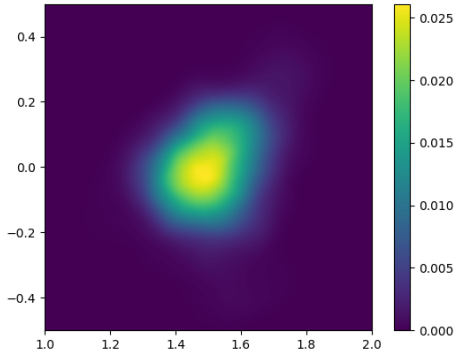
3.2.1 Pre-Process

The first step was to use linear interpolation to generate a 200x200x200 node 3D grid of the LES simulated free jet. Figure 5.11, below, shows the mid-planes of this 200x200x200 grid. These graphs, in particular Figure 3.21c, show that the mean velocity field is not perfectly axi-symmetric but is close.



(a) x-y slice at z=0

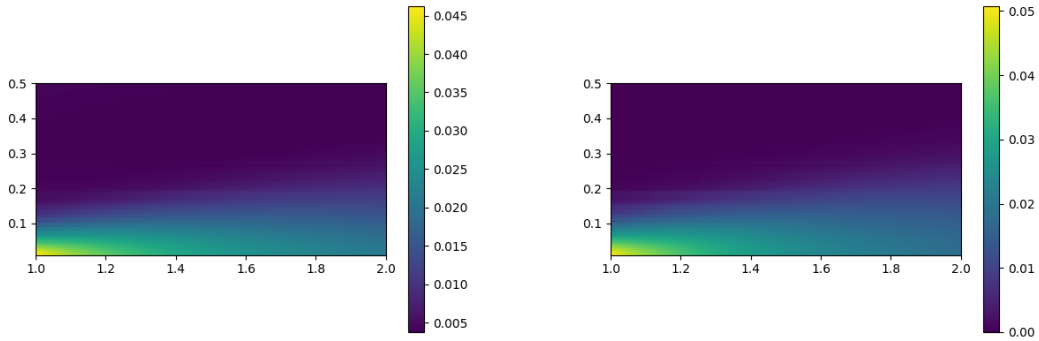
(b) x-z slice at y=0



(c) y-z slice at x=1.5m

Figure 3.21: RANS Jet Optimization Study - Flow field slices showing mass fraction of passive scalar, ϕ. YALES2 LES simulation of a 3D jet interpolated onto a 200x200x200 cell grid using python library scipy.interpolate.griddata() function.

Next, the radial averages was taken resulting the graphs shown below. Artifacts of the radial averaging can be seem in the graphs as faint horizontal lines. This is further evidence the LES free jet is not perfectly symmetrical.



(a) Velocity magnitude flow field. *(b) Mass fraction of passive scalar flow field.*

Figure 3.22: RANS Jet Optimization Study - YALES2 LES simulation of 3D jet reduced to a 2D radially averaged 200x100 cell grid.

The final interpolate takes place after every RANS simulation. The 2D axi-symmetric domain is interpolation onto the universal grid using a cubic interpolation scheme. This cubic interpolation scheme is only offered by *scipy.interpolation* for 2D grids. No radial averaging means no horizontal lines are seen in the grid diagram, unlike the LES universal grid.

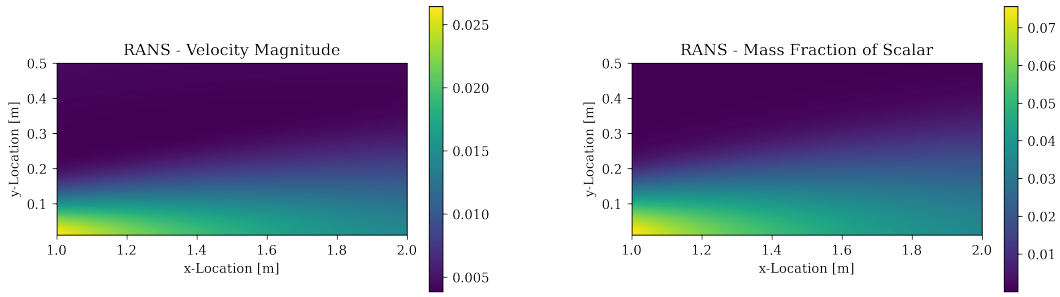


Figure 3.23: RANS Jet Optimization Study - RANS k-omega SST axi-symmetric jet flow field interpolated onto a "universal" grid.

The horizontal line, artifacts from the radial averaging, become most apparent

when the absolute value of the difference between the RANS and LES universal grids is taken. The results are shown below.

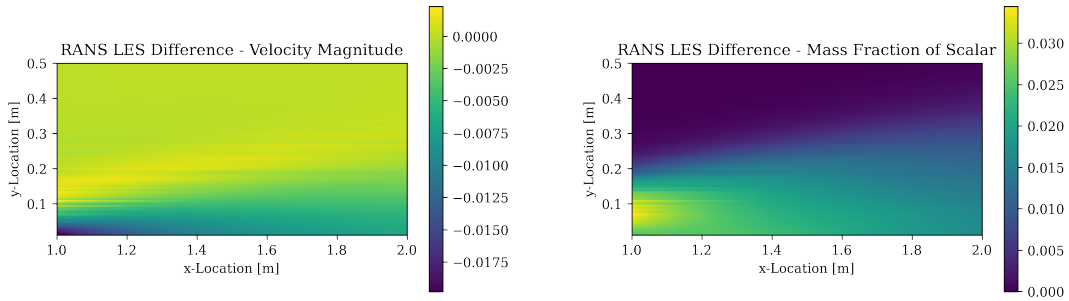


Figure 3.24: RANS Jet Optimization Study - Absolute value of the difference between the RANS k-omega SST axi-symmetric jet flow field and the radially averaged YALES2 LES 3D simulation flow field.

The far-field of the LES ZNMF jet has now been interpolated onto a universal grid and is ready to be compared with the RANS steady jet far-field. A preliminary case is run to obtain a design to objective space mapping reference. The reference point will be the most intuitive solution to the problem. To justify the use of an optimization study, the resulting optimum solution should be significantly better than the intuitive solution. We know the far-field of the ZNMF jet is dominated by the outward breath. Therefore, the most intuitive solution would be to use the ZNMF jet's outward breath peak velocity, $u_{out,peak} = 0.2\left[\frac{m}{s}\right]$, as the RANS steady jet velocity. The scalar concentration is kept at 100% mass fraction at the inlet and the mouth diameter is kept the same at $D_{mouth} = 0.02[m]$. The question we are asking is how close can a RANS steady jet get to predicting the far-field of a LES ZNMF jet when all other boundary conditions are kept the same? The comparison between passive scalar momentum flux along the axis of symmetry in the LES versus

the RANS simulation is show below.

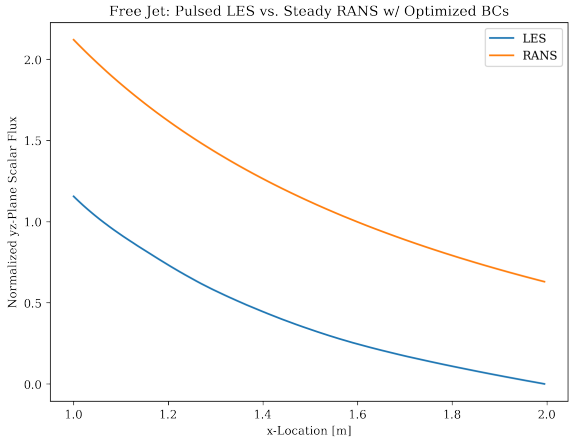


Figure 3.25: RANS Jet Optimization Study - Comparing the 3D LES pulsatile free jet simulation to the simplified 2D RANS axi-symmetric steady free jet with $u_{breath} = 0.2\,[\frac{m}{s}]$ *and* $D_{mouth} = 0.02[m]$.

The first generation of cases indicates that certain regions of the parameter space show very little promise. In addition, there are some regions of the objective space that cannot be mapped to with the limitations of the parameter space. More precise relationships between parameters and objectives will be apparent after mapping generation 1.

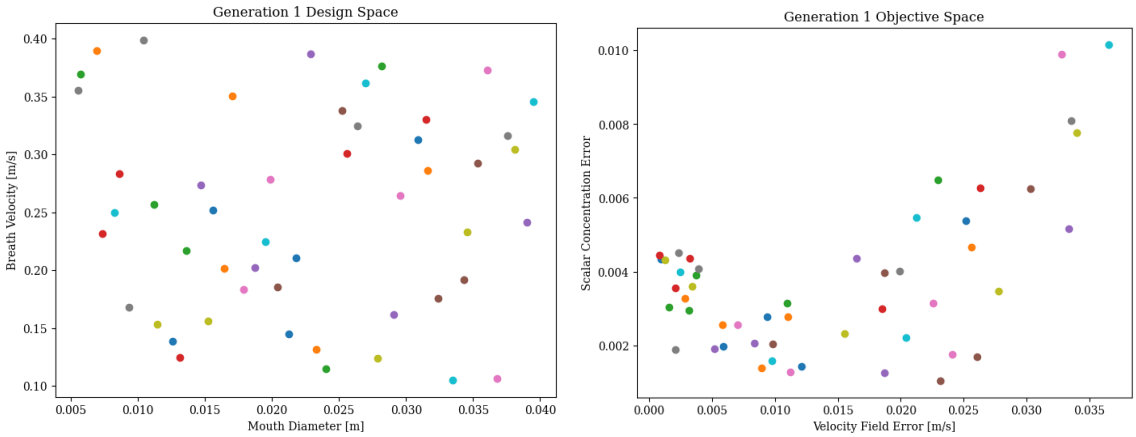


Figure 3.26: RANS Jet Optimization Study - Generation 1

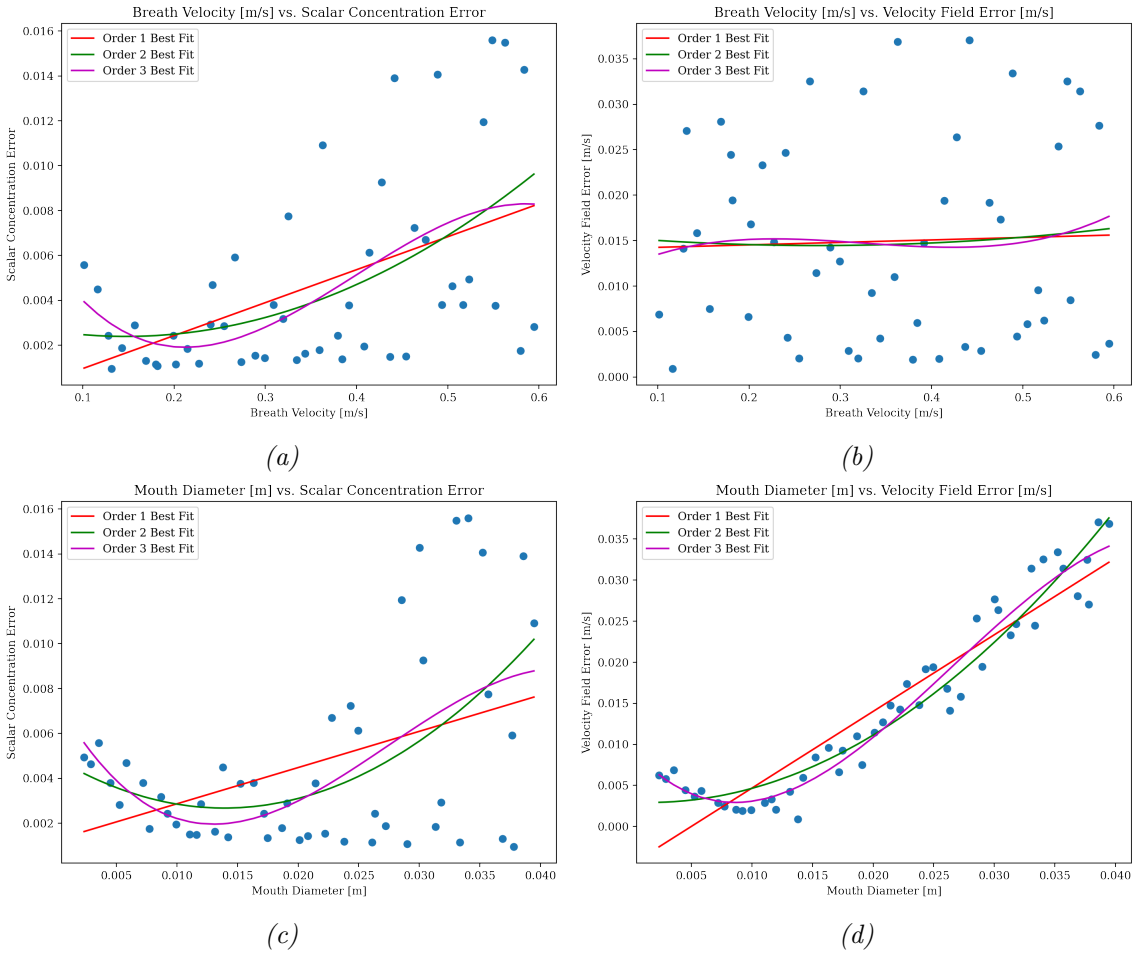


Figure 3.27: Parameters versus objectives for generation 1; which is a Latin hypercube sampling of the entire parameter space. First through third order best fit lines plotted as well.

The first parameter, breath velocity, has a direct relationship with scalar concentration error. The third order best fit line indicates that the optimum solution will exist around 0.2, which is the peak velocity of the LES ZNMF jet and the velocity used for our intuitive RANS steady jet simulation.

No correlation is found between the breath velocity and the velocity field error. This was not expected. The assumption was that there would be a strong correlation

between breath velocity and velocity field error. However, within the range of breath velocities explored by the parameter space and while the mouth diameter is also being varied, the breath velocity cannot be correlated to the velocity field error.

The mouth diameter has a strong correlation to both scalar concentration and velocity field error. This indicates that mouth diameter is the dominant parameter. A dominant parameter is a parameter that has a greater influence on the objectives than any other parameter. The second and third order best fit lines indicate that a concentration of the optimal solutions will exist around a mouth diameter of about $0.15[m]$. Though cases minimizing scalar concentration error exist even at $D_{mouth} = 0.37[m]$ meaning that more isolated solutions could exist at large mouth diameters.

Finally, there is the mouth diameters correlation to the velocity field error. This is the strongest correlation of the parameter objective relationships. The graph indicates an optimal set of solutions will be found around $D_{mouth} = 0.01[m]$. After $D_{mouth} = 0.01[m]$ the mouth diameter has a strong direct correlation to velocity field error. No minimal solutions exist at larger mouth diameters. Since breath velocity has little to no correlation to velocity field error, it makes sense that mouth diameter is so closely correlated with velocity field error.

3.2.2 Post-Process

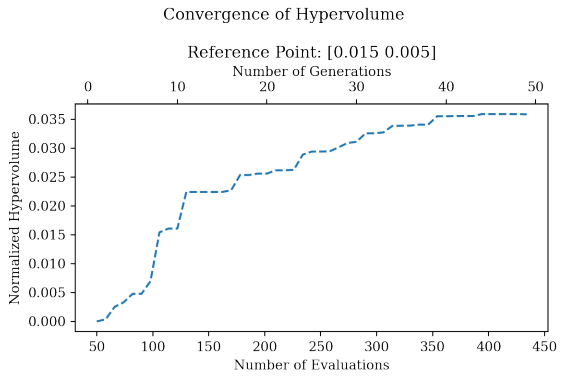


Figure 3.28: RANS Jet Optimization Study - Convergence of normalized hypervolume.

The hypervolume shows good convergence at around 350 evaluations, or at around Generation 38.

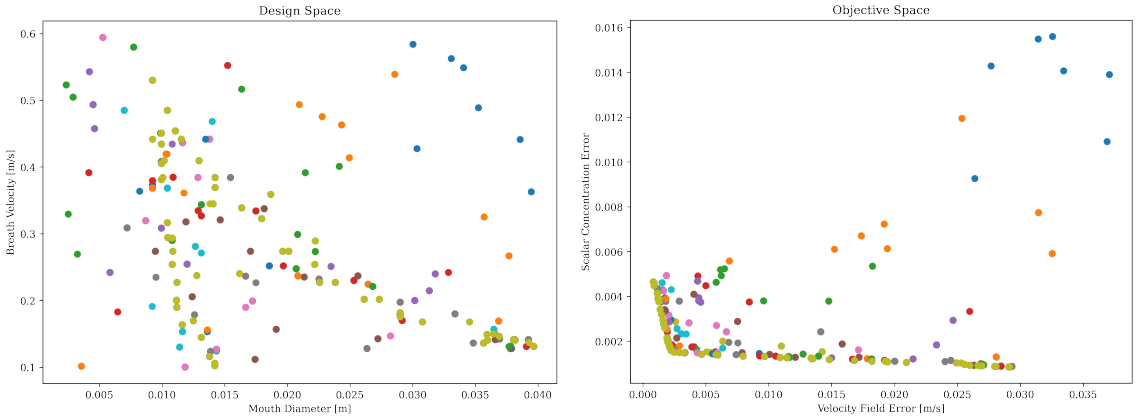
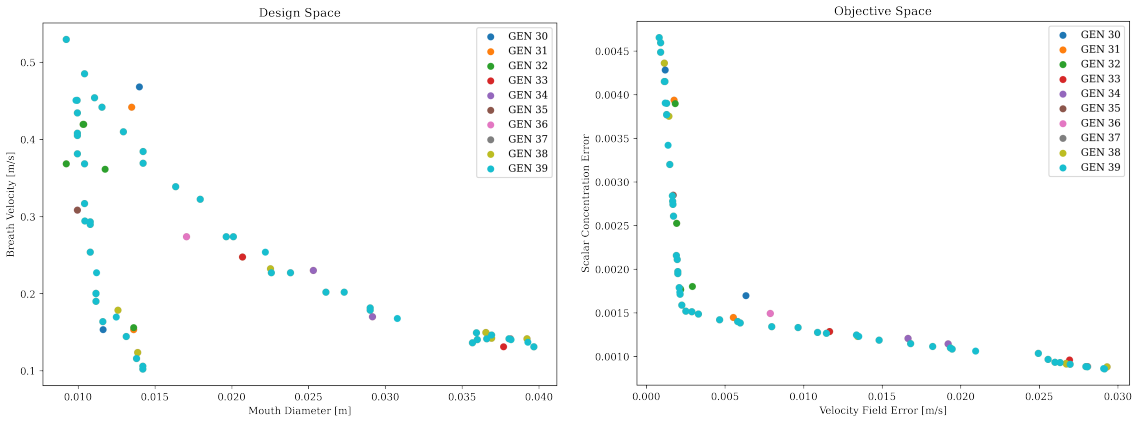


Figure 3.29: RANS Jet Optimization Study - Entire Design and Objective Spaces - Generations 1 through 50

In Figure 3.29, above, the graphs show the first generation produced the outliers which were discussed above. These outliers are discarded in the following generation.

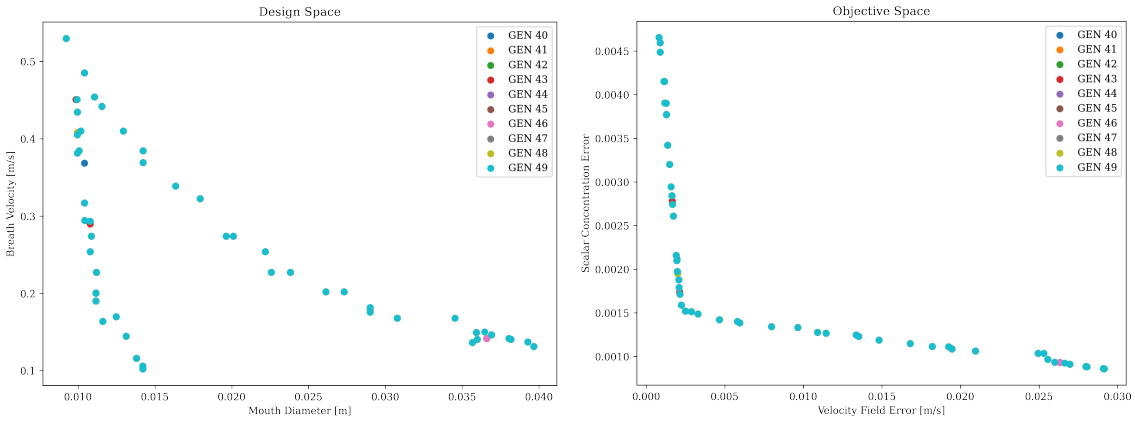
The following figures show convergence at around Generation 38 which is in agreement with the convergence of hypervolume seen in Figure 3.28.



(a) *(b)*

Figure 3.30: RANS Jet Optimization Study - Generations 30 to 40 - Objective and Design Spaces.

Little progress was made in finding more optimum after Generation 37. This can be seen in Figure 3.34, below.



(a) *(b)*

Figure 3.31: RANS Jet Optimization Study - Generations 40 to 50 - Objective and Design Spaces.

Good convergence of this optimization algorithm means that the final generation and final optimum are the same population. The case which uses a breath velocity equivalent to the peak breath velocity of the LES ZNMF case, i.e. $u_{const.jet} = u_{peak.ZNMF.jet}$, is also plotted alongside the optimum.

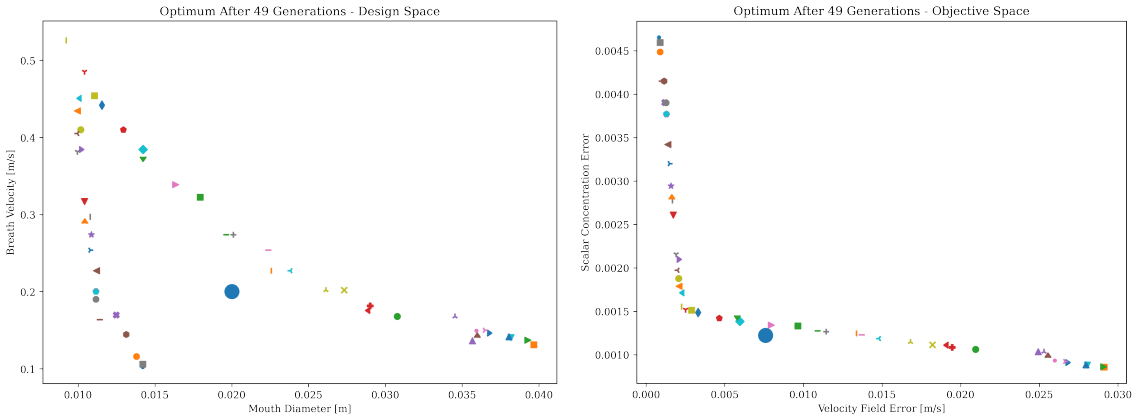


Figure 3.32: RANS Jet Optimization Study - Design and Objective Space Pareto Optimum. The large blue dot indicates RANS jet with constant velocity equivalent to peak velocity of ZNMF jet and all other boundary conditions the same, $u_{jet} = 0.2\left[\frac{m}{s}\right]$ *and* $D_{mouth} = 0.02[m]$.

This shows that the optimization algorithm seems to have missed a Pareto front solution. This is likely an isolated solution which is not part of the overall pattern found by the optimization algorithm. The optimization algorithm's inability to find this solution highlights the shortcomings of this process. Discovering isolated solutions is not a strength of genetic algorithms. The evolutionary framework used by genetic algorithms favors finding groups, or families, of solutions. In most CFD optimization problems, this is beneficial. Often the goal is to discover a predictable relationship within a design space that can be leveraged to optimize the design. However, when working with such highly-dimensional problems, the likelihood that isolated solutions exist should always be considered.

To avoid missing isolated solutions, population size can be increased. This makes for an extremely long first generation run time. The advantage is a more thorough sampling of the entire parameter space, making it more likely the optimization algorithm will find any isolated solutions.

A random selection of 20 of the final generation's optimum are plotted below.

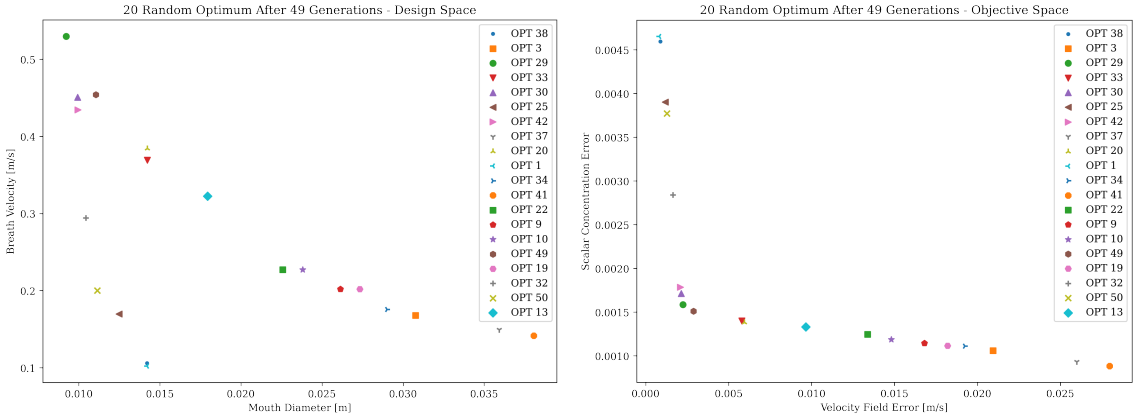
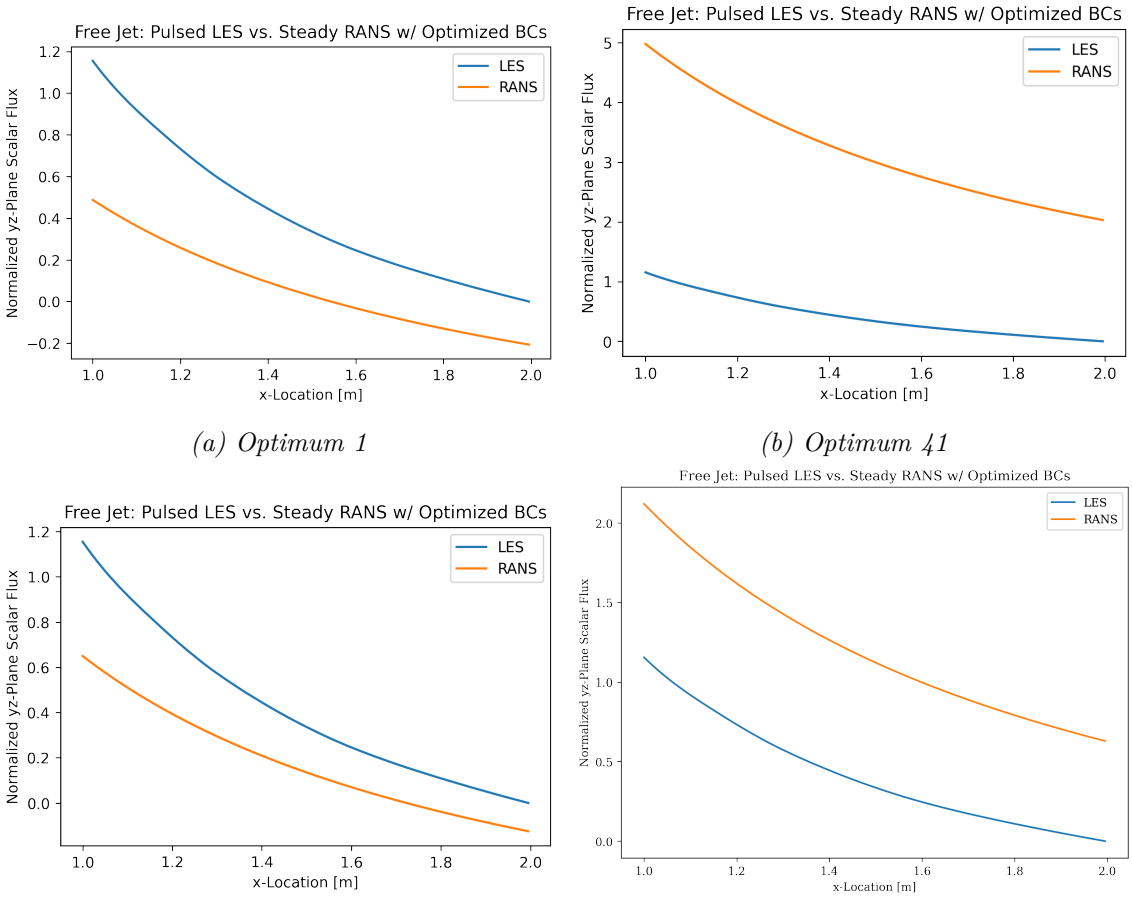


Figure 3.33: RANS Jet Optimization Study - Final Generation, Generation 50 - 20 Optimum

Scalar momentum flux graphs of optimum solutions from across the Pareto front can be seen below.



(a) Optimum 1

(b) Optimum 41

(c) Optimum 29

(d) $u_{const-jet} = u_{peak-ZNMF-jet}$

Figure 3.34: RANS Jet Optimization Study - Pareto front examples.

This shows the trade-off between over predicting and underpredicting momentum flux which exists along the Pareto front.

3.2.3 Follow-Up

A follow-up study was conducted where another parameter was added. This third parameter is the mass fraction of scalar injected into the domain at the jet inlet.

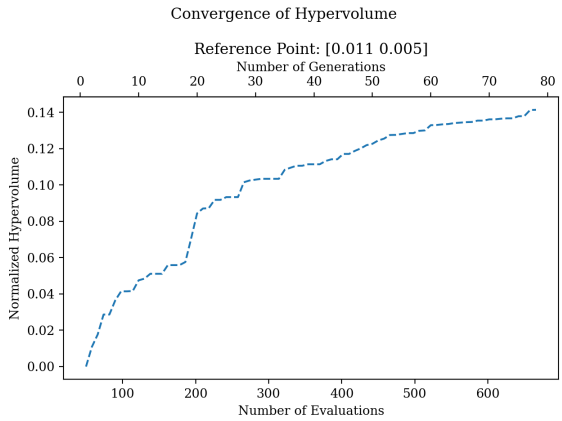


Figure 3.35: Convergence of normalized hypervolume.

The hypervolume shows difficulty converging. This is most likely due to the large parameter space. While decent convergence seems to be achieved after about 520 evaluations or about 60 generations, it is apparent from the jump in hypervolume, at about Generation 78, that full convergence has not been achieved.

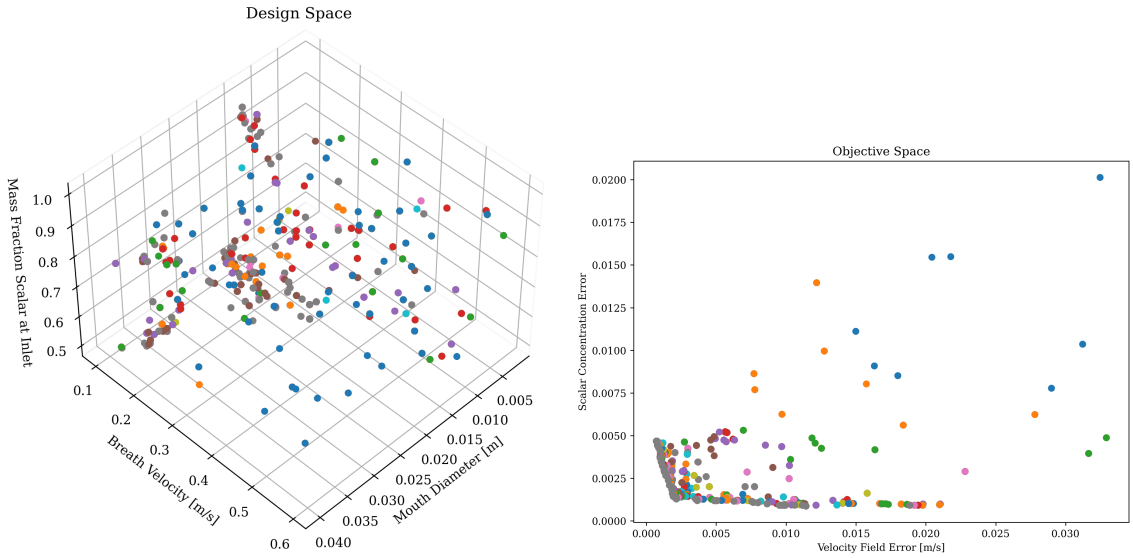
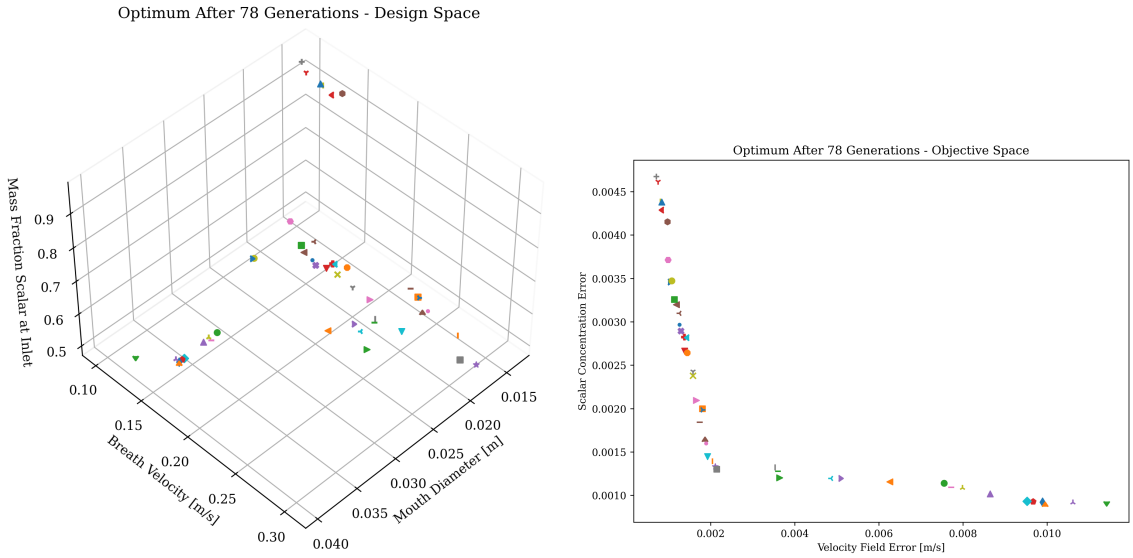


Figure 3.36: RANS Jet Optimization Study - Entire Design and Objective Spaces - Generations 1 through 79



(a) *(b)*

Figure 3.37: RANS Jet Optimization Study - RANS Jet 3 Parameter Optimization Study - Final Generation Optimum

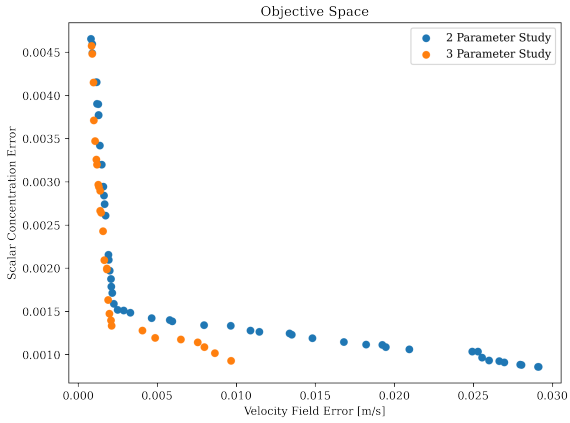
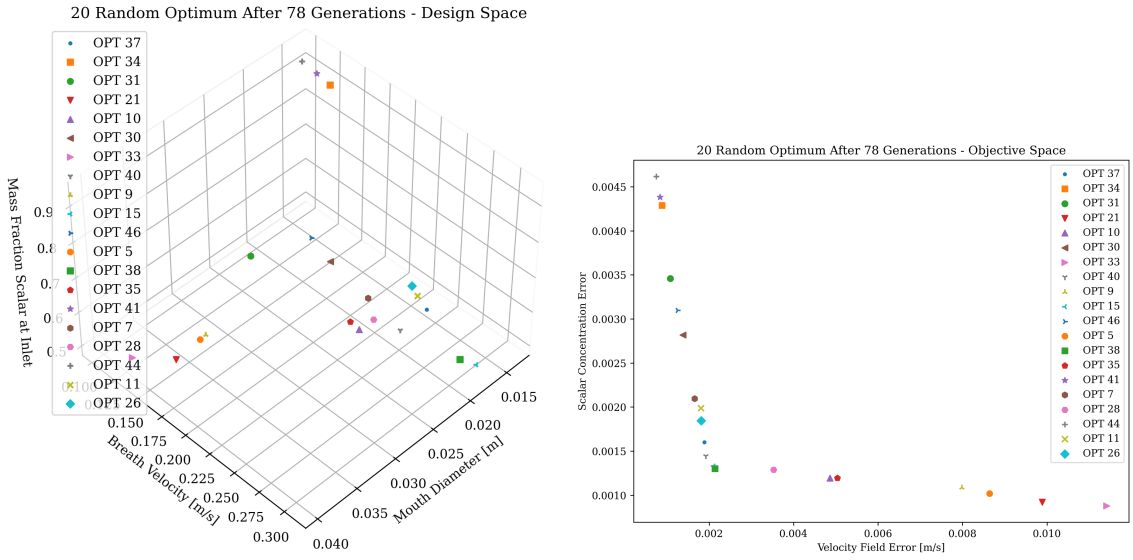


Figure 3.38: RANS Jet Optimization Study - Comparison between 2 parameter study optimum and 3 parameter study optimum.

The results show that the addition of the mass fraction of scalar at the inlet as a parameter helped to achieve better optimization of the scalar concentration field. However, the larger parameter space took about 200 more evaluations to converge,

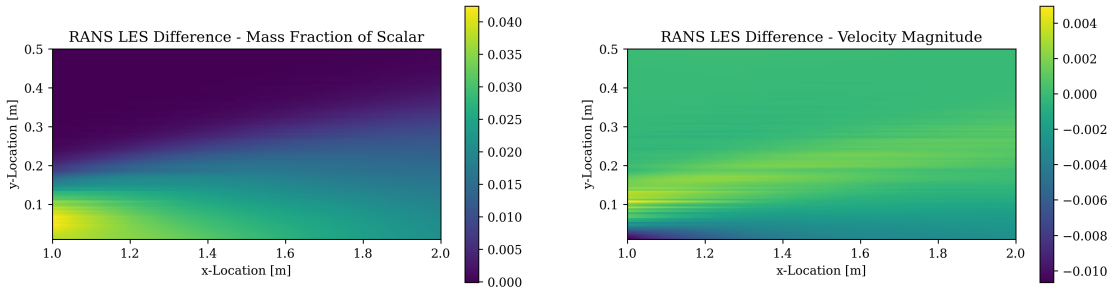
and even then it is hard to say good convergence was fully realized.



(a) *(b)*

Figure 3.39: RANS Jet 3 Parameter Optimization Study - Final Generation Optimum

The critical point, Optimum 38, has a scalar concentration error slightly below the critical optimum found in the two parameter optimization study.



(a) *(b)*

Figure 3.40: RANS Jet 3 Parameter Optimization Study - Optimum 38

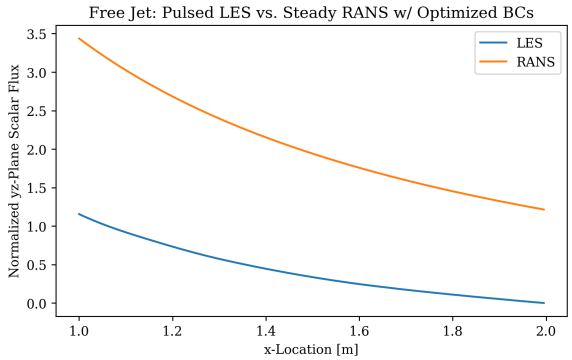


Figure 3.41: RANS Jet 3 Parameter Optimization Study - Optimum 38 - RANS versus LES flux normalized.

The reduction in scalar concentration field error is accomplished by changing the mass fraction of scalar at the inlet from 1.0 to 0.517.

3.3 Air Purifier Configuration

3.3.1 Pre-Process

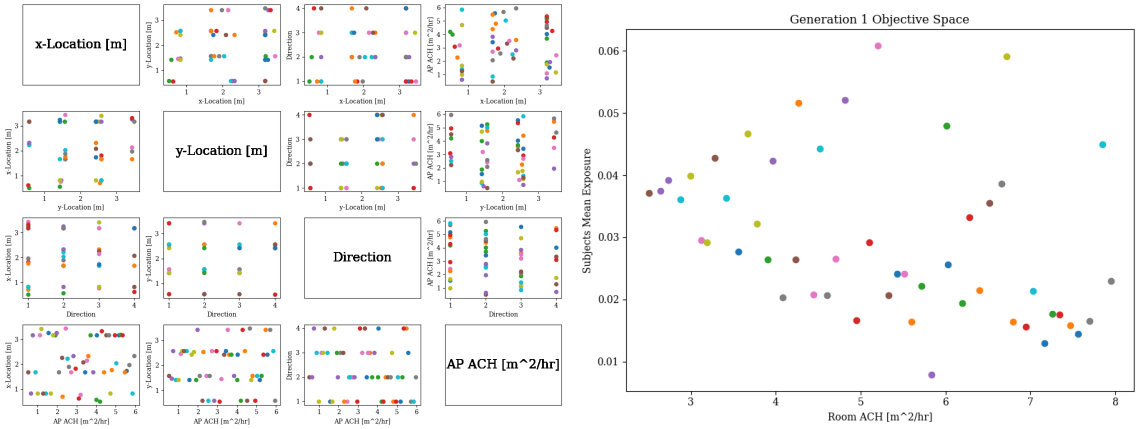
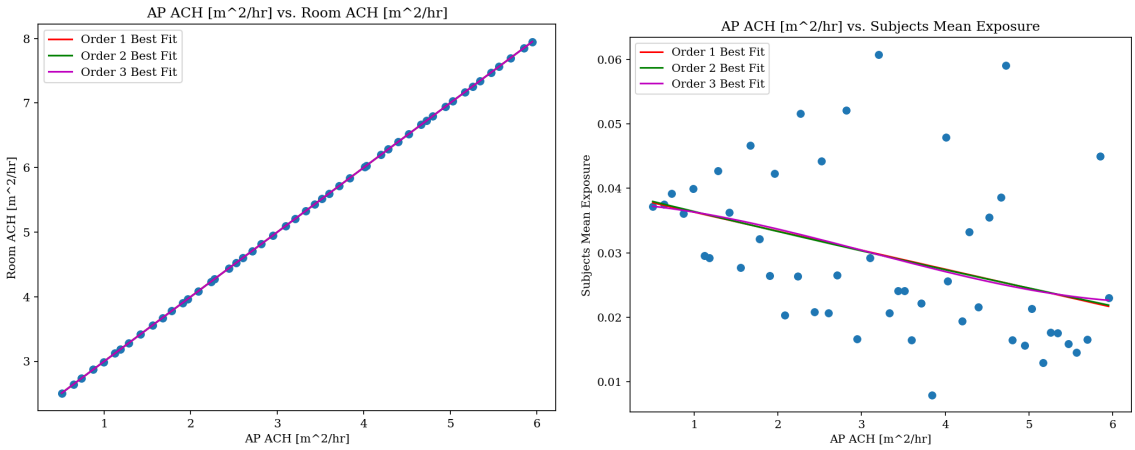


Figure 3.42: Air Purifier Configuration Optimization Study - First Generation - Design and Objective Spaces.

The use of a high dimensional parameter space makes interpretation of the parameter to objective space mapping more difficult. However, some trends can still be detected.

First, let's look at the air purifier ACH. The air purifier ACH is directly related to the room ACH because the room ACH is simply $ACH_{room} = ACH_{AP} + 2$. The room ACH is in some sense a "mock" objective, since we know a higher ACH means lower exposure already. The room ACH objective serves to establish a Pareto front where at every ACH level the optimal air purifier configuration is found. So it is no surprise that the air purifier ACH is inversely correlated with the subjects' mean exposure.



(a) *(b)*

Figure 3.43: Air Purifier Configuration Optimization Study - Mapping generation 1.

Looking at Figure 3.44b, the x-location is correlated to the subjects' mean exposure. The lowest subjects' exposures are correlated strongly with x-location in the center of the room. Lower subject exposure is also correlated with locations closer to the wall with the room outlet more so than x-location close to the wall with the room inlet. Direction and y-location have no strong correlation with the subject exposure.

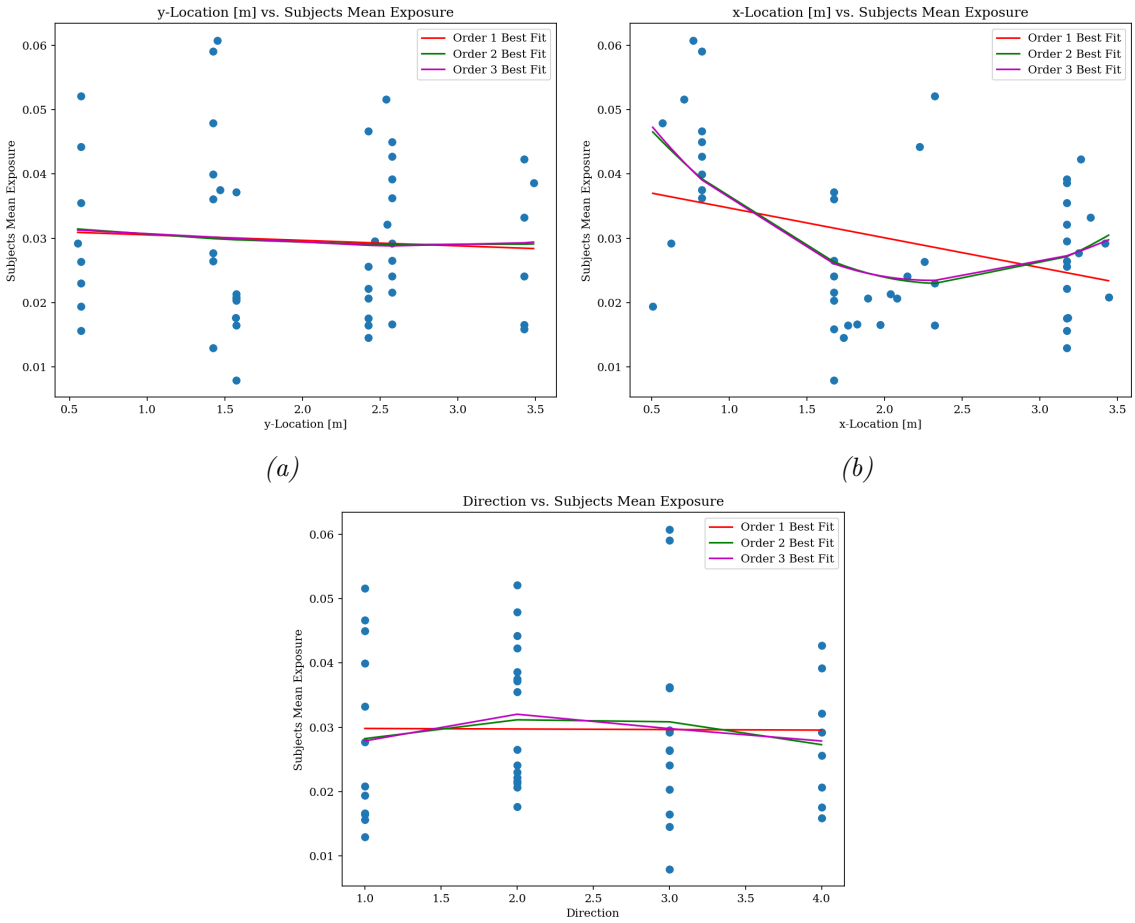


Figure 3.44: Air Purifier Configuration Optimization Study - Mapping generation 1.

3.3.2 Post-Process

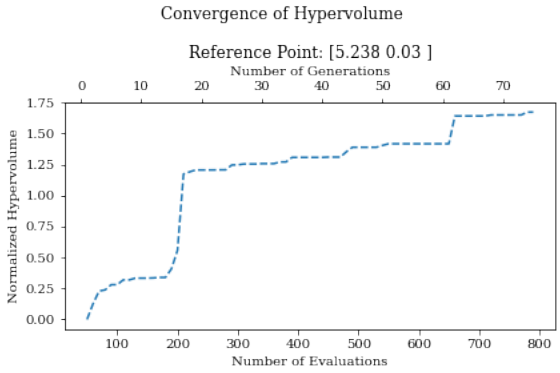


Figure 3.45: Air Purifier Configuration Optimization Study - Convergence of the hypervolume measured from the reference point to the Pareto front.

Convergence of the hypervolume is achieved at around 675 evaluations, or after about 61 generations. After about 200 evaluations, the optimization algorithm is only able to make an incremental step until at about 650 evaluations another jump to better Pareto front convergence is made. This makes it hard to say for certain if the algorithm is fully converged; but after 800 evaluations, the turn around time and computational resource are reaching impractical levels.

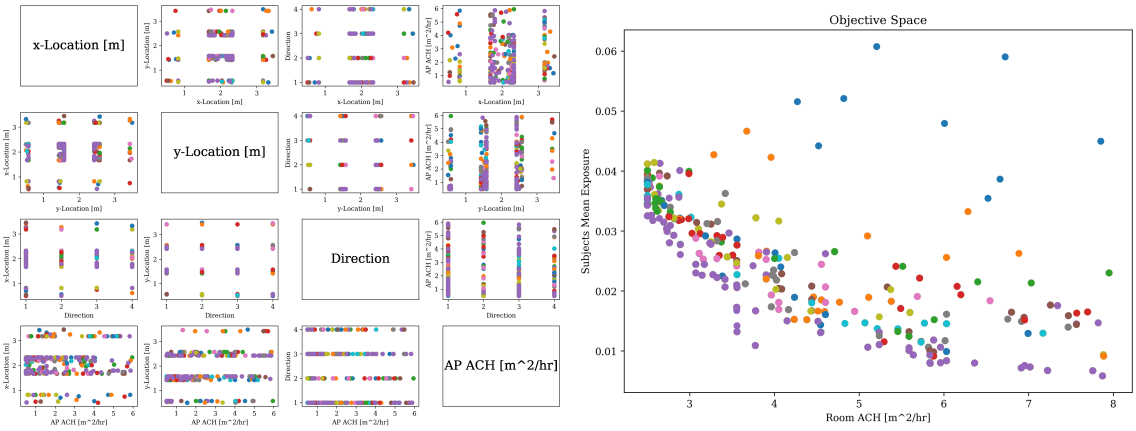


Figure 3.46: Generations 1 through 75 design and objective spaces.

The progress made in the first 20 generations is shown below. The discovery of the individual isolated in the bottom left of the objective space is the cause of the spike seen in the hypervolume convergence graph, Figure 3.45.

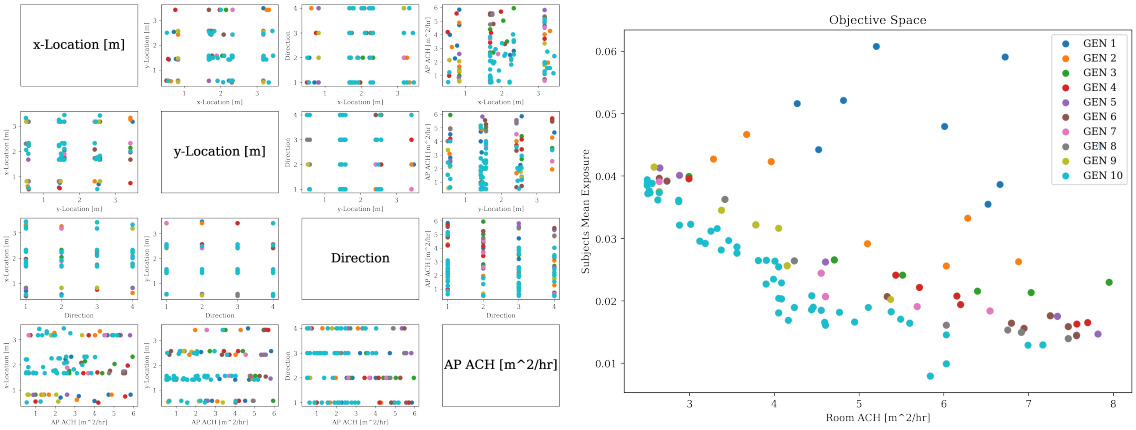


Figure 3.47: Generations 1 through 10 design and objective spaces.

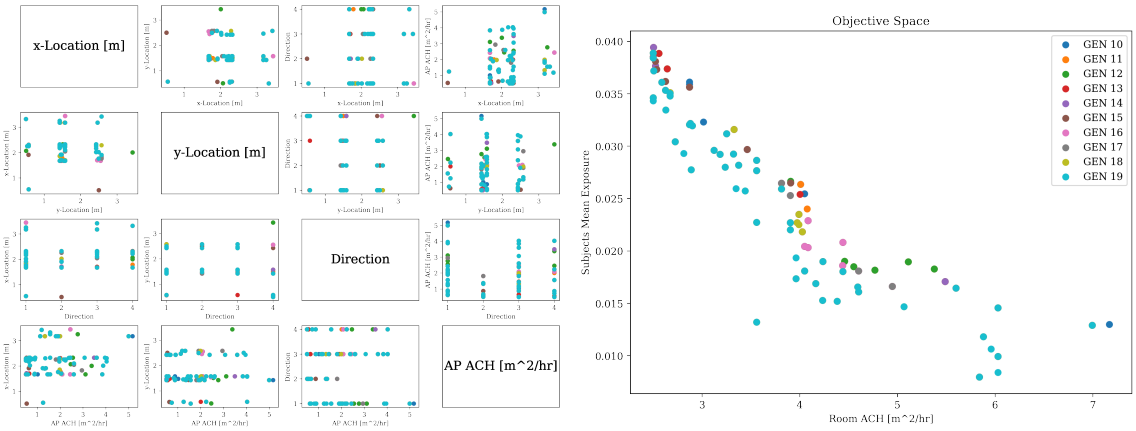


Figure 3.48: Air Purifier Configuration Optimization Study - Generations 10 through 20 design and objective spaces.

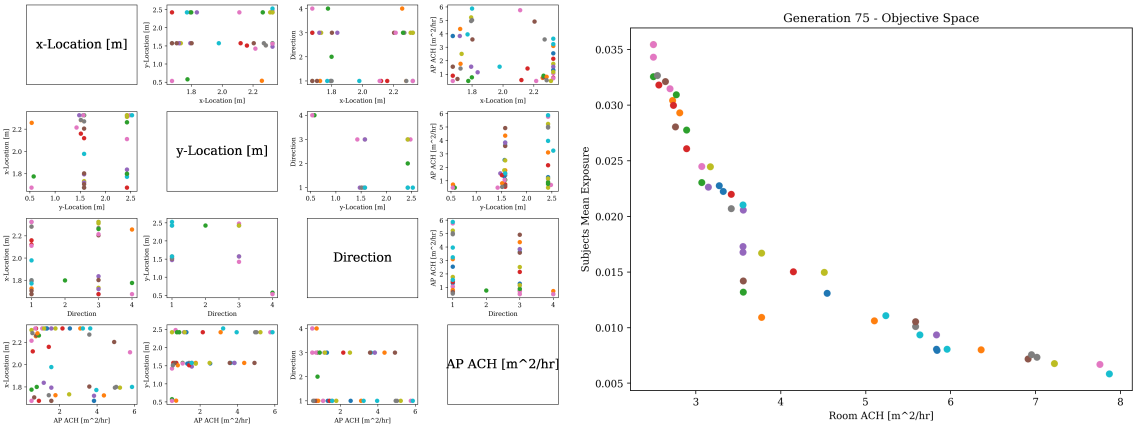


Figure 3.49: Air Purifier Configuration Optimization Study - Final Generation - Generation 75

The final generation is not the same as the optimum. This indicates that complete convergence has not been achieved. One reason the optimization algorithm may have a problem converging is a flaw in the methodology. Using the custom *Repair* subclass has likely made convergence more difficult. When a new population is created, through selection, crossover and mutation, this population is then "repaired". The custom *Repair* subclass changes the individuals in the population before they

are evaluated. This means the family of solutions the algorithm is trying to converge around could be getting constantly adjusted by the custom *Repair* subclass. In other words, the best set of solutions may be invalid, but, the algorithm does not know this and still tries to generate these possible solutions for evaluation. These sets of solutions are then manipulated by the custom *Repair* class before they are ever evaluated. Other possible methods are discussed in the **Conclusion** section.

Next, the optimum after the final generation are compared with the intuitive solution of placing the air purifier at the center of the room. The results are locations along the Pareto front. This does not bode well for justifying the use of an optimization study. However, in some respects, this is simply a preliminary study where the optimal set of solutions, the Pareto front, is known. One goal is to see if the optimization algorithm is capable of finding this solution, and if so, how long would it take.

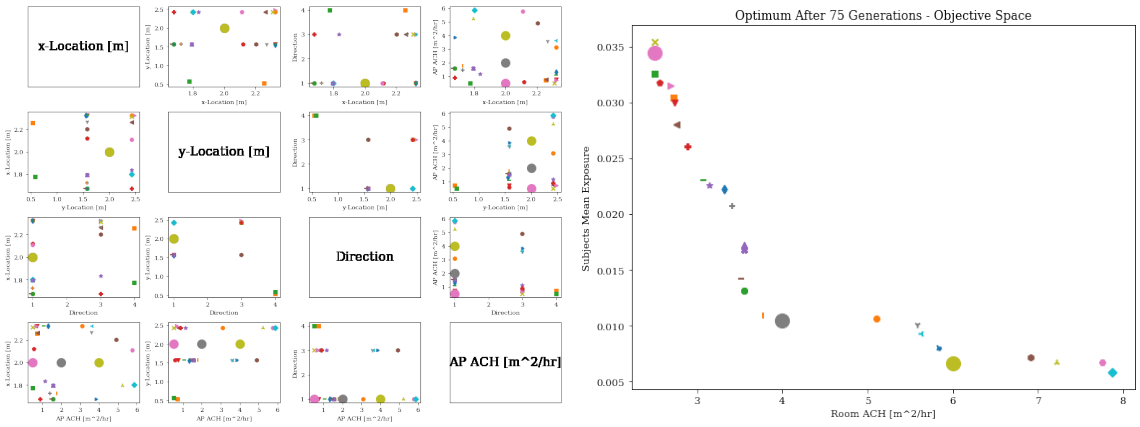


Figure 3.50: Air Purifier Configuration Optimization Study - Air purifier located in middle of room blowing in the positive y-direction with an ACH of 4, 2, and 0.5. These map to room ACHs of 6, 4, and 2.5.

Here, the optimum are labeled so that we can examine cases from a across the Pareto front.

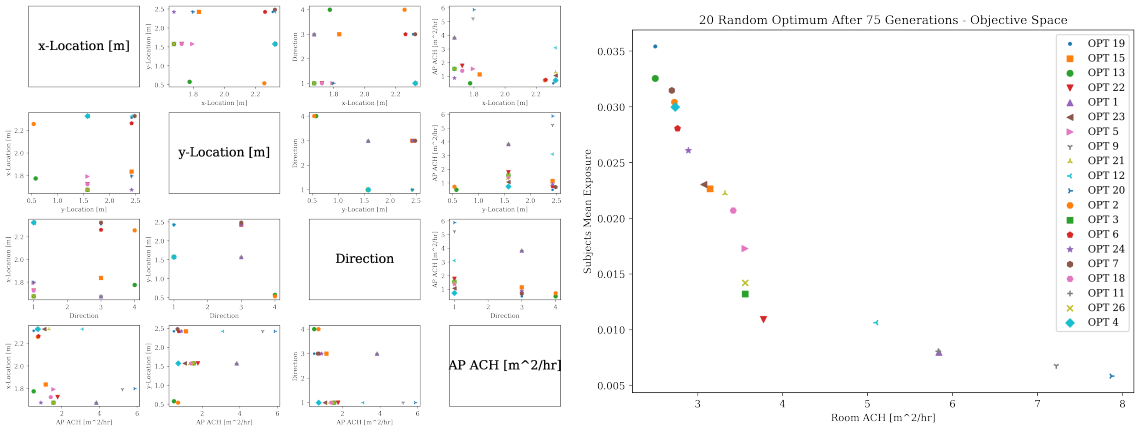
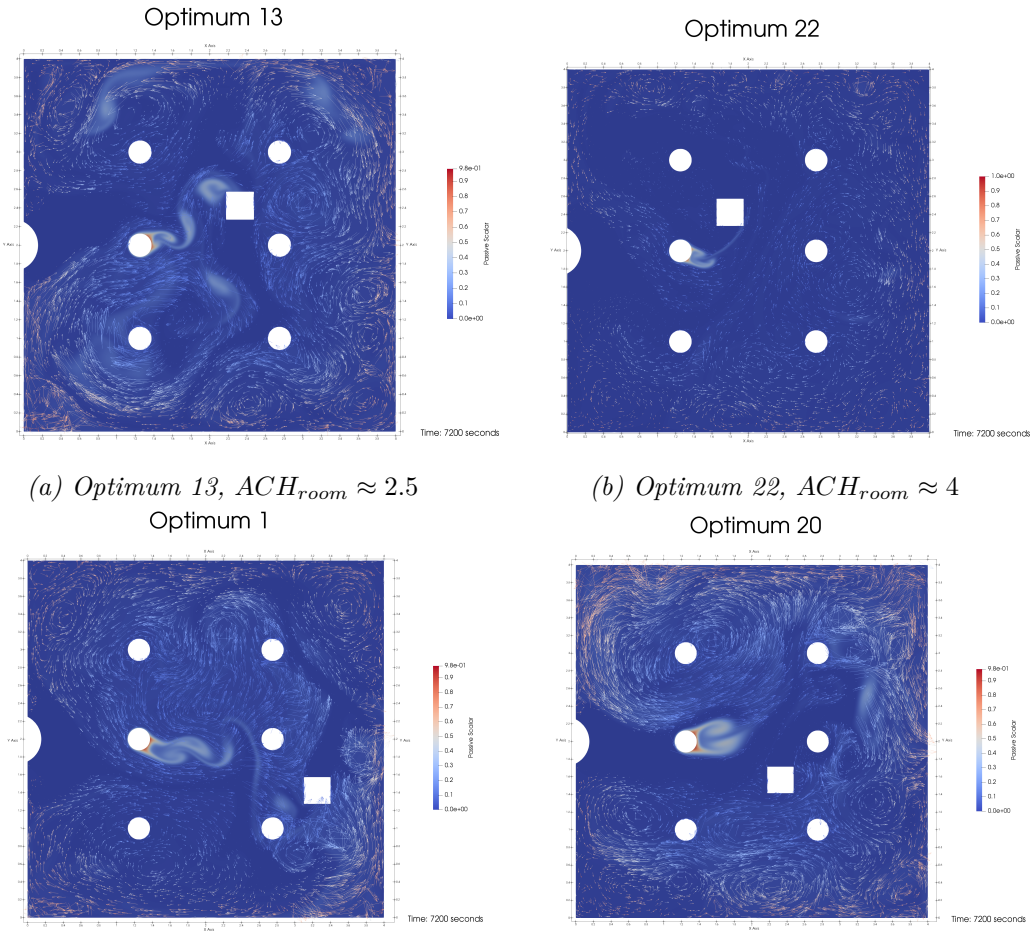


Figure 3.51: Air Purifier Configuration Optimization Study - 20 optimum after 75 generations. Randomly selected and labeled.



(a) Optimum 13, $ACH_{room} \approx 2.5$

(b) Optimum 22, $ACH_{room} \approx 4$

(c) Optimum 1, $ACH_{room} \approx 6$

(d) Optimum 20, $ACH_{room} \approx 8$

Figure 3.52: Air Purifier Configuration Optimization Study - Single air purifier optimum configurations found at different room ACH values.

Optimum 1 is the only case selected from the Pareto front where the air purifier is not located close to the center of the room. Instead, it is located close to the room outlet. However, a central location would achieve the similar results. This shows that multiple solutions exist at the same room ACH. The current optimization methodology is incapable of capturing all the possible best solutions at each ACH. This is another indication that a reworking of the methodology could generate better

results. Discussion of other possible methods is found in the **Conclusions** section.

3.4 Two Air Purifiers Configuration

3.4.1 Pre-Process

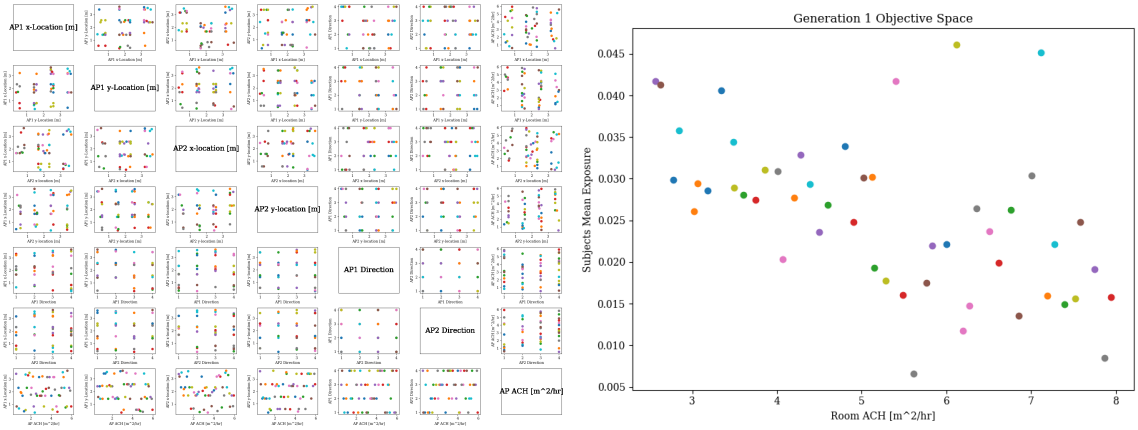


Figure 3.53: Air Purifier Configuration Optimization Study - First Generation - Design and Objective Spaces.

The inverse relationship between the room ACH and the subjects' mean exposure is clearly visible in the objective space plot of the first generation, just like with the single air purifier configuration optimization.

When working with high dimensional parameter spaces, such as the one used in this optimization study, is can become difficult to visualize the space in a meaningful way. Therefore, only the objective space will be included in the figures that follow.

3.4.2 Post-Process

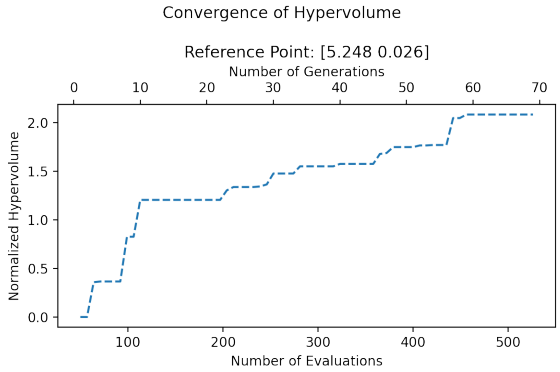


Figure 3.54: Air Purifier Configuration Optimization Study - Convergence of the hypervolume measured from the reference point to the Pareto front.

Convergence seems to have been reached at about 450 evaluations, or after about 57 generations.

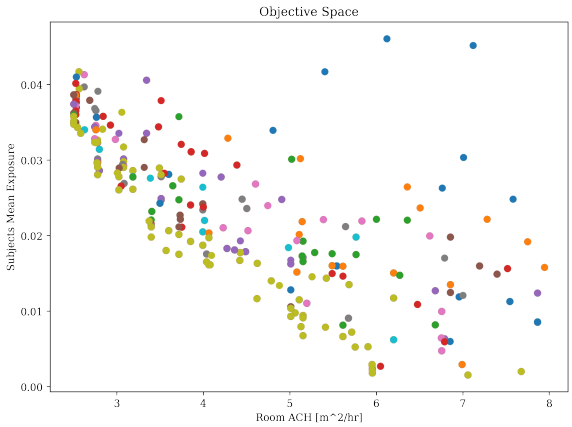


Figure 3.55: Air Purifier Configuration Optimization Study - Every generation.

The convergence at about Generation 57 can also be seen in the following figures. These figures give an idea of the algorithm's progress during the change in hypervolume seen around evaluation 450, a.k.a. Generation 57.

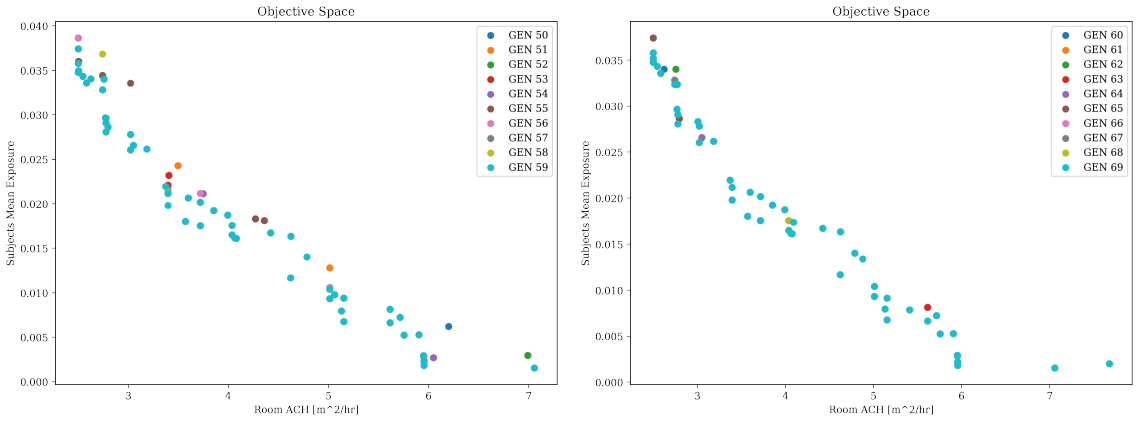
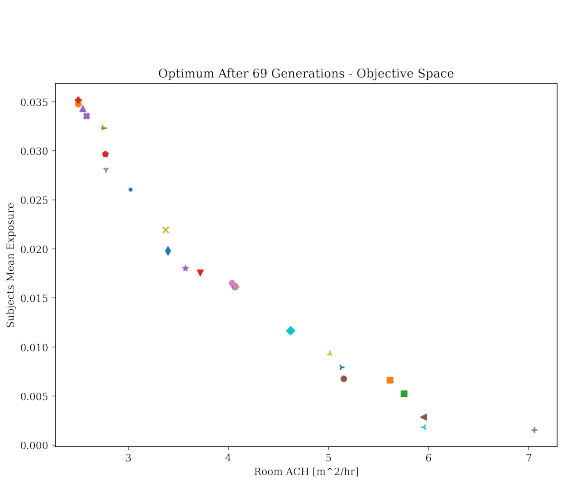
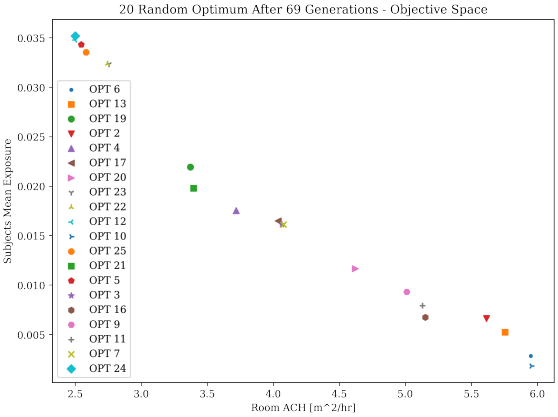


Figure 3.56: Air Purifier Configuration Optimization Study - Generations 50 through 70.

The final Pareto front after 69 generations is displayed below along with a set of 20 randomly selected and labeled optimum.

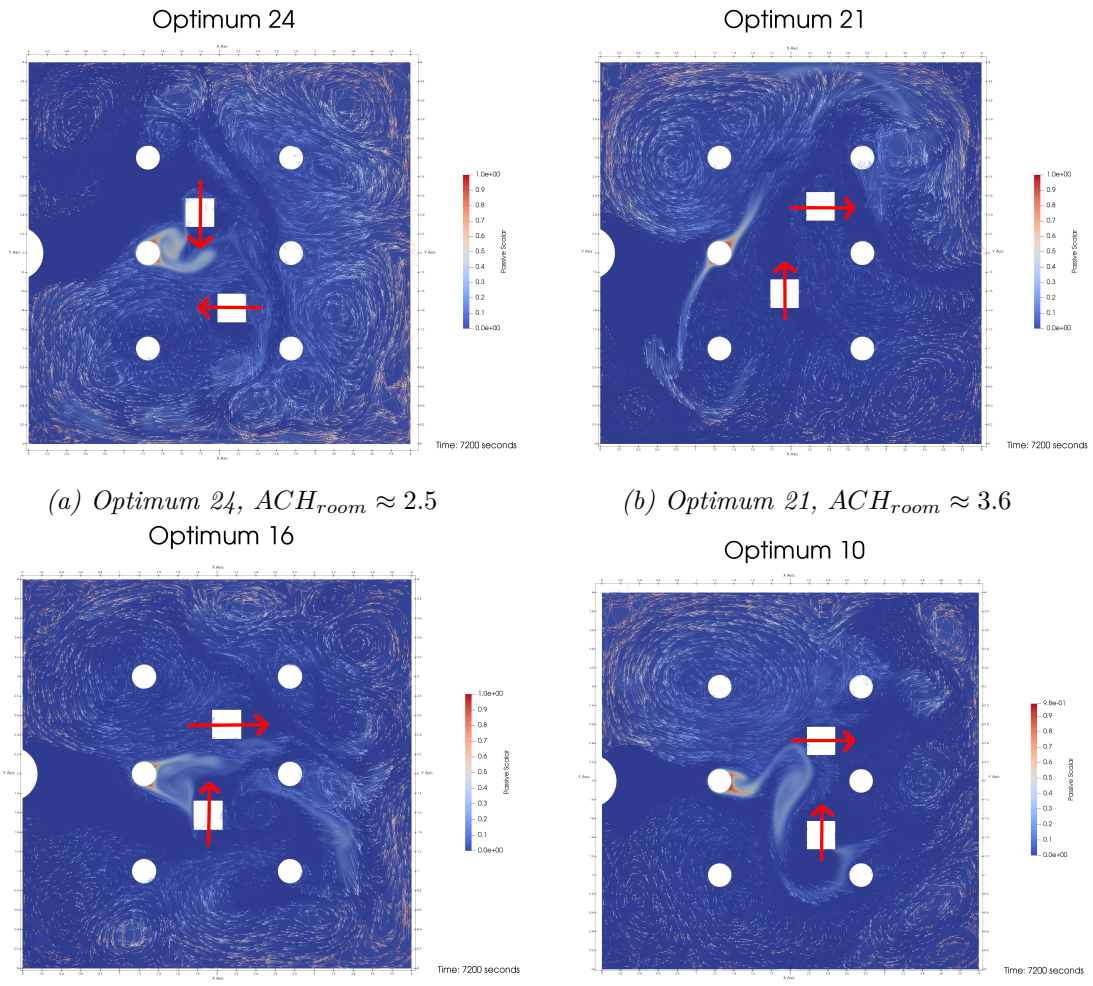


(a) Entire Pareto front.



(b) Air Purifier Configuration Optimization Study - 20 randomly selected and labeled individuals from the Pareto front.

A selection of critical points along the Pareto front are visualized below. Critical optimum are optimum along the Pareto front that form a "corner". These "corner" points indicate that the optimum has achieved better optimization than the surrounding optimum.



(a) Optimum 24, $ACH_{room} \approx 2.5$

(b) Optimum 21, $ACH_{room} \approx 3.6$

(c) Optimum 16, $ACH_{room} \approx 5.1$

(d) Optimum 10, $ACH_{room} \approx 6$

Figure 3.58: Air Purifier Configuration Optimization Study - Single air purifier optimum configurations found at different room ACH values.

Every critical optimal solution selected for visualization places the two air purifiers in the center of the room with perpendicular directionality relative to each other. Overall, 15/25 optimum have both air purifiers in a central location and perpendicular directionalities and 16/25 optimum have air purifiers with perpendicular directionality. Every optimum except for Optimum 1 has both air purifier centrally located, and even Optimum 1 has one air purifier centrally located. Centrally located

is defined as $1 < x_{ap} < 3$ and $0.5 < y_{ap} < 3.5$.

3.4.3 Follow-Up

Finally, the single air purifier and two air purifier optimization studies are compared.

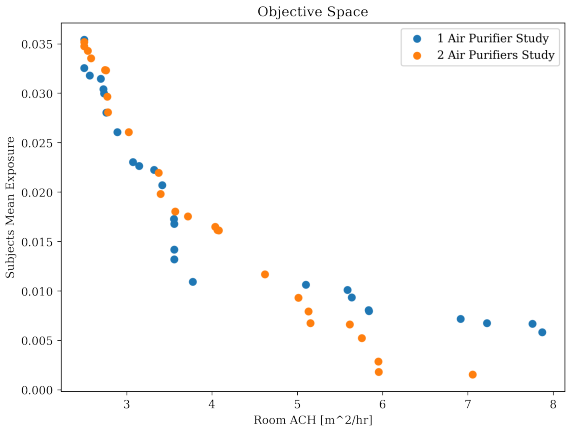


Figure 3.59: Air Purifier Configuration Optimization Study - Comparison between Optimum / Pareto front of 2 air purifiers optimization study versus 1 air purifier optimization study.

The results are somewhat surprising. The single air purifier achieves lower subject exposure at room ACHs between about 3.6 and 4.5. This could be a lack of convergence of the two air purifier optimization problem. However, there are other reasons a single air purifier may be able to achieve better mitigation of contaminants. One factor could be the strength of the single air purifier. In the two air purifier configuration the air purifier ACH is split between the two air purifiers, so neither is as powerful as the single air purifier case at the same ACH. Another factor could be simply the existence of a second air purifier. For all room ACH values less than about 4.7 the single air purifier out preforms or at least performance comparably to the two air purifier configuration.

It seems safe to assume that the optimization algorithms didn't fail to converge across this entire region of the Pareto front. This leads to the conclusion that there may be little to no benefit to using two air purifiers at room ACH levels less than 4.7.

The most likely explanation for the region of the Pareto front where the single air purifier configuration outperforms the two air purifiers is a combination of two factors. The strength of the single air purifier combined with the fact the domain only contains one momentum source (in addition to the room ventilation and artificial background turbulence). Intuitively, it would make sense that two air purifier would always have the advantage when mitigating air-borne disease. It is true that spacial distribution of contaminant "sinks", multiple locations where the contaminants are destroyed, is beneficial. However, the air purifiers modeled here do not only destroy passive scalars, they are also momentum sources. This means they inject momentum into the domain wherever they are located. This can be both beneficial and harmful. An increase in momentum is generally good for improving indoor air quality because it serves to dissipate contaminants, making them less concentrated and therefore less harmful. However, when the momentum is injected into the domain in a certain direct and at a specific location, it may cause contaminants to be circulated inside the room instead of finding their way to the room outlet. This re-circulation increases subject exposure.

In a higher fidelity aerosol contaminant models, the individual aerosol particles are modeled instead of using passive scalar to represent the aerosol particles. In this kind of model, the effects of momentum dissipation on the reduction of exposure would be greater. This is due to the breaking down of aerosols into particles too small to contain pathogens and the deposit of particles on surfaces. The size at which aerosol

particle are considered too small to contain contaminants is generally accepted as less than $0.3\mu m$ in diameter. [65] In this study, the individual particles are not modeled.

Chapter 4

Conclusion

Conducting the oscillating cylinder drag reduction optimization study provides a detailed example of how CFD population based optimization studies are conducted. This optimization problem also provides verification of the *pymooCFD* platform features, as was the goal. In addition to these original goals, the optimization conducted here is the only one known to this author that considers the skin friction force as an objective in a multi-objective optimization problem. This methodology makes the exploration of a practical implementation more realistic, as the energy cost should be considered fundamental to flow control. Many studies conducted during the explosion in the exploration of flow control techniques during the emergence of modern CFD in the 1990s and early 2000s failed to adequately address the energy cost of implementation. [36]

Additionally, the large search space and data generated by almost 800 LES simulations allowed the optimization algorithm to find a well converged Pareto set and Pareto front. These relationships should help to clarify questions regarding the practicality of this flow control method and the underlying fluid dynamics causing this

phenomenon. For example, many papers have shown that an increase in amplitude results in a larger range of forcing frequencies that are capable of reducing drag. [42] [40] [5] [41] However, when you factor in the minimization of the resistive force, the range of viable frequency stays the same as amplitude is increased.

Another interesting discovery found in this paper is that there appears to be an upper limit of frequency and amplitude where there is no further reduction of drag. While most papers on the subject state that this upper limit should exist, no papers found by this author actually explore what this upper limit is. This is likely due to the computational resources available at the time these studies were conducted. Overall, while many papers explore the influence of amplitude on the reduction of drag, they do not simultaneously consider the resistive forcing from the fluid on the cylinder.

Of the optimization studies conducted in this paper, the RANS jet boundary conditions provides the best justification for the use of the *pymooCFD* platform. The resulting Pareto set is non-intuitive and provides a family of solutions that allow the user to select which of the modeling predictions they want to favor.

This boundary condition calibration model simplification methodology shows promise. The methodology is straightforward, relatively simply and capable of being applied to a wide range of model simplification problems. However, this study only shows this method's ability to perform highly specific model simplification. More research needs to be conducted to explore the generalizability of models generated by this boundary condition calibration method.

One misstep in this study was the use of a peak velocity of $0.2\left[\frac{m}{s}\right]$. This is not a realistic breathing velocity. One study was found that uses a value of $0.5\left[\frac{m}{s}\right]$ as the

peak velocity. [56] All other similar studies found and all experimental measurement taken have greater peak velocity values.

In addition to this peak velocity not being realistic for modeling breathing, it also does not trigger the amount of turbulence that would cause a significant discrepancy between an LES and RANS simulation. This factor, along with the low peak breathing velocity, indicates that more high fidelity LES simulations should be conducted with a range of peak velocities. These numerical experiments can serve as a database for exploring the generalizability of boundary condition calibration. The calibrated steady jet RANS models produced here would also serve to capture the essential flow characteristics needed to model the spread of contaminants from different breathing patterns, all at a fraction of the computational cost of ZNMF jet LES models.

The air purifier configuration optimization studies were the original motivation behind this research paper. Over the past few years, studies have shown that COVID-19 is spread most effectively in indoor environments through to air-borne respiratory droplet containing viral loads. Room ventilation and air purifiers have been proven to be effective in preventing the spread of air-borne infectious disease in indoor environments. [16] [65] These ventilation and filtration methods are non-invasive to the individual, unlike masks. Additionally, masks can not be worn while eating and, in some cases, exercising. Therefore, proper ventilation and filtration are the preferred, and ultimately a more effective option in reducing the spread of air-borne infectious diseases in indoor environments. Compared to the modification of room ventilation, portable air purifiers are an affordable and easy to install. The next logical set of questions for fluid dynamics researchers were where these air purifiers should be placed to be most effective, how many are needed in a given space and how powerful should

they be?

The quest to answer these questions met many road blocks. Firstly, was how to model a high exposure risk scenario. High risk scenarios involve multiple subjects in a closed space all breathing, and at least one of which is breathing out infected respiratory droplets. Additionally, there is the modeling of room ventilation and air purifier filtration, as well as the heating sources, which includes lighting and the subjects' breaths and bodies. There is also the outside influence of doorways opening and closing. On top of all this, there is the complex geometry of a room containing furniture and human beings. As discussed in Section 2, **Cases**, this would all be impossible to model given the computational limitation of today's technology. So vast simplification must be made. This led to the exploration of model simplification using optimization algorithm and resulted in the RANS jet optimization study discussed above.

In the end, a 2D room model was chosen as the only viable model for use with an optimization algorithm due to its low computational cost. The goal of producing quantitative results was abandon, and the resulting model was purely qualitative. The results, once validation, may provide some interesting insights into configuration of indoor air purifier in a room.

At lower room ACH values, the results show that a single strong air purifier may be just as effective, if not more effective, compared to a two air purifier configuration each with half the power of the single air purifier. Another conclusion that can be drawn is that the most effective configuration of two air purifiers is placing them in central locations with directionalities perpendicular to each other. This causes flow patterns which force contaminants into turbulent layers along the outer walls

until the contaminants are filtered or exit through the room's outlet. The issue with this conclusion is that it is a somewhat intuitive solution. However, the fact that the optimization algorithm arrived at this conclusion after testing hundreds of possibilities proves that this is the most effective configuration at almost every room ACH level. Additionally, with two air purifier configuration, there is almost no advantage to an ACH level higher than 6. This supports other literature on the subject, which recommends an ACH of about 6 as a general guideline for mitigating the spread of airborne diseases. [66] Of course, all these conclusions are restricted by the model. When discussing air purifier directionality, for example, an entire third spacial dimension is not being considered. In reality, air purifiers typically point upward into the third dimension that is not modeled. Still, these result should serve to inform the experimentation that must be conducted to draw practical conclusions.

4.1 Future Work

Future works should include the previously mentioned experimental validation of the air purifier optimization results, as well as the expansion of the ZNMF jet LES simulation database to produce simplified steady jet RANS models. In addition, the use of the *pymoo Repair* subclass to restrict air purifier placement is problematic, as discussed in Section 3, **Results & Discussion**. It may be beneficial to restrict the air purifiers so that they are only placed in pre-selected locations. This could prevent the overlap of the air purifiers and remove the need for a *Repair* subclass that prevents the genetic algorithms from working as intended. The drawback to this method was thought to be that it would not allow the production of non-intuitive

placement, but this seems to have happened anyway. This change would also reduce the number of parameters needed, as there would no longer be two parameters, an x- and y-location, for each air purifier.

The use of different metrics for subject exposure and room ACH should also be experimented with. Even the addition of a third objective metric should be considered. Possibilities include tracking the average contaminant levels in the entire domain or measuring residence time of the contaminants. A more radically different approach would the removal of the subjects and the use of a point source for contaminants, or simply have the initial condition be the entire domain filled with contaminants. This way the air purifiers would be free to be placed is a wider range of location throughout the room.

Finally, there is always opportunity for further development of the *pymooCFD* platform. For one thing, thorough testing of the use of the *pymoo* constraints interface has not been conducted. Another untested feature is the use of adaptive parallel computing through the use of the python Dask library.

An important note is that the development of this platform makes it easy to conduct a study comparing the effectiveness of different optimization algorithms on CFD problems.

There is also an extensive set of features that could be added, and re-working of the program's architecture should be conducted alongside further development.

One possible feature would be a problem database which stored the results of any CFD case preventing the re-running of high-cost evaluations across optimization algorithm runs. This development was started and can be seen in the *CaseDB* branch on GitHub. A database such as this could help to address the stochastic nature of

CFD problems. By compiling multiple runs of the same CFD case, the database could average these results, creating more confidence in the CFD results.

Another possible feature would be the creation of a separate IPython notebook file for each optimization problem based on a template, this way the notebook could be edited and customized for each optimization problem.

In terms of program architecture, better integration into the *pymoo* platform could be achieved. For example, instead of using a *plotScatter* method in the *OptRun* class, the creation of a *pymoo.visualization.scatter.Scatter* subclass would be a better solution. Lastly, there is the integration of the newly released *pysamoo*, surrogate-assisted multi-objective optimization. [32] This extension to *pymoo* has optimization algorithms specifically designed for high-cost evaluation problems such as CFD problems. These algorithms are likely the future of research in high-cost evaluation optimization problems. The CFD research community would benefit immensely from the use of such algorithms.

Bibliography

[1] Javier Lobato Perez. Genetic algorithms applied in computer fluid dynamics for multiobjective optimization.

[2] Reshape. Low-carbon building codes â local vs global optima.

[3] Reza Zadeh. The hard thing about deep learning, 2016.

[4] Andrew Ng.

[5] Sungho Choi, Haecheon Choi, and Sangmo Kang. Characteristics of flow over a rotationally oscillating cylinder at low reynolds number. *Physics of Fluids*, 14(8):2767–2777, 2002.

[6] Lin (Íć) Lu, Jian-Min (ç§ŚÍŧşć) Qin, Bin (ćŧć) Teng, and Yu-Cheng (ćçć) Li. Numerical investigations of lift suppression by feedback rotary oscillation of circular cylinder at low reynolds number. *Physics of Fluids*, 23(3):033601, 2011.

[7] Ansys 17.1 delivers optimized system performance for early design process, 2016.

[8] Wolfram Stadler. *Multicriteria Optimization in Engineering and in the Sciences*, volume 37. Springer Science & Business Media, 1988.

[9] Suyun Liu and Luis Nunes Vicente. The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning, 2019.

[10] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[11] Debabrata Mahapatra and Vaibhav Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In Hal DaumĂŠ III and Aarti Singh, editors, *Proceedings of the 37th International*

Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, pages 6597–6607. PMLR, 13–18 Jul 2020.

[12] Xingchao Liu, Xin Tong, and Qiang Liu. Profiling pareto front with multi-objective stein variational gradient descent. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14721–14733. Curran Associates, Inc., 2021.

[13] Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[14] Forrester T. Johnson, Edward N. Tinoco, and N. Jong Yu. Thirty years of development and application of cfd at boeing commercial airplanes, seattle. *Computers Fluids*, 34(10):1115–1151, 2005.

[15] Nina Morozova, R Capdevila, FX Trias, and A Oliva. Towards real-time cfd simulation of indoor environment. In *Proceedings of 10th International Conference on Computational Fluid Dynamics*, volume 755, 2018.

[16] Haoguang Yang, Mythra V. Balakuntala, Jhon J. QuiĂąones, Upinder Kaur, Abigayle E. Moser, Ali Doosttalab, Antonio Esquivel-Puentes, Tanya Purwar, Luciano Castillo, Xin Ma, Lucy T. Zhang, and Richard M. Voyles. Occupant-centric robotic air filtration and planning for classrooms for safer school reopening amid respiratory pandemics. *Robotics and Autonomous Systems*, 147:103919, 2022.

[17] Xiaodong Kang and Wei Li. Moth-inspired plume tracing via multiple autonomous vehicles under formation control. *Adaptive Behavior*, 20(2):131–142, 2012.

[18] Shi-Jie Cao. Challenges of using cfd simulation for the design and online control of ventilation systems. *Indoor and Built Environment*, 28(1):3–6, 2019.

[19] C. A. Mack. Fifty years of moore's law.

[20] R. R. Schaller. Moore's law: past, present and future. *EEE Spectrum, vol. 34, no. 6, pp. 52-59, June 1997*, 34, no. 6:52–59, 1997.

[21] Parviz Moin and Krishnan Mahesh. Direct numerical simulation: a tool in turbulence research. *Annual review of fluid mechanics*, 30(1):539–578, 1998.

[22] Marcel Lesieur and Olivier Metais. New trends in large-eddy simulations of turbulence. *Annual review of fluid mechanics*, 28(1):45–82, 1996.

[23] Vincent Moureau, Pascale Domingo, and Luc Vervisch. Design of a massively parallel cfd code for complex geometries. *Comptes Rendus Mécanique*, 339(2-3):141–148, 2011.

[24] BA Pettersson Reif and PA Durbin. *Statistical theory and modeling for turbulent flows*. John Wiley & Sons, 2011.

[25] Ian Pond, Alireza Ebadi, Yves Dubief, and Christopher M White. An integral validation technique of rans turbulence models. *Computers & Fluids*, 149:150–159, 2017.

[26] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51(1):357–377, 2019.

[27] William Oberkampf, Frederick Blottner, and Daniel Aeschliman. Methodology for computational fluid dynamics code verification/validation. In *Fluid dynamics conference*, page 2226, 1995.

[28] Christopher Roy and William Oberkampf. A complete framework for verification, validation, and uncertainty quantification in scientific computing. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 124, 2010.

[29] handsomeboy. how to draw hypervolume with tikz, 2016.

[30] Ali Ahrari, Julian Blank, Kalyanmoy Deb, and Xianren Li. A proximity-based surrogate-assisted method for simulation-based design optimization of a cylinder head water jacket. *Engineering Optimization*, 53(9):1574–1592, 2021.

[31] Julian Blank and Kalyanmoy Deb. Psa: A family of probabilistic surrogate-assisted algorithms for single-objective optimization.

[32] Julian Blank and Kalyanmoy Deb. pysamoo: Surrogate-assisted multi-objective optimization in python, 2022.

[33] Mary Kate Joyce. Fortune adds ansys to 2020 future 50 list, 2020.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[35] Hamed Safikhani. Modeling and multi-objective pareto optimization of new cyclone separators using cfd, anns and nsga ii algorithm. *Advanced Powder Technology*, 27(5):2277–2284, 2016.

[36] P. T. Tokumaru and P. E. Dimotakis. Rotary oscillation control of a cylinder wake. *Journal of Fluid Mechanics*, 224:77â90, 1991.

[37] S. Taneda. Visual observations of the flow past a sphere at reynolds numbers between 104 and 106. *Journal of Fluid Mechanics*, 85(1):187â192, 1978.

[38] Atsushi Okajima. Strouhal numbers of rectangular cylinders. *Journal of Fluid Mechanics*, 123:379â398, 1982.

[39] J. R. Filler, P. L. Marston, and W. C. Mih. Response of the shear layers separating from a circular cylinder to small-amplitude rotational oscillations. *Journal of Fluid Mechanics*, 231:481â499, 1991.

[40] J.-W. He, R. Glowinski, R. Metcalfe, A. Nordlander, and J. Periaux. Active control and drag optimization for flow past a circular cylinder: I. oscillatory cylinder rotation. *Journal of Computational Physics*, 163(1):83–117, 2000.

[41] D. Shiels and A. Leonard. Investigation of a drag reduction on a circular cylinder in rotary oscillation. *Journal of Fluid Mechanics*, 431:297â322, 2001.

[42] Michel Bergmann, Laurent Cordier, and Jean-Pierre Brancher. On the power required to control the circular cylinder wake by rotary oscillations. *Physics of Fluids*, 18(8):088103, 2006.

[43] C H K Williamson. Vortex dynamics in the cylinder wake. *Annual Review of Fluid Mechanics*, 28(1):477–539, 1996.

[44] F. M. Mahfouz and H. M. Badr. Flow Structure in the Wake of a Rotationally Oscillating Cylinder . *Journal of Fluids Engineering*, 122(2):290–301, 11 1999.

[45] Seong-Jae Kim and Choung Mook Lee. Numerical investigation of cross-flow around a circular cylinder at a low-reynolds number flow under an electromagnetic force. *KSME International Journal*, 16(3):363–375, Mar 2002.

[46] Jeongyoung Park, Kiyoung Kwon, and Haecheon Choi. Numerical solutions of flow past a circular cylinder at reynolds numbers up to 160. *KSME International Journal*, 12(6):1200–1205, Nov 1998.

[47] Sintu Singha and K.P. Sinhamahapatra. Flow past a circular cylinder between parallel walls at low reynolds numbers. *Ocean Engineering*, 37(8):757–769, 2010.

[48] Ronald D. Henderson. Nonlinear dynamics and pattern formation in turbulent wake transition. *Journal of Fluid Mechanics*, 352:65â112, 1997.

[49] Taha Khademinejad, Pouyan Talebizadeh Sardari, and Hassan Rahimzadeh. Numerical study of unsteady flow around a square cylinder in compare with circular cylinder. 02 2015.

[50] Nikolaos D. Katopodes. Chapter 5 - viscous fluid flow. In Nikolaos D. Katopodes, editor, *Free-Surface Flow*, pages 324–426. Butterworth-Heinemann, 2019.

[51] Anna BuliŃska and Zbigniew BuliŃski. A cfd analysis of different human breathing models and its influence on spatial distribution of indoor air parameters. *Computer Assisted Methods in Engineering and Science*, 22(3):213–227, 2017.

[52] Erik Bjørn and Peter V. Nielsen. *CFD Simulations of Contaminant Transport between two Breathing Persons*. Number 96 in Indoor Environmental Engineering. Dept. of Building Technology and Structural Engineering, Aalborg University, Denmark, 1998. Proceedings of ROOMVENT '98, Sixth International Conference on Air Distribution in Rooms, Stockholm, Sweden, 1998, Vol. 2, pp. 133-140 PDF for print: 15 pp.

[53] Debbie Leusink, David Alfano, and Paola Cinnella. Multi-fidelity optimization strategy for the industrial aerodynamic design of helicopter rotor blades. *Aerospace Science and Technology*, 42:136–147, 2015.

[54] Application of low- and high-fidelity simulation tools to helicopter rotor blade optimization. *Journal of the American Helicopter Society*, 58(4), 2013.

[55] John E. Cater and Julio Soria. The evolution of round zero-net-mass-flux jets. *Journal of Fluid Mechanics*, 472:167â200, 2002.

[56] Manouk Abkarian, Simon Mendez, Nan Xue, Fan Yang, and Howard A Stone. Speech can produce jet-like transport relevant to asymptomatic spreading of virus. *Proceedings of the National Academy of Sciences*, 117(41):25237–25245, 2020.

[57] Yuhao (ä¸Ľé¨ćľŠ) Yan, Chunning (ÍćĽÍŽ) Ji, and Narakorn Srinil. Three-dimensional flip-flopping flow around a pair of dual-stepped circular cylinders in a side-by-side arrangement. *Physics of Fluids*, 32(12):123608, 2020.

[58] D. W. Guillaume and J. C. LaRue. Investigation of the flopping regime with two-, three- and four-cylinder arrays. *Experiments in Fluids*, 27(2):145–156, Jul 1999.

[59] T. Mullin, James Seddon, M. Mantle, and A.J. Sederman. Bifurcation phenomena in the flow through a sudden expansion in a circular pipe. *Physics of Fluids*, 21, 01 2009.

[60] Iain S Walker. Best practices guide for residential hvac retrofits. 8 2003.

[61] P.E. Mike Moore. The case for mechanical ventilation and air tightness requirements in florida. Technical report, Home Ventilation Institute, 2018.

[62] Sai Ranjeet Narayanan and Suo Yang. Airborne transmission of virus-laden aerosols inside a music classroom: Effects of portable purifiers and aerosol injection rates. *Physics of Fluids*, 33(3):033307, 2021.

[63] Don T. Stevens. Building codes, ventilation rates, and certified rates, and certified performance. Technical report, Ventilation Track Comfortech, 2007.

[64] Environmental Protection Agency. What is a hepa filter?, 2022.

[65] WG Lindsley, RC Derk, and et al. Coyle, JP. Efficacy of portable air cleaners and masking for reducing indoor exposure to simulated exhaled sars-cov-2 aerosols â united states, 2021.

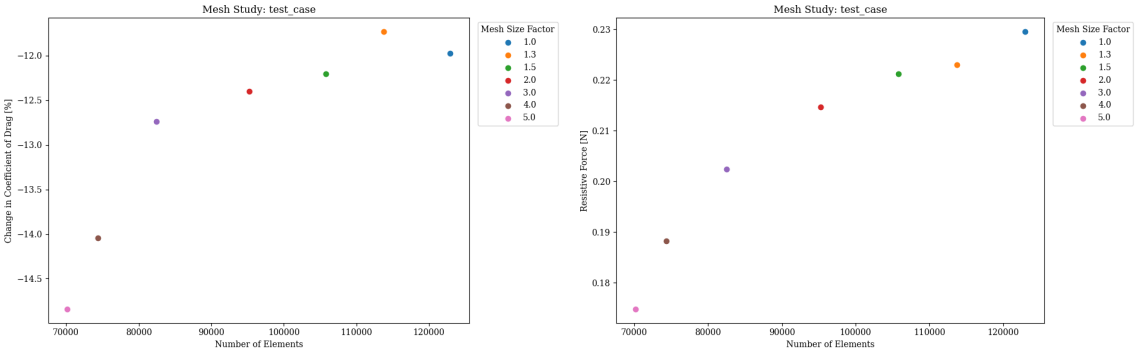
[66] Craig Hoellwarth. Indoor ventilation - best practices guide based on ashrae 62.2. Technical report, Home Ventilation Institute, 2010.

Chapter 5
Appendix

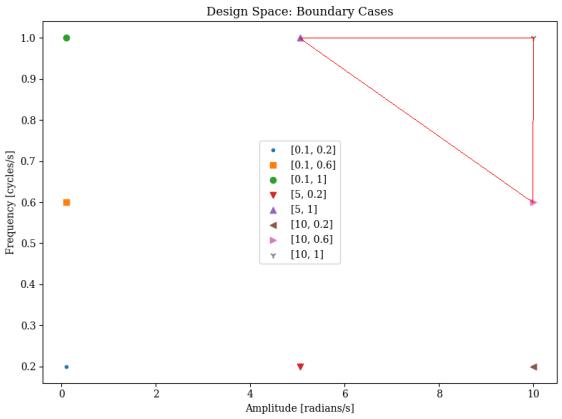
5.1 Mesh Studies

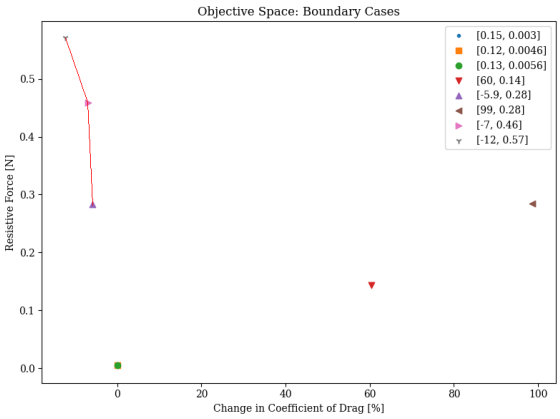
5.1.1 Oscillating Cylinder

Test Case: $\omega_{amp} = 5.05,\ f = 0.6$

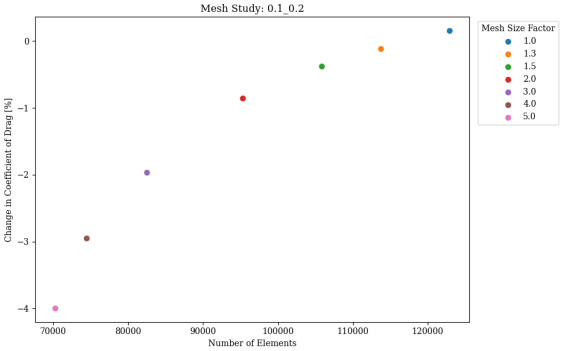


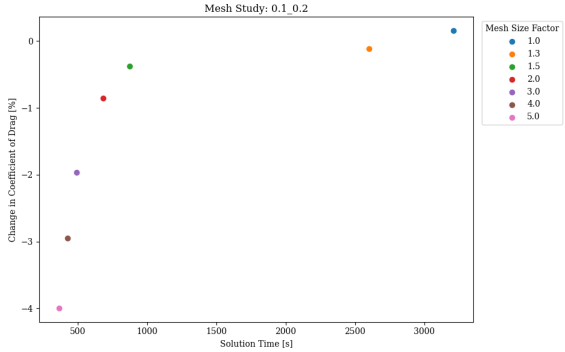
Boundary Cases:

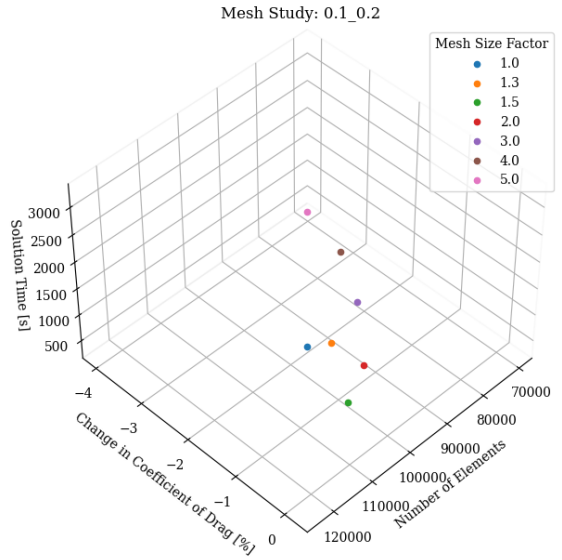


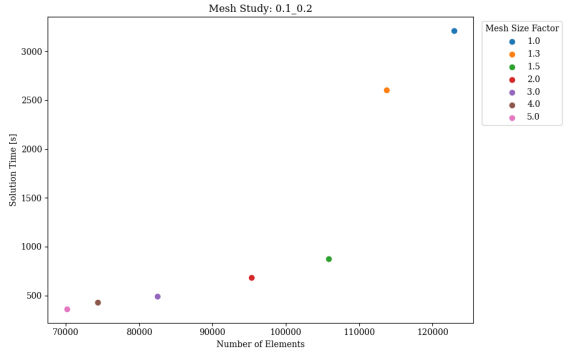


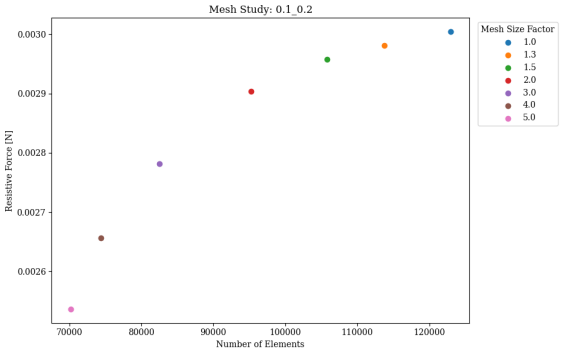
Boundary Case: $\omega_{amp} = 0.1,\ f = 0.2$

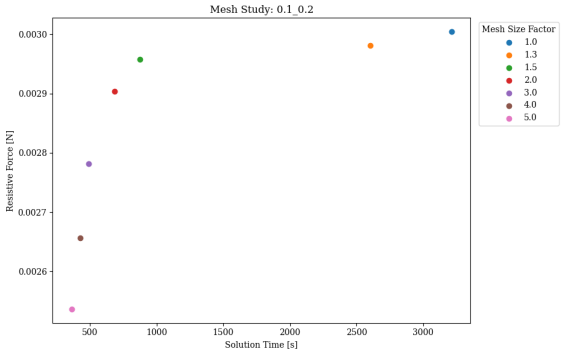


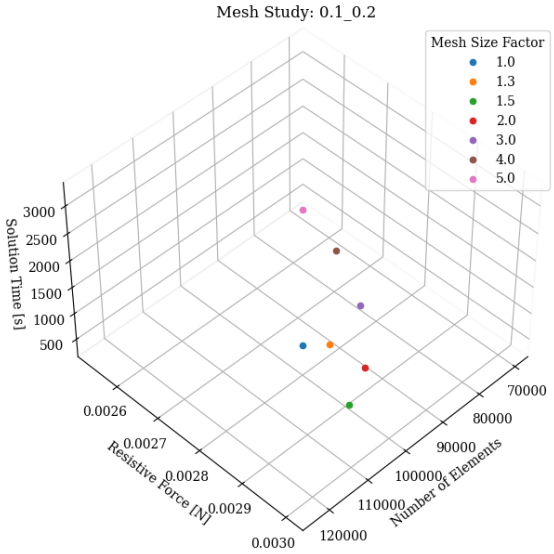




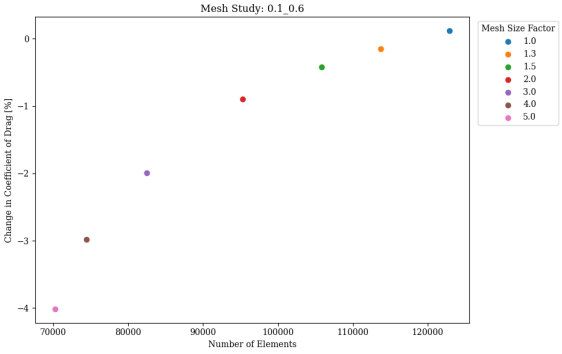


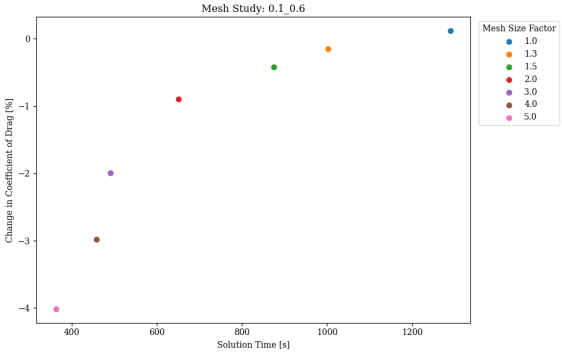


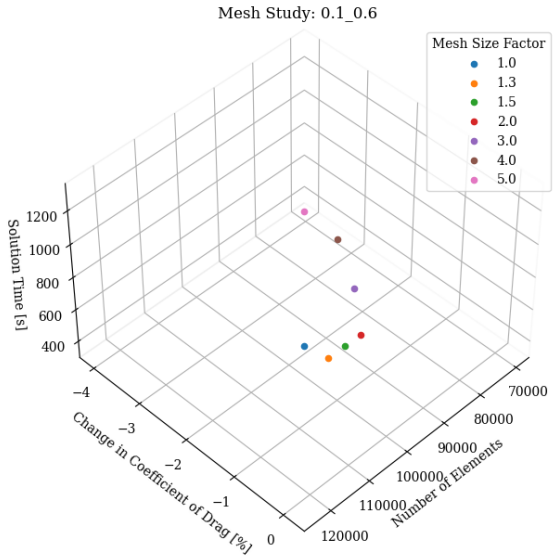


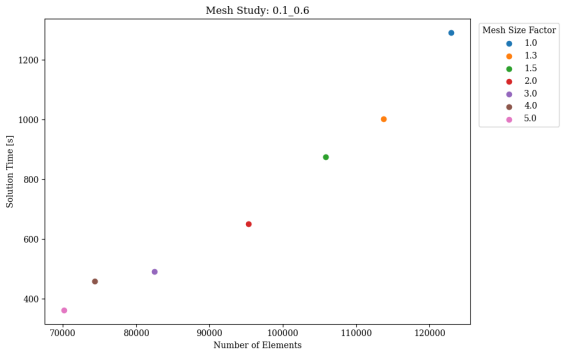


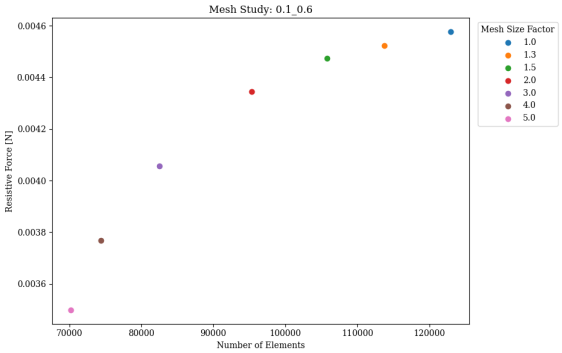
Boundary Case: $\omega_{amp} = 0.1,\ f = 0.6$

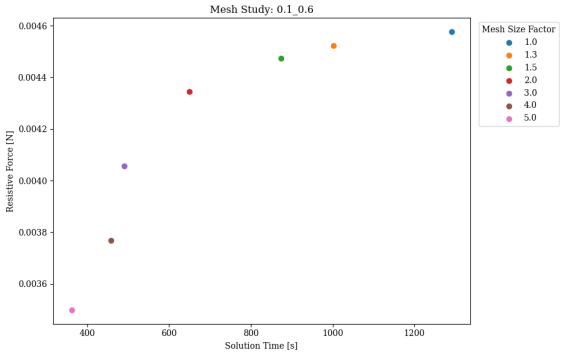


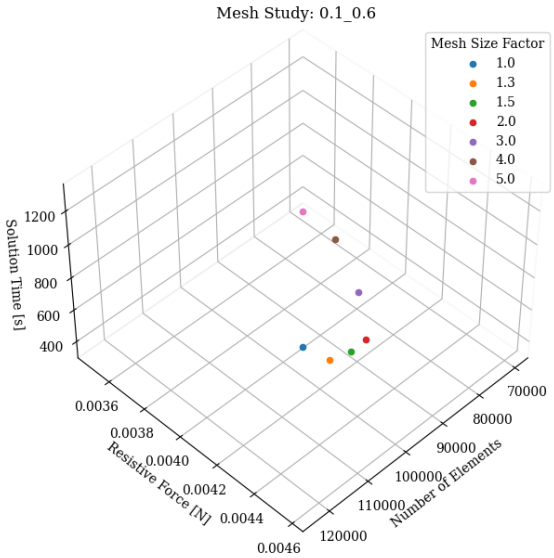



Mesh Study: 0.1_0.6
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
1200
1000
800
600
400
−4
−3
−2
−1
0
Change in Coefficient of Drag [%]
70000
80000
90000
100000
110000
120000
Number of Elements

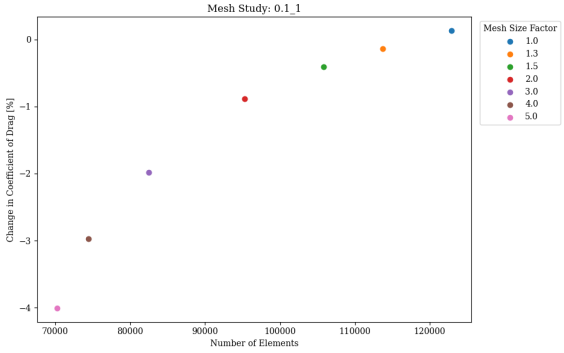

Mesh Study: 0.1_0.6
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
1200
1000
800
600
400
70000
80000
90000
100000
110000
120000
Number of Elements

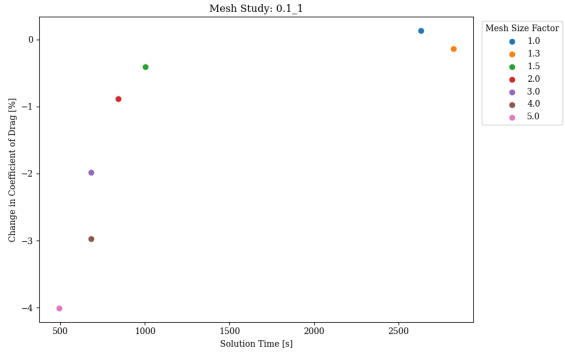

Mesh Study: 0.1_0.6
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.0046
0.0044
0.0042
0.0040
0.0038
0.0036
70000
80000
90000
100000
110000
120000
Number of Elements

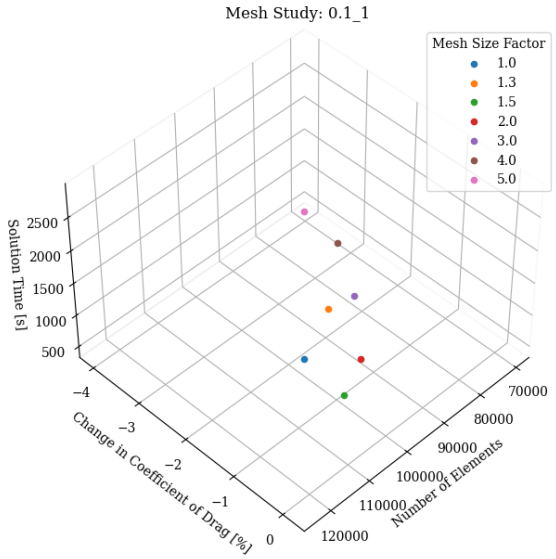

Mesh Study: 0.1_0.6
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.0046
0.0044
0.0042
0.0040
0.0038
0.0036
400
600
800
1000
1200
Solution Time [s]

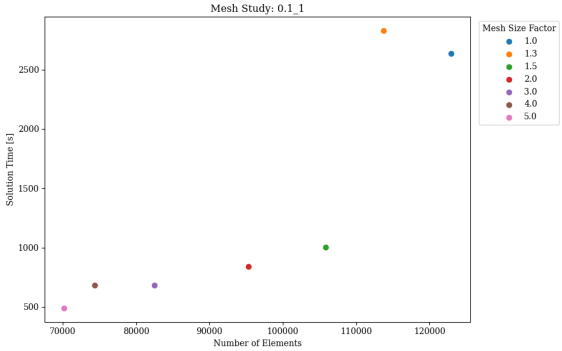


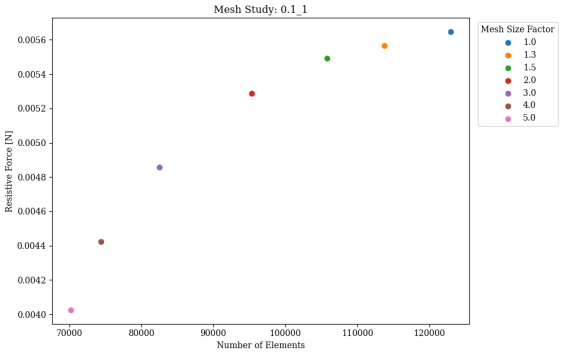
Boundary Case: $\omega_{amp} = 0.1,\ f = 1.0$

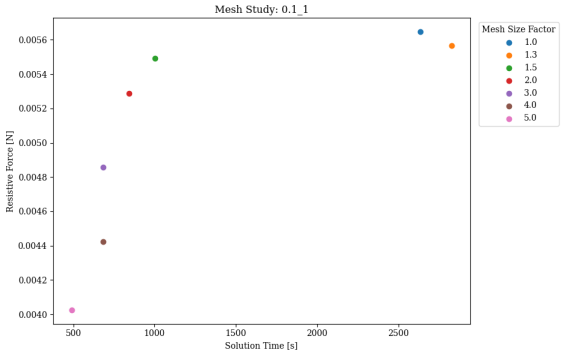


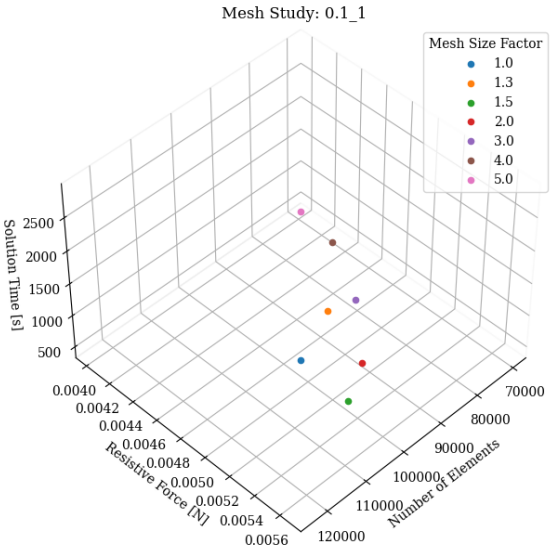



Mesh Study: 0.1_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
2500
2000
1500
1000
500
−4
−3
−2
−1
0
Change in Coefficient of Drag [%]
70000
80000
90000
100000
110000
120000
Number of Elements

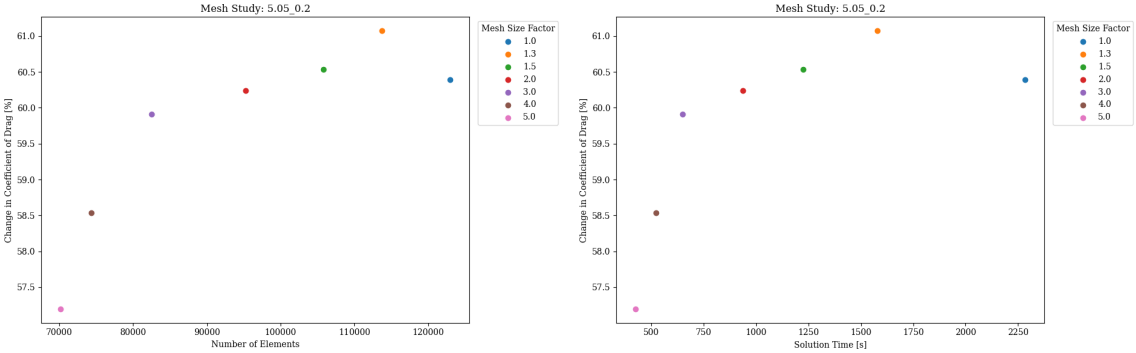

Mesh Study: 0.1_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
2500
2000
1500
1000
500
70000
80000
90000
100000
110000
120000
Number of Elements

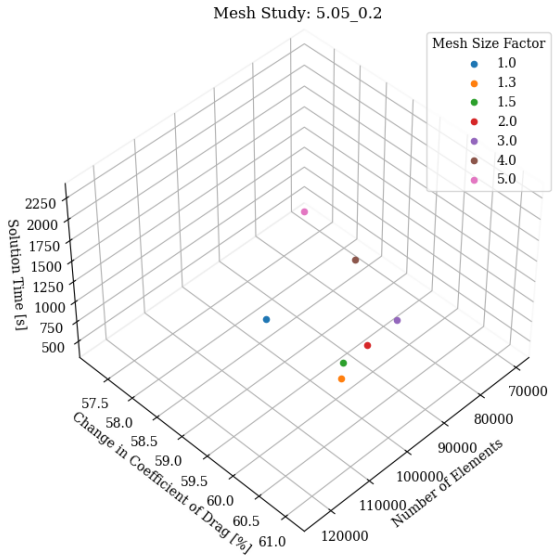

Mesh Study: 0.1_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.0056
0.0054
0.0052
0.0050
0.0048
0.0046
0.0044
0.0042
0.0040
70000
80000
90000
100000
110000
120000
Number of Elements

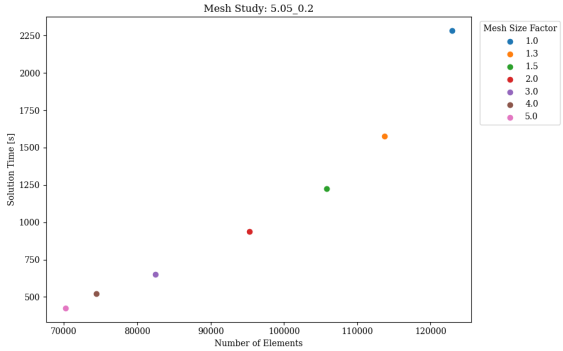

Mesh Study: 0.1_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.0056
0.0054
0.0052
0.0050
0.0048
0.0046
0.0044
0.0042
0.0040
500
1000
1500
2000
2500
Solution Time [s]

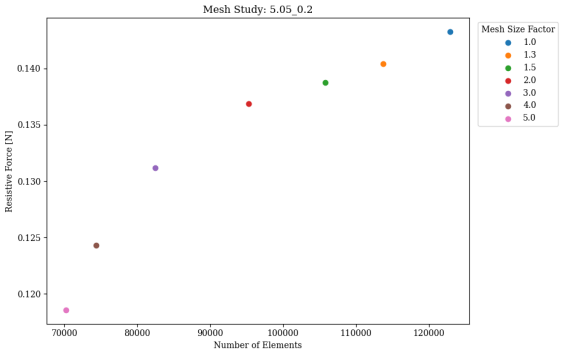


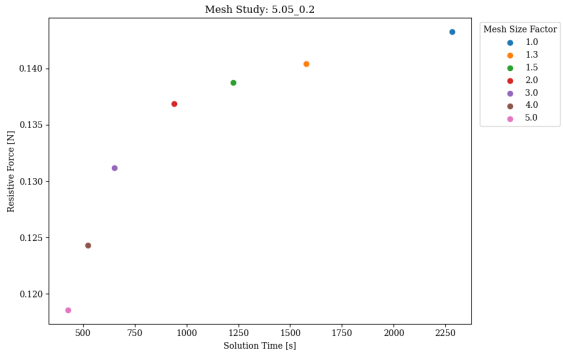
Boundary Case: $\omega_{amp} = 5.0,\ f = 0.2$

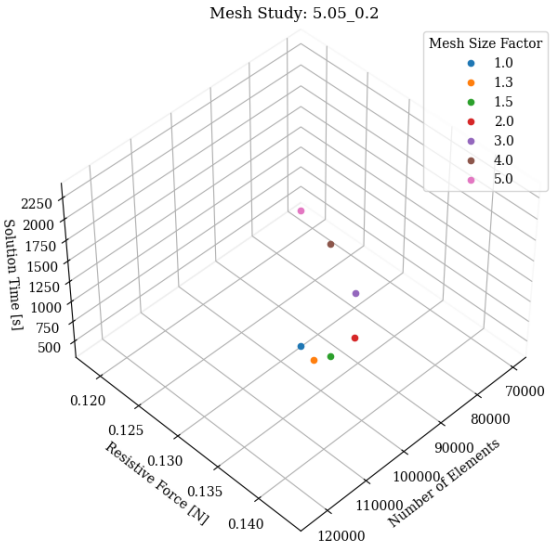



Mesh Study: 5.05_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
2250
2000
1750
1500
1250
1000
750
500
Change in Coefficient of Drag [%]
57.5
58.0
58.5
59.0
59.5
60.0
60.5
61.0
Number of Elements
70000
80000
90000
100000
110000
120000

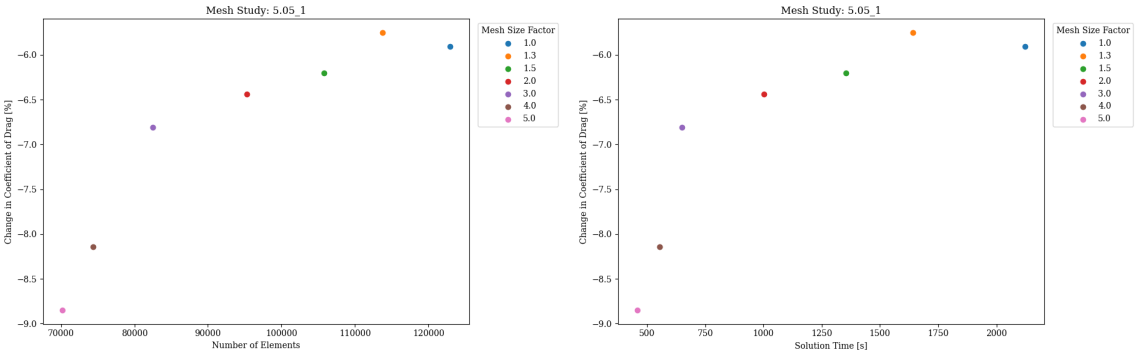

Mesh Study: 5.05_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
2250
2000
1750
1500
1250
1000
750
500
70000
80000
90000
100000
110000
120000
Number of Elements

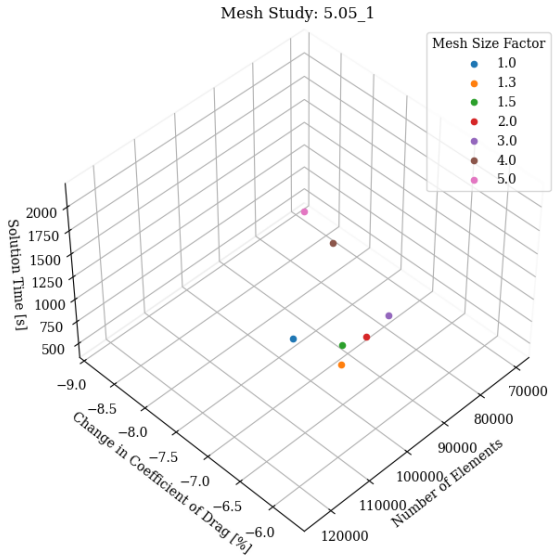

Mesh Study: 5.05_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.140
0.135
0.130
0.125
0.120
70000
80000
90000
100000
110000
120000
Number of Elements

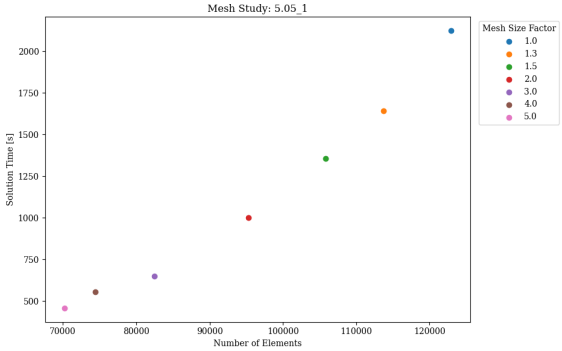

Mesh Study: 5.05_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.140
0.135
0.130
0.125
0.120
500
750
1000
1250
1500
1750
2000
2250
Solution Time [s]

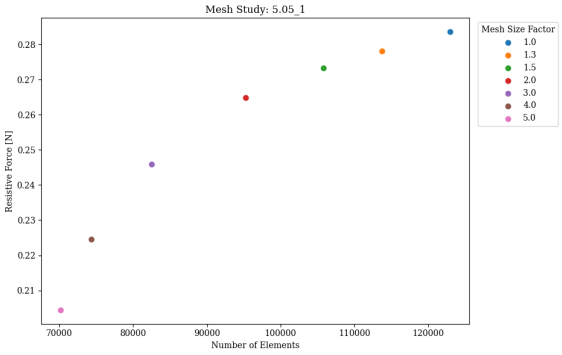


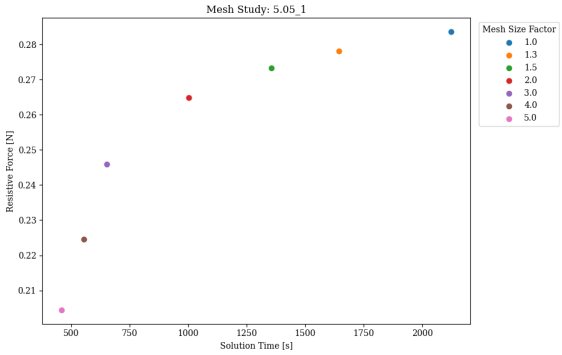
Boundary Case: $\omega_{amp} = 5.05,\ f = 1.0$

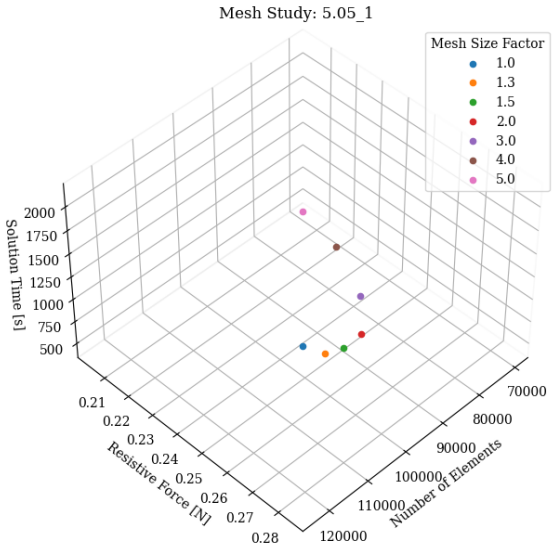



Mesh Study: 5.05_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
2000
1750
1500
1250
1000
750
500
−9.0
−8.5
−8.0
−7.5
−7.0
−6.5
−6.0
Change in Coefficient of Drag [%]
70000
80000
90000
100000
110000
120000
Number of Elements

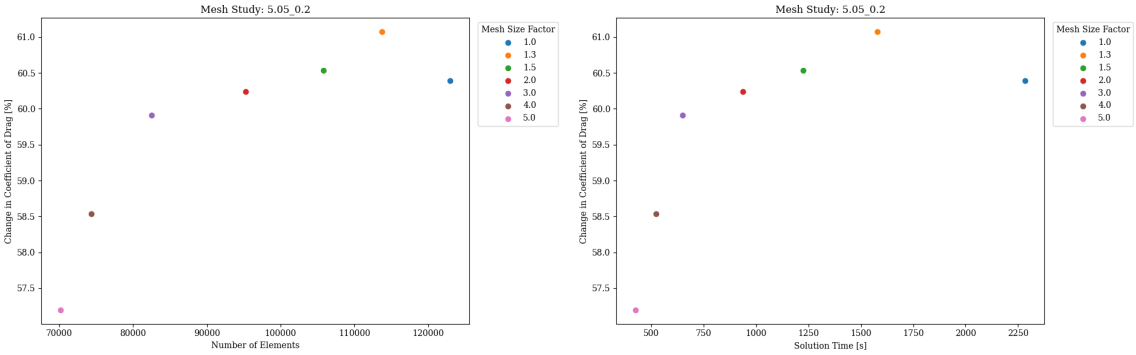

Mesh Study: 5.05_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
2000
1750
1500
1250
1000
750
500
70000
80000
90000
100000
110000
120000
Number of Elements

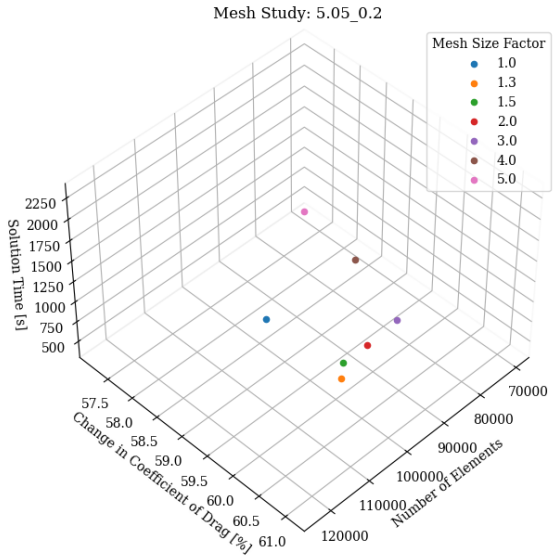

Mesh Study: 5.05_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.28
0.27
0.26
0.25
0.24
0.23
0.22
0.21
70000
80000
90000
100000
110000
120000
Number of Elements

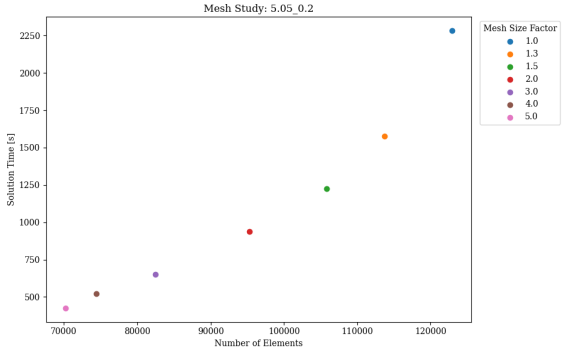

Mesh Study: 5.05_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.28
0.27
0.26
0.25
0.24
0.23
0.22
0.21
500
750
1000
1250
1500
1750
2000
Solution Time [s]

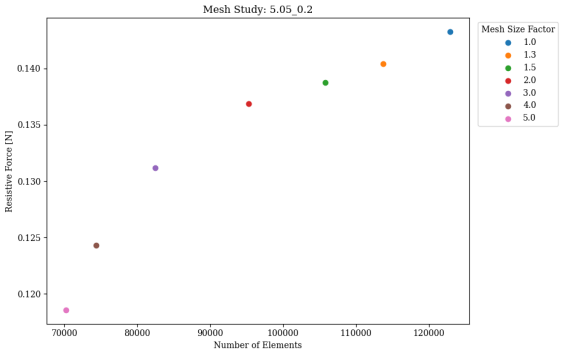


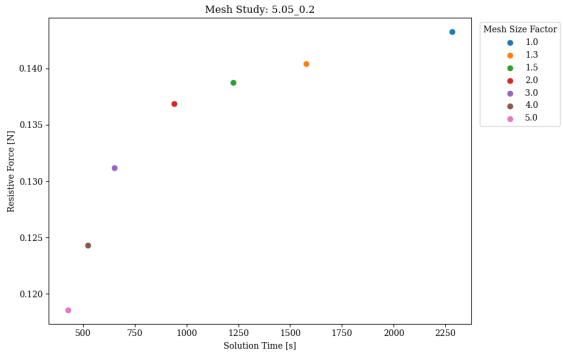
Boundary Case: $\omega_{amp} = 5.05,\ f = 0.2$

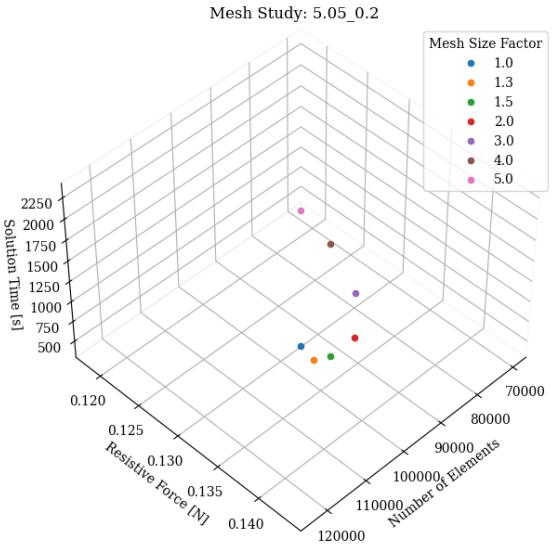



Mesh Study: 5.05_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
2250
2000
1750
1500
1250
1000
750
500
Change in Coefficient of Drag [%]
57.5
58.0
58.5
59.0
59.5
60.0
60.5
61.0
Number of Elements
70000
80000
90000
100000
110000
120000

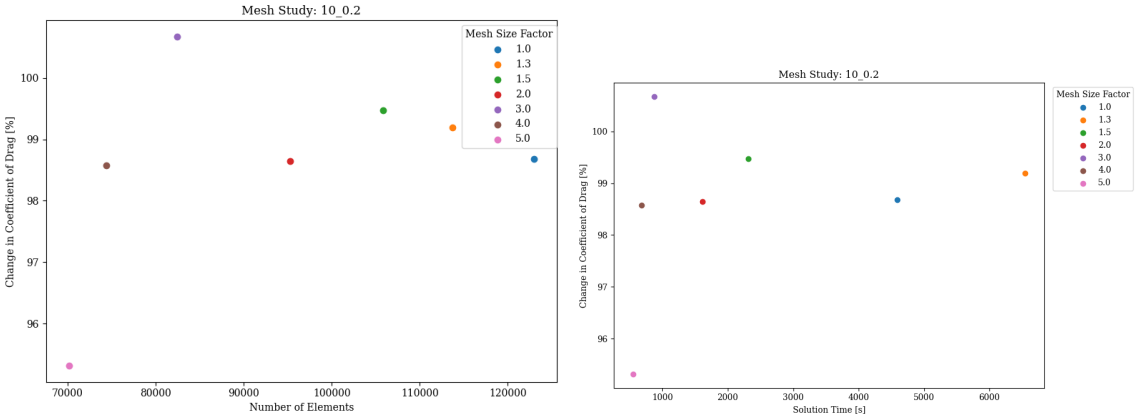

Mesh Study: 5.05_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
2250
2000
1750
1500
1250
1000
750
500
70000
80000
90000
100000
110000
120000
Number of Elements

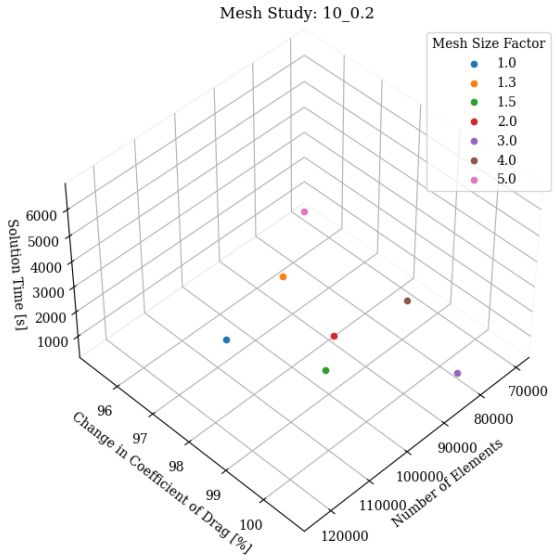

Mesh Study: 5.05_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.140
0.135
0.130
0.125
0.120
70000
80000
90000
100000
110000
120000
Number of Elements

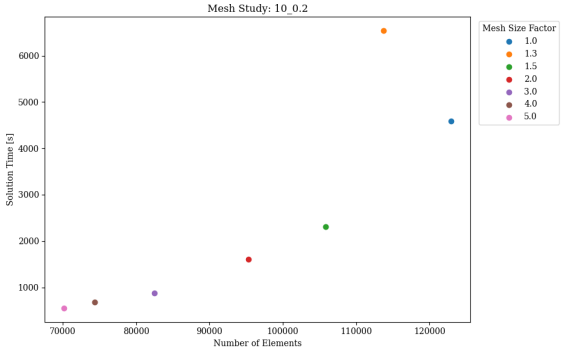

Mesh Study: 5.05_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.140
0.135
0.130
0.125
0.120
500
750
1000
1250
1500
1750
2000
2250
Solution Time [s]

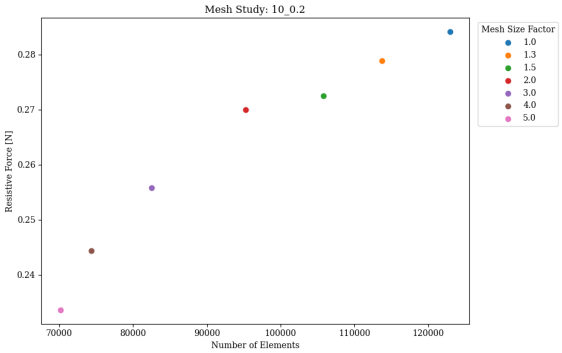


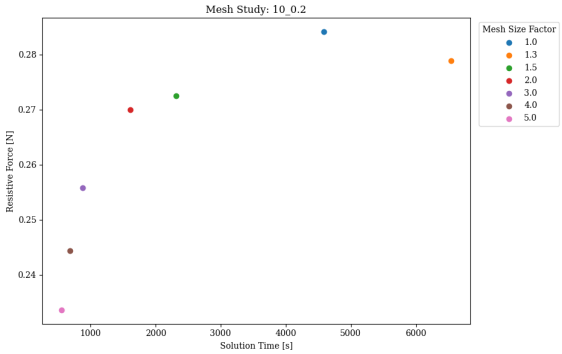
Boundary Case: $\omega_{amp} = 10,\ f = 0.2$

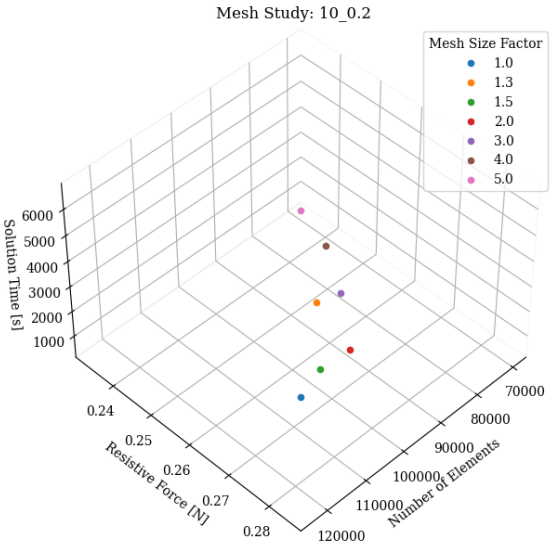



Mesh Study: 10_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
6000
5000
4000
3000
2000
1000
Change in Coefficient of Drag [%]
96
97
98
99
100
Number of Elements
70000
80000
90000
100000
110000
120000

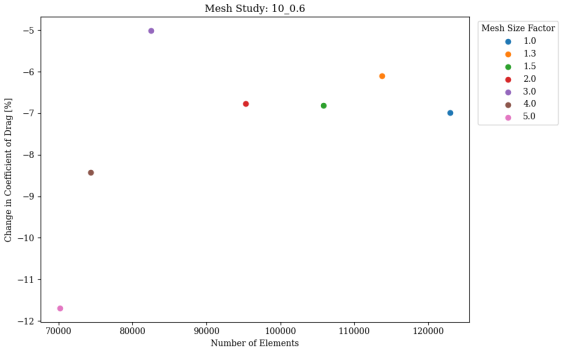

Mesh Study: 10_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
6000
5000
4000
3000
2000
1000
70000
80000
90000
100000
110000
120000
Number of Elements

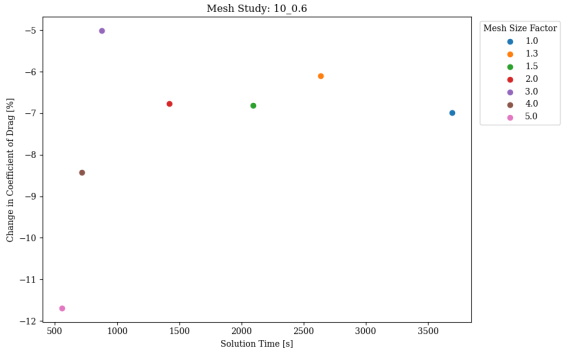

Mesh Study: 10_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.28
0.27
0.26
0.25
0.24
70000
80000
90000
100000
110000
120000
Number of Elements

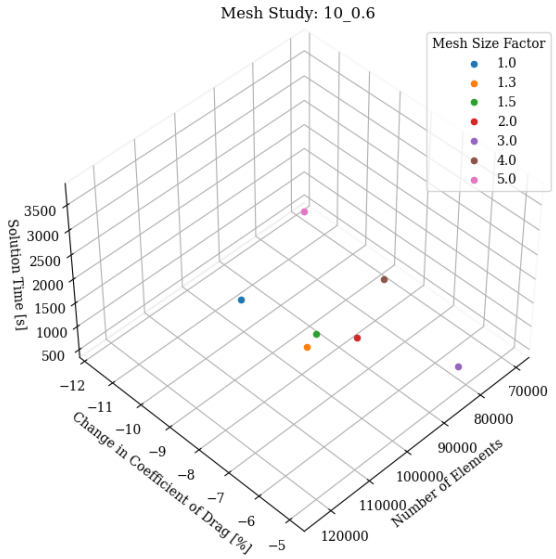

Mesh Study: 10_0.2
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.28
0.27
0.26
0.25
0.24
1000
2000
3000
4000
5000
6000
Solution Time [s]

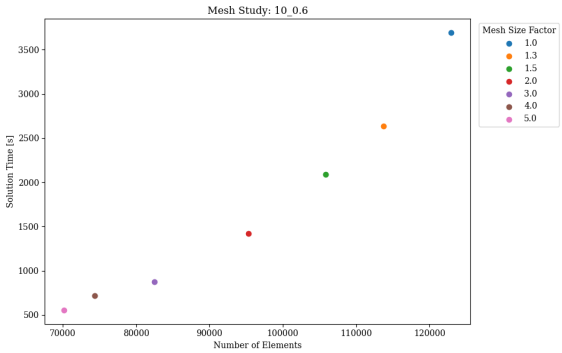


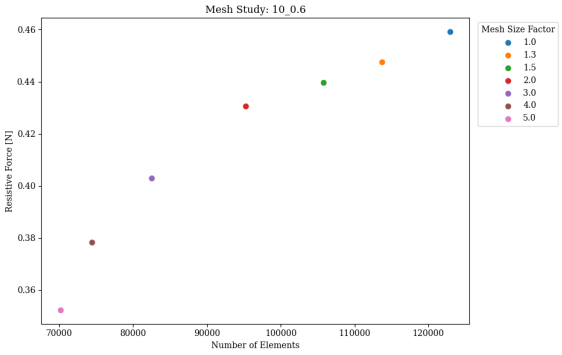
Boundary Case: $\omega_{amp} = 10,\ f = 0.6$

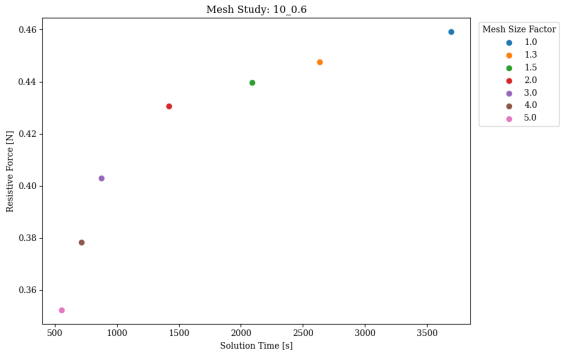


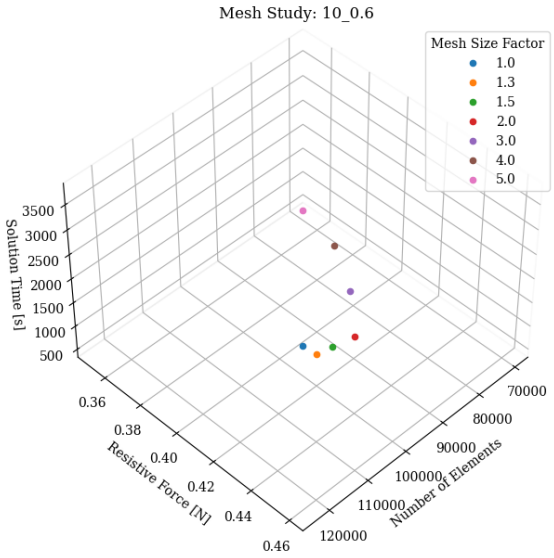



Mesh Study: 10_0.6
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
3500
3000
2500
2000
1500
1000
500
Change in Coefficient of Drag [%]
−12
−11
−10
−9
−8
−7
−6
−5
Number of Elements
70000
80000
90000
100000
110000
120000

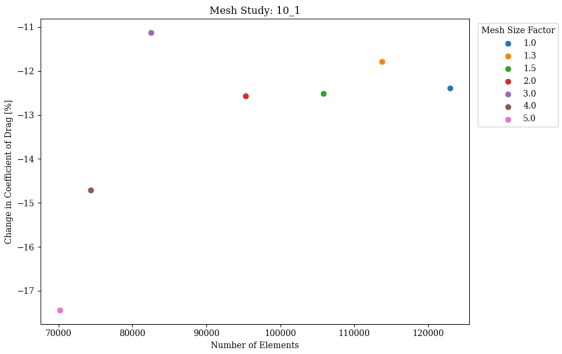

Mesh Study: 10_0.6
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
3500
3000
2500
2000
1500
1000
500
70000
80000
90000
100000
110000
120000
Number of Elements

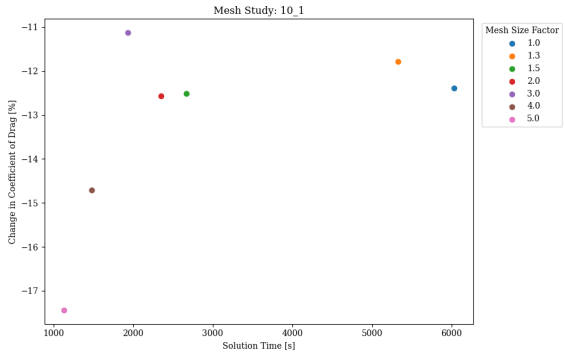

Mesh Study: 10_0.6
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.46
0.44
0.42
0.40
0.38
0.36
70000
80000
90000
100000
110000
120000
Number of Elements

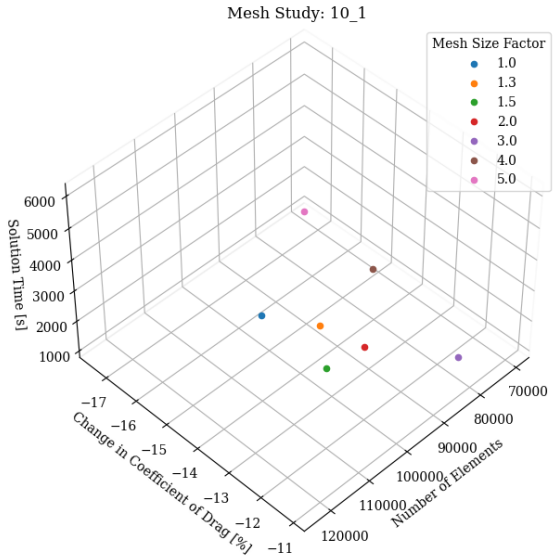

Mesh Study: 10_0.6
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.46
0.44
0.42
0.40
0.38
0.36
500
1000
1500
2000
2500
3000
3500
Solution Time [s]

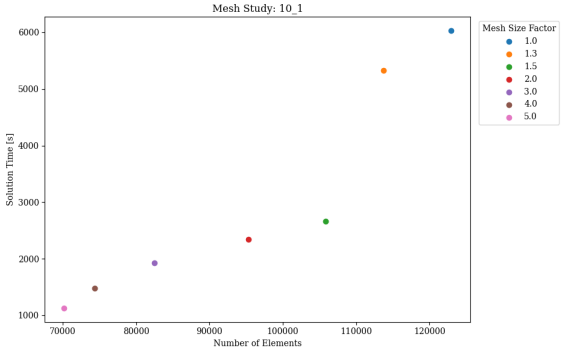


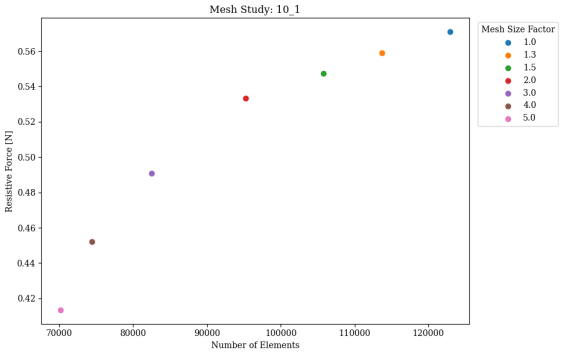
Boundary Case: $\omega_{amp} = 10,\ f = 1.0$

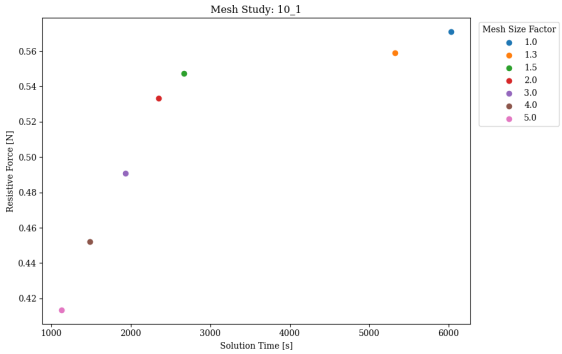


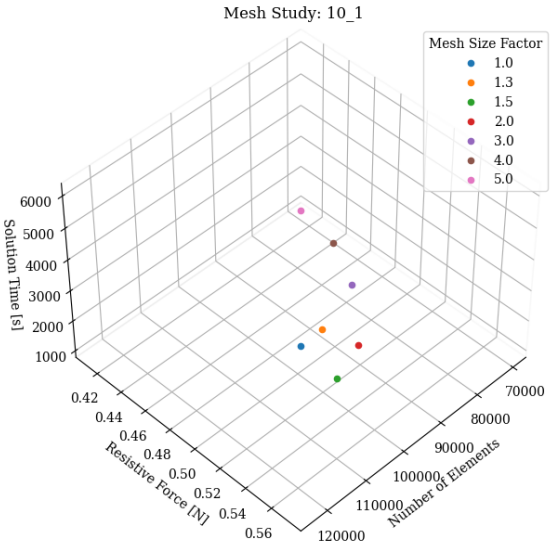



Mesh Study: 10_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
6000
5000
4000
3000
2000
1000
Change in Coefficient of Drag [%]
−17
−16
−15
−14
−13
−12
−11
Number of Elements
70000
80000
90000
100000
110000
120000

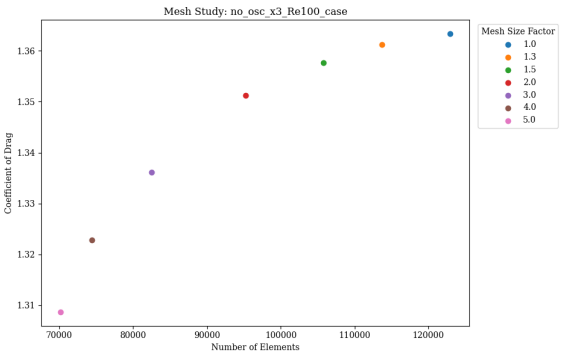

Mesh Study: 10_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Solution Time [s]
6000
5000
4000
3000
2000
1000
70000
80000
90000
100000
110000
120000
Number of Elements

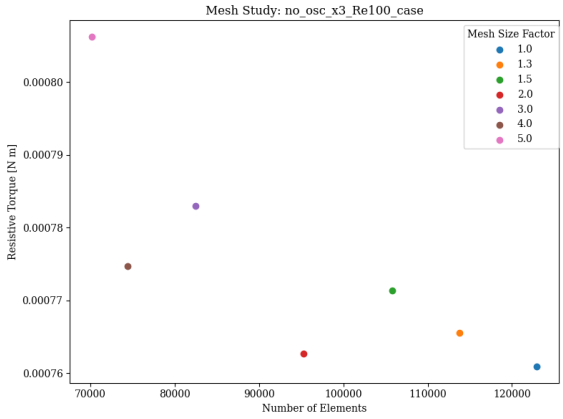

Mesh Study: 10_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.56
0.54
0.52
0.50
0.48
0.46
0.44
0.42
70000
80000
90000
100000
110000
120000
Number of Elements


Mesh Study: 10_1
Mesh Size Factor
1.0
1.3
1.5
2.0
3.0
4.0
5.0
Resistive Force [N]
0.56
0.54
0.52
0.50
0.48
0.46
0.44
0.42
1000
2000
3000
4000
5000
6000
Solution Time [s]

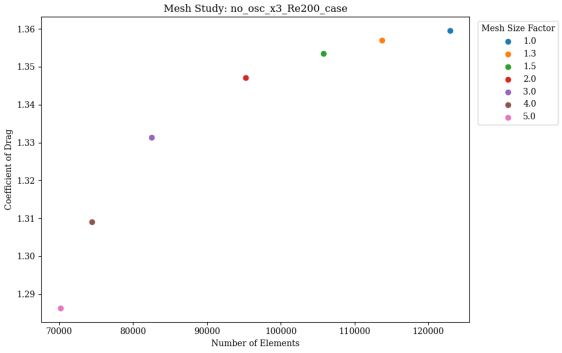


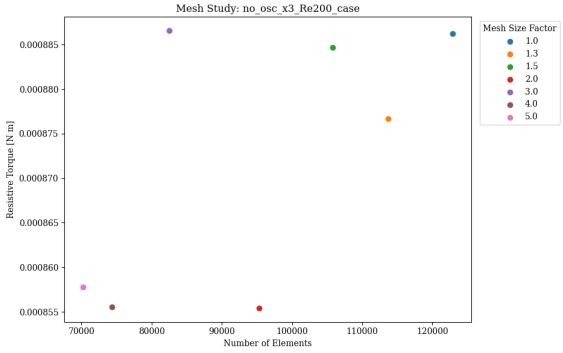
No Oscillation Case: Re = 100



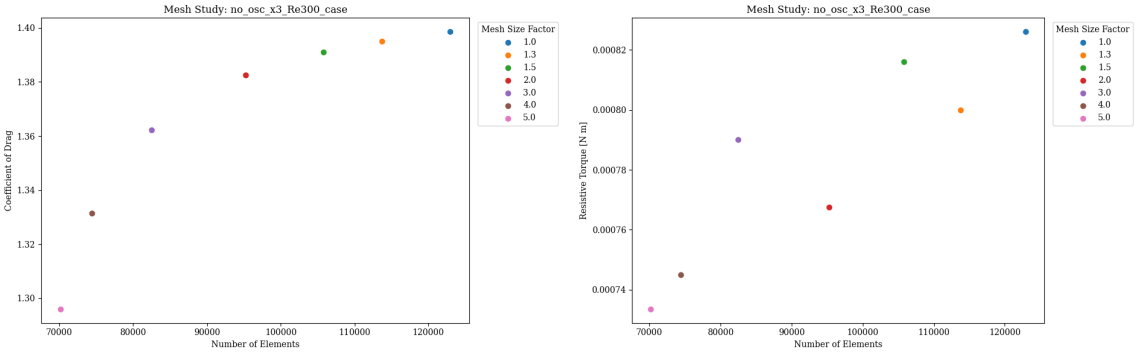


No Oscillation Case: Re = 200

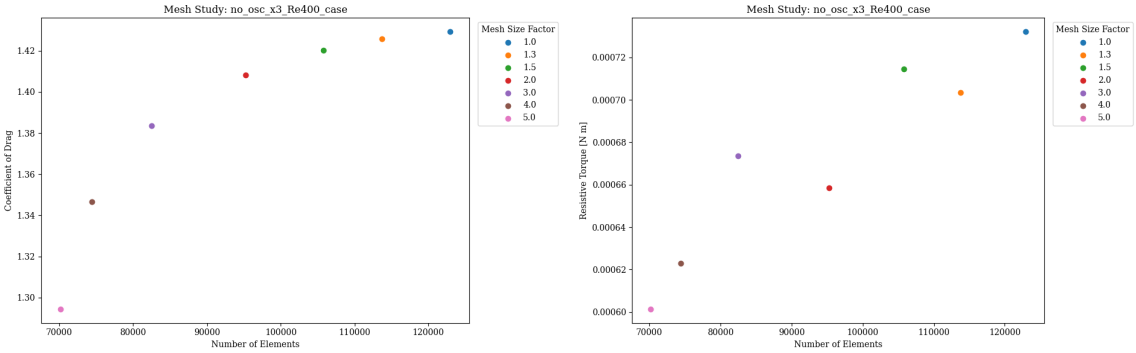




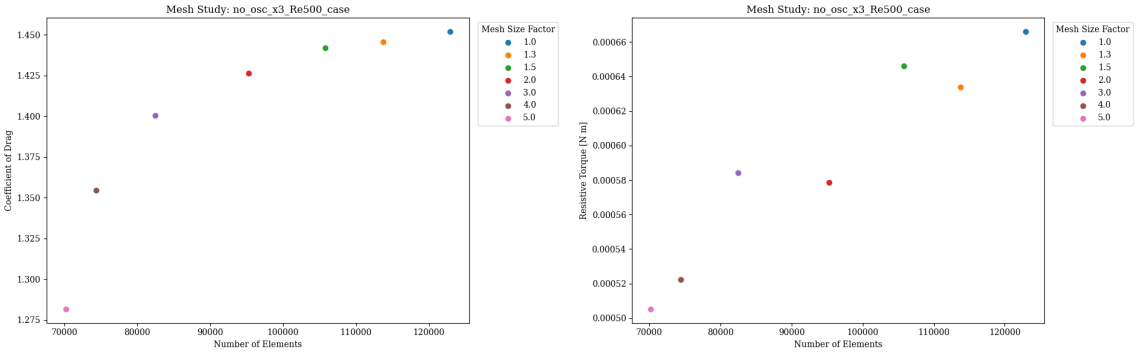
No Oscillation Case: Re = 300



No Oscillation Case: Re = 400



No Oscillation Case: Re = 500



5.1.2 RANS Jet Simplification

Test Case

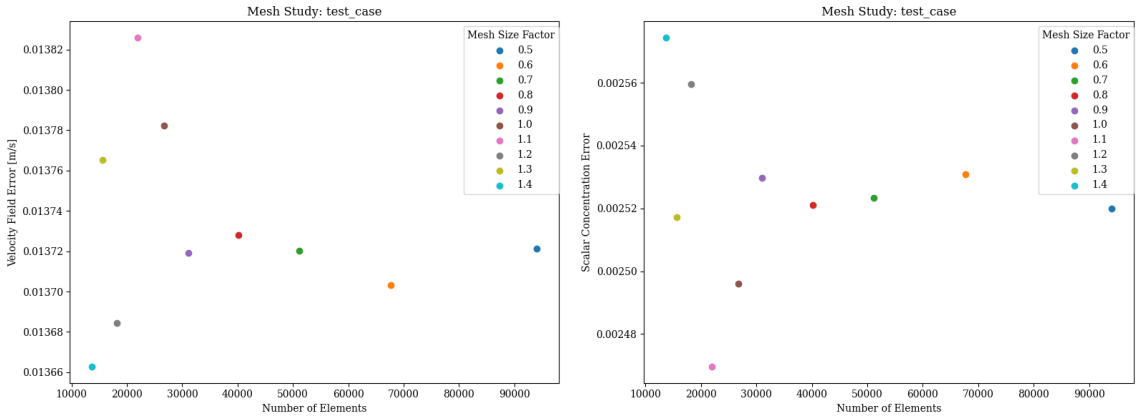


Figure 5.1: Test case mesh study. Breath velocity, $u_{breath} = 0.35\,[\frac{m}{s}]$, diameter of the circular mouth opening, $D_{mouth} = 0.021[m]$.

Boundary Cases

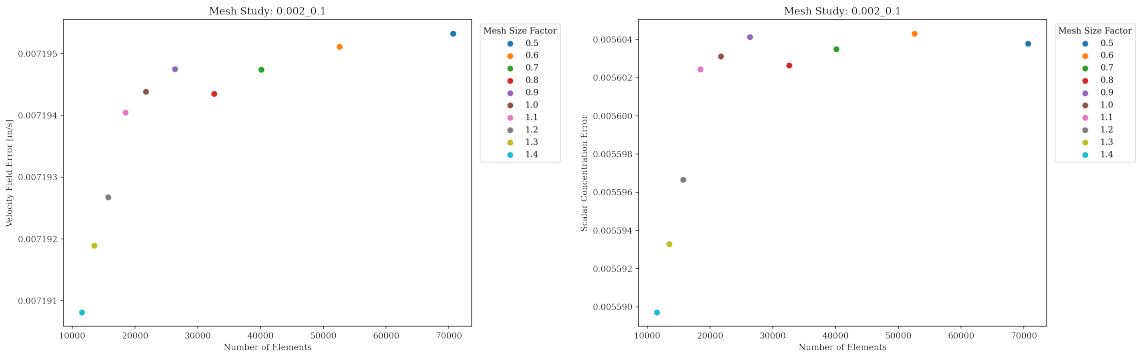


Figure 5.2: Boundary case mesh study. Breath velocity, $u_{breath} = 0.1\,[\frac{m}{s}]$, diameter of the circular mouth opening, $D_{mouth} = 0.002[m]$.

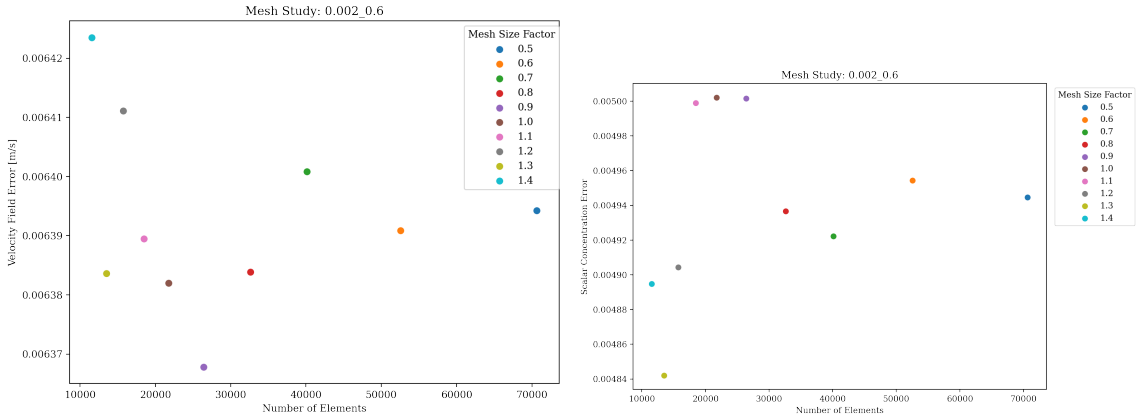


Figure 5.3: Boundary case mesh study. Breath velocity, $u_{breath} = 0.6\,[\frac{m}{s}]$, diameter of the circular mouth opening, $D_{mouth} = 0.002[m]$.

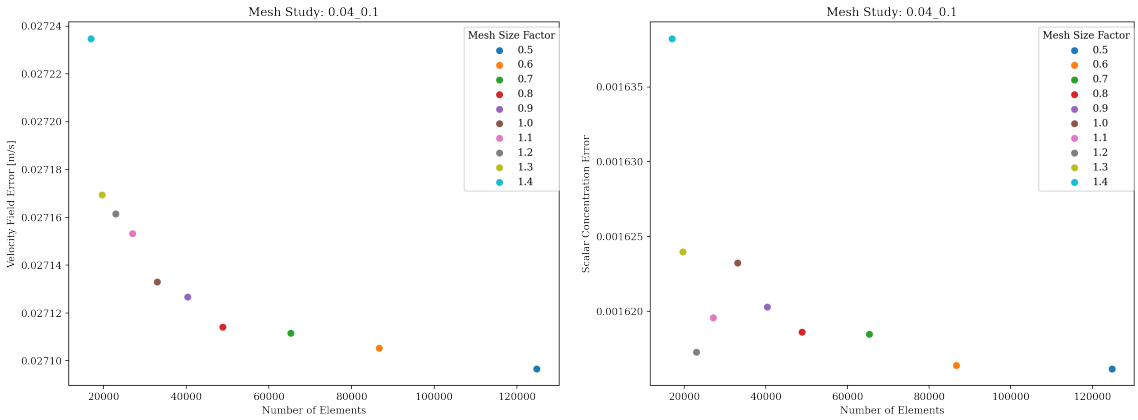


Figure 5.4: Boundary case mesh study. Breath velocity, $u_{breath} = 0.1\,[\frac{m}{s}]$, diameter of the circular mouth opening, $D_{mouth} = 0.04[m]$.

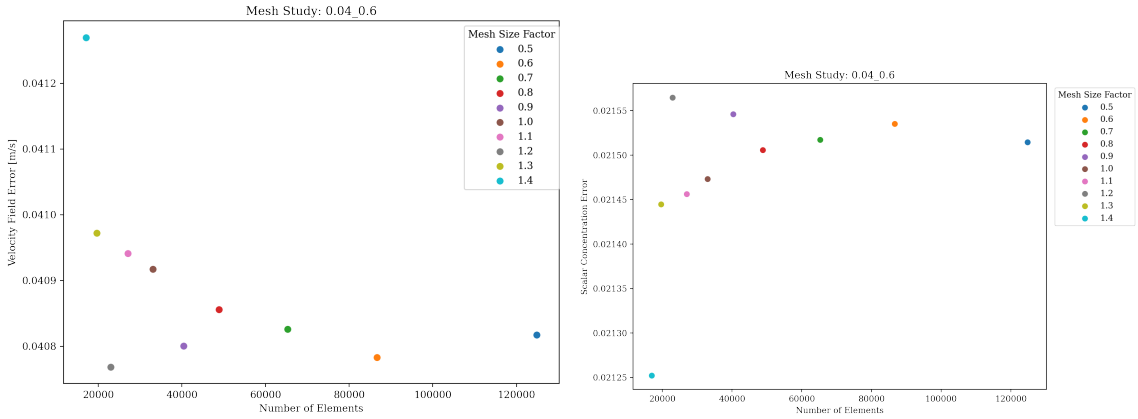


Figure 5.5: Boundary case mesh study. Breath velocity, $u_{breath} = 0.6\,[\frac{m}{s}]$, diameter of the circular mouth opening, $D_{mouth} = 0.04[m]$.

5.2 Optimization Studies

5.2.1 Oscillating Cylinder

Convergence of Optimum Objective Means

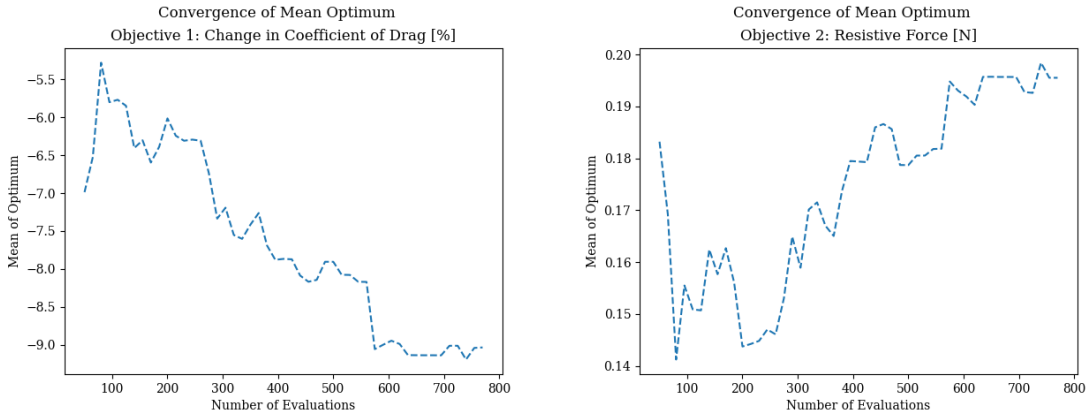


Figure 5.6: Oscillating Cylinder Optimization Study - Mean of Optimum Convergence

Boundary Cases: Parameter and Objective Spaces

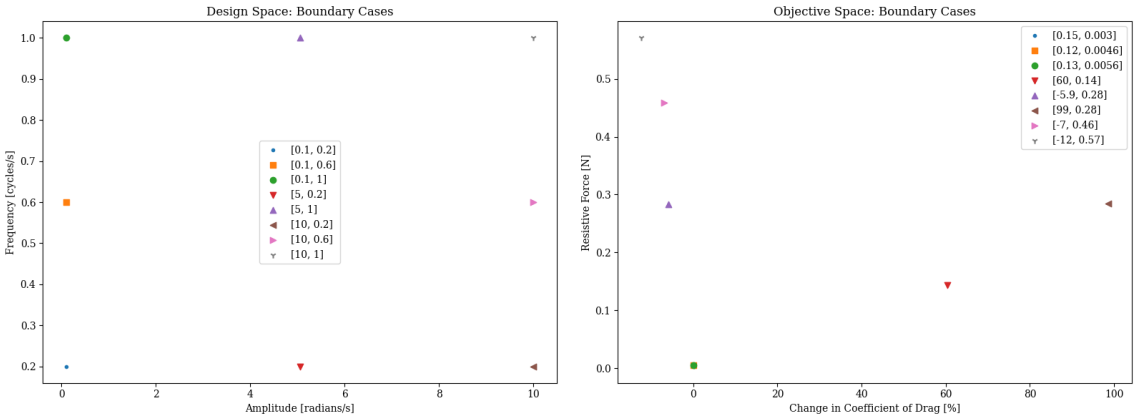


Figure 5.7: Oscillating Cylinder Optimization - Boundary cases parameter and objective spaces.

Generation 1: Parameter and Objective Spaces

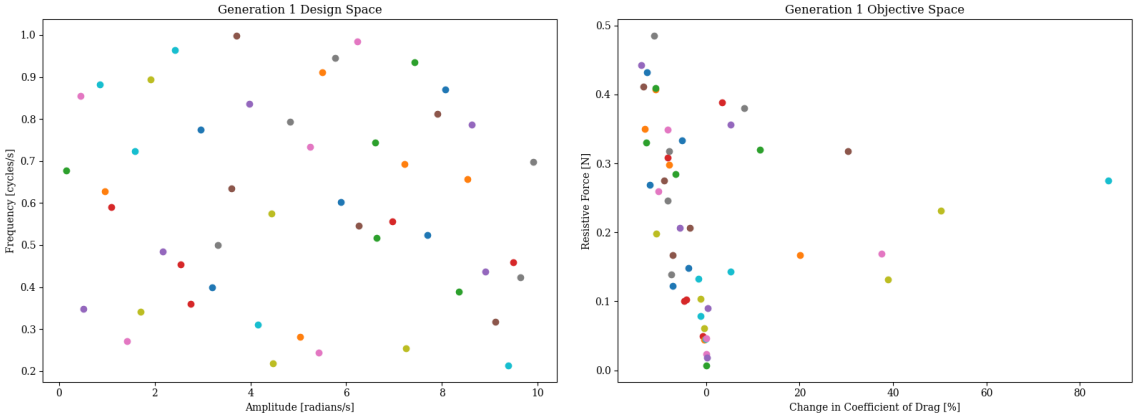


Figure 5.8: Oscillating Cylinder Optimization - Generation 1.

Map Generation 1 Population

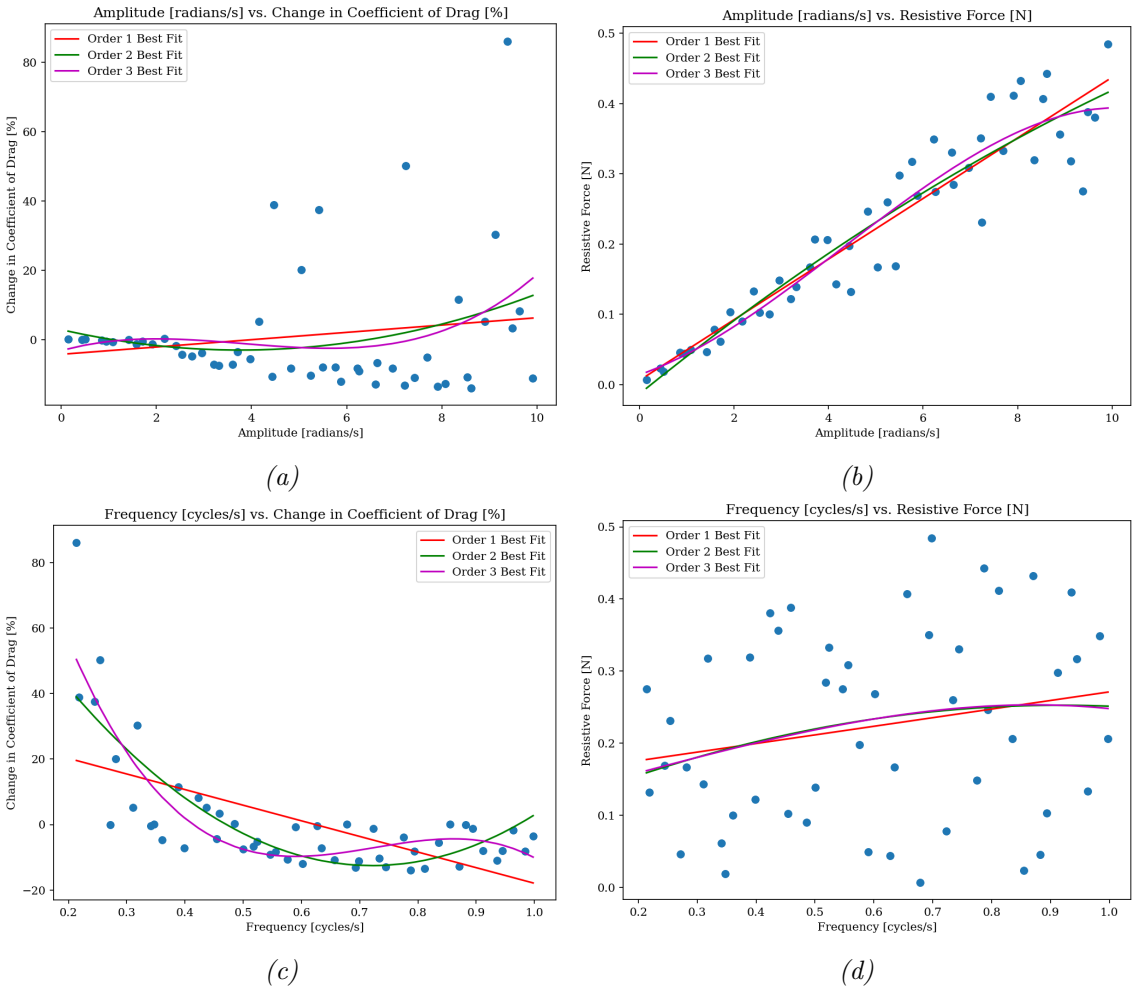


Figure 5.9: Parameters versus objectives for generation 1; which is a Latin hypercube sampling of the entire parameter space. First through third order best fit lines plotted as well.

Final 10 Generations: Parameter and Objective Spaces

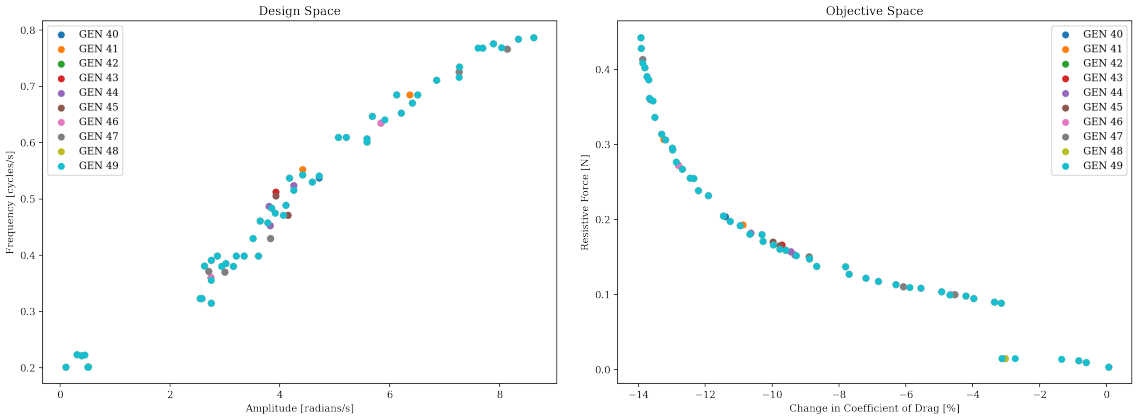
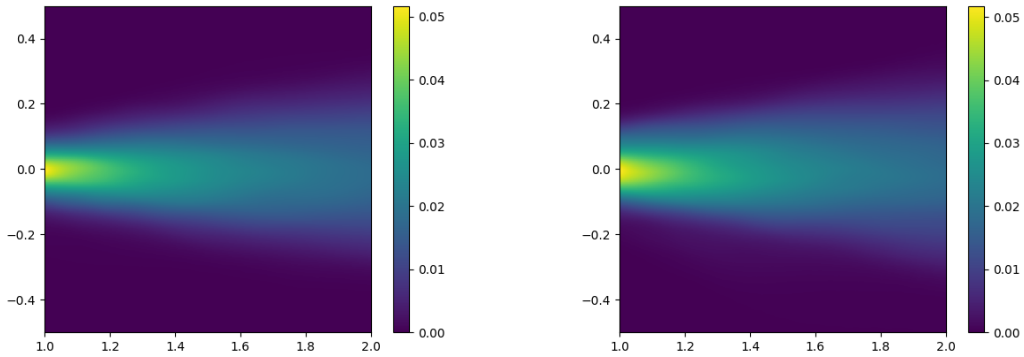


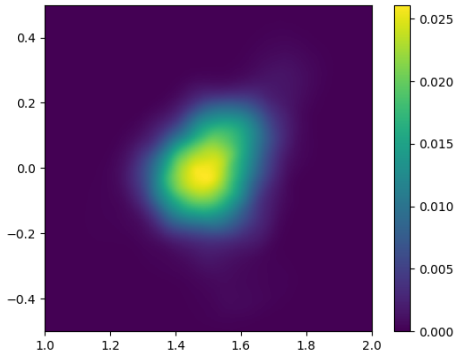
Figure 5.10: Oscillating Cylinder Optimization - Generations 40 through 50.

5.2.2 Jet Simplification



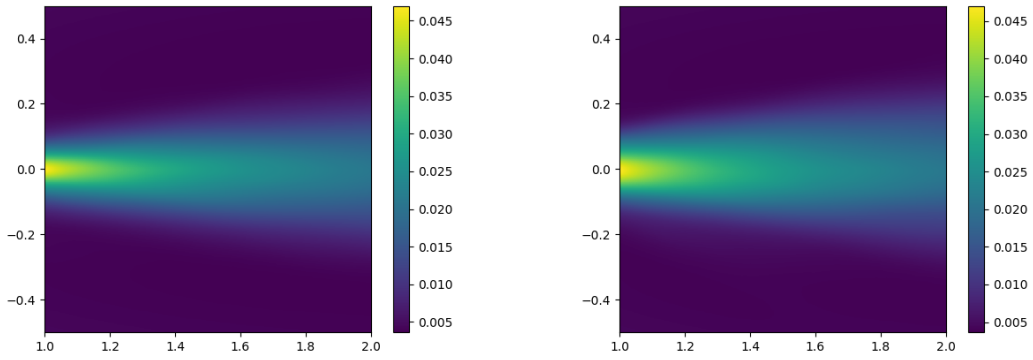
(a) x-y slice at z=0

(b) x-z slice at y=0



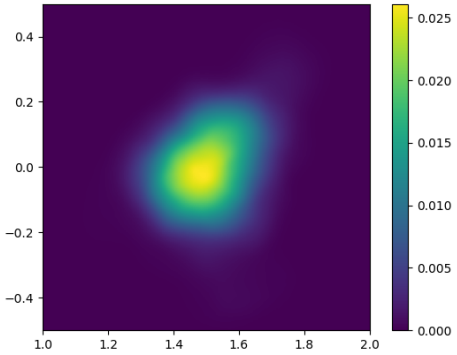
(c) y-z slice at x=1.5m

Figure 5.11: RANS Jet Optimization - Flow field slices showing mass fraction of passive scalar, ϕ. YALES2 LES simulation of a 3D jet interpolated onto a 200x200x200 cell grid using python library scipy.interpolate.griddata() function.



(a) x-y slice at z=0

(b) x-z slice at y=0



(c) y-z slice at x=1.5m

Figure 5.12: Flow field slices showing velocity magnitude. YALES2 LES simulation of a 3D jet interpolated onto a 200x200x200 cell grid using python library scipy.interpolate.griddata() function.

Final Generation

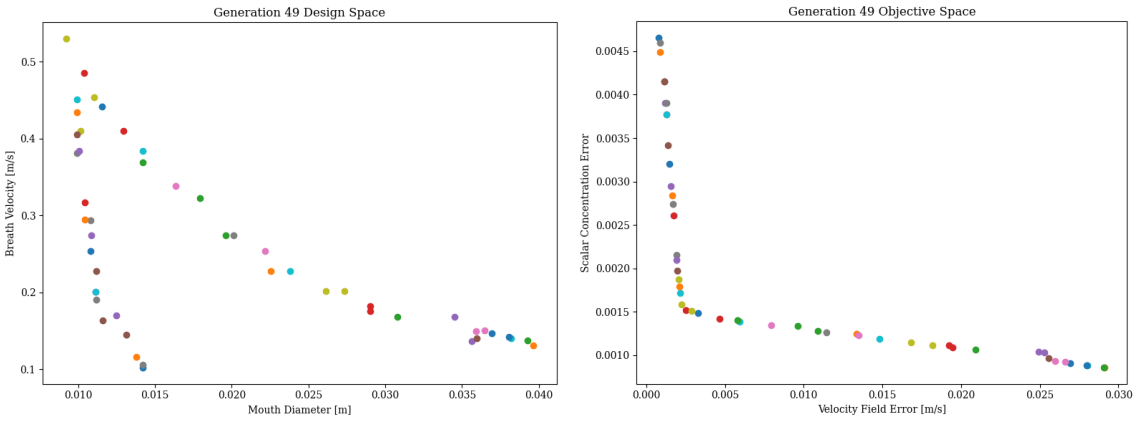


Figure 5.13: RANS Jet Optimization Study - Final Generation, Generation 50

Final Optimum

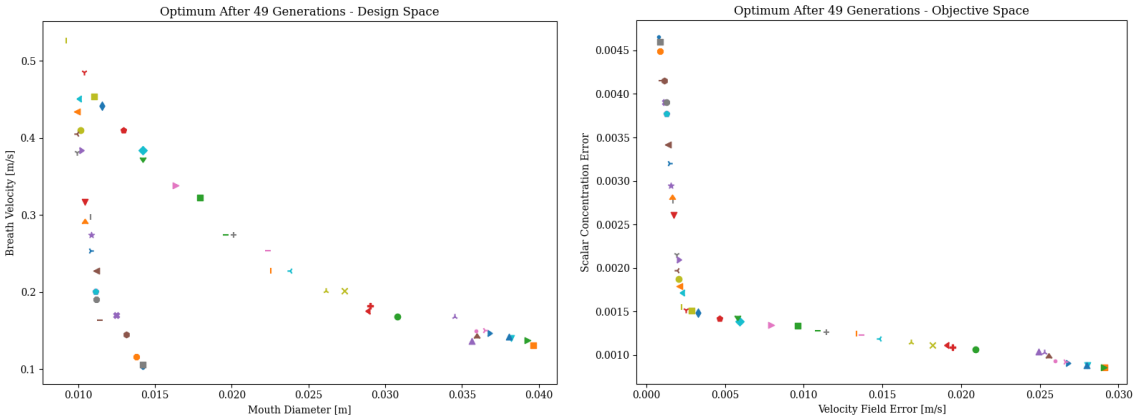
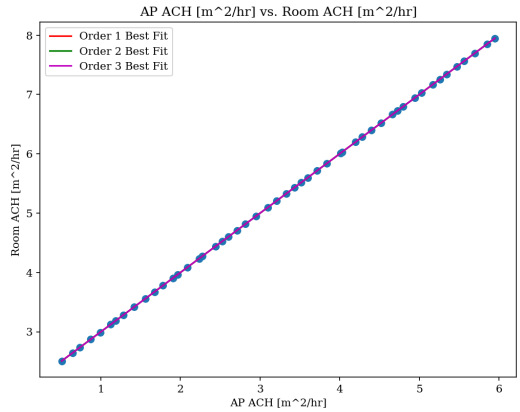


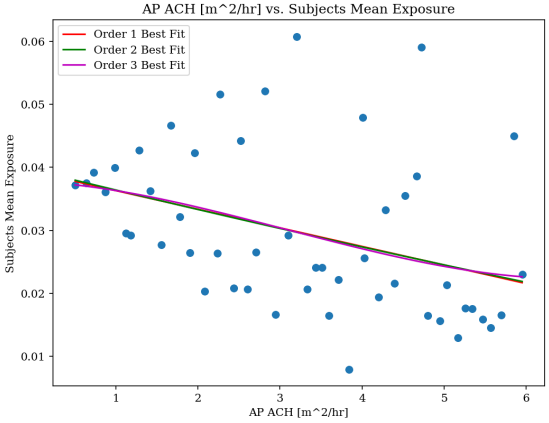
Figure 5.14: RANS Jet Optimization Study - Optimum After 50 Generations

5.2.3 Air Purifier Configuration

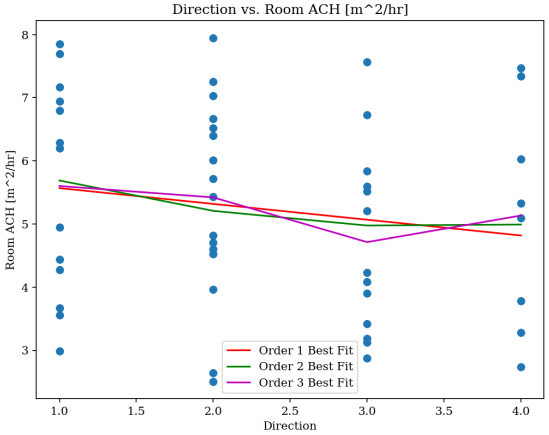
Mapping generation 1 design space versus objective space. Generation 1 is made up of 50 individuals sampled using a Latin hyper-cube distribution across entire parameter space.


AP ACH [m^2/hr] vs. Room ACH [m^2/hr]
Order 1 Best Fit
Order 2 Best Fit
Order 3 Best Fit
Room ACH [m^2/hr]
AP ACH [m^2/hr]

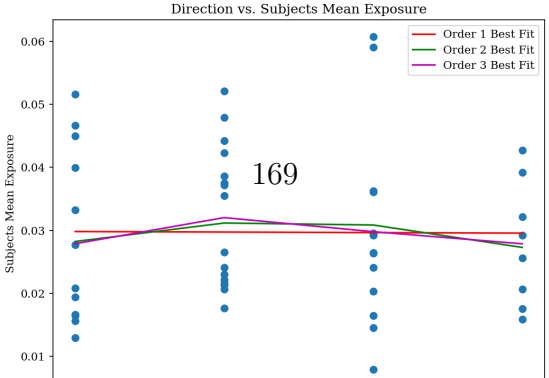
(a)


AP ACH [m^2/hr] vs. Subjects Mean Exposure
Order 1 Best Fit
Order 2 Best Fit
Order 3 Best Fit
Subjects Mean Exposure
AP ACH [m^2/hr]

(b)


Direction vs. Room ACH [m^2/hr]
Order 1 Best Fit
Order 2 Best Fit
Order 3 Best Fit
Room ACH [m^2/hr]
Direction

(c)


Direction vs. Subjects Mean Exposure
Order 1 Best Fit
Order 2 Best Fit
Order 3 Best Fit
Subjects Mean Exposure

Final Generation (Generation 50)

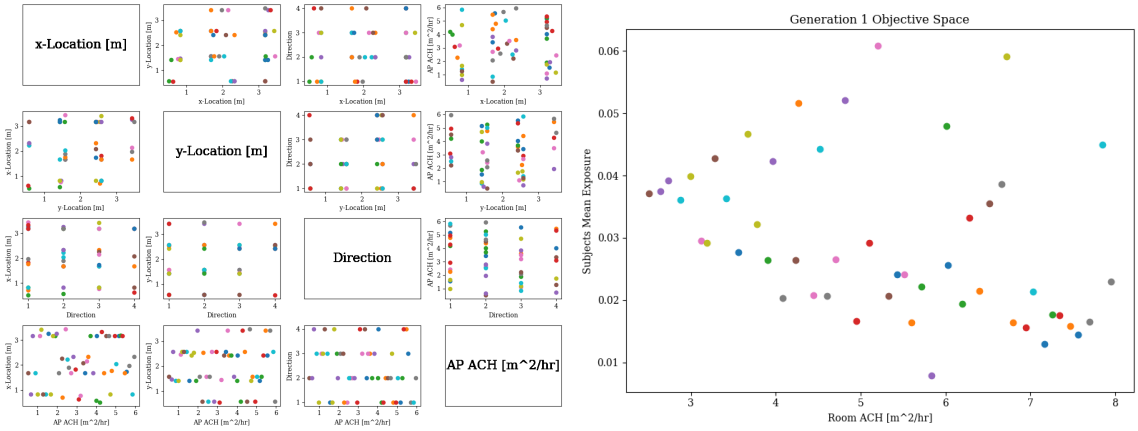


Figure 5.16: Air Purifier Configuration Optimization Study - First Generation - Design and Objective Spaces.

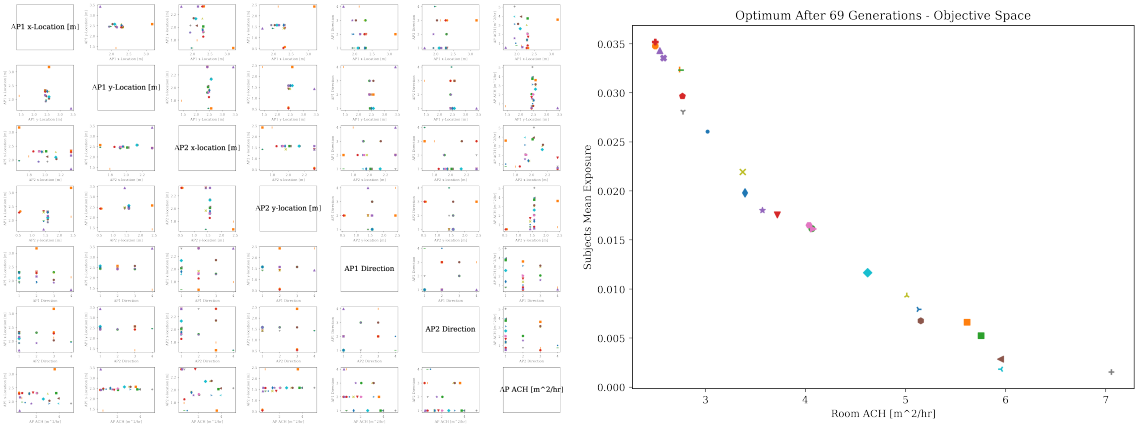


Figure 5.17: Two Air Purifier Configuration Optimization Study - Final Generation Optimum - Generation 69

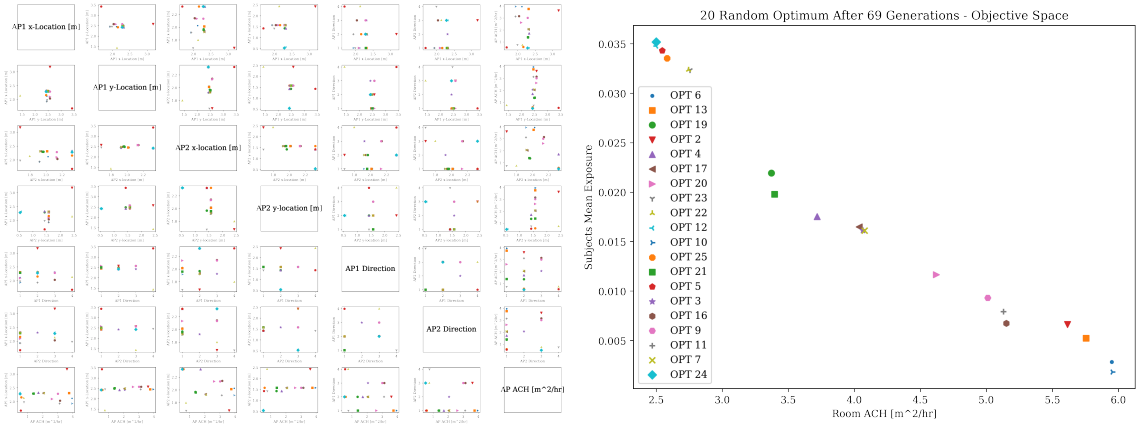


Figure 5.18: Two Air Purifier Configuration Optimization Study - Final Generation 20 Optimum - Generation 69

5.2.4 Code

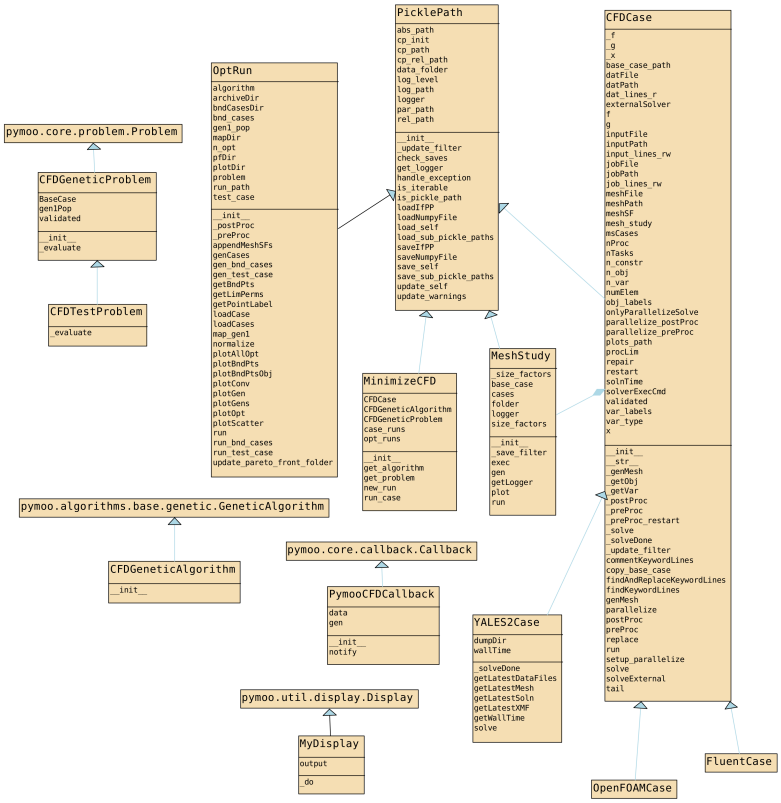


Figure 5.19: pymooCFD.core Unified Modeling Language Class Diagram. Modules within pymooCFD.core: pymooBase, cfdCase, optRun, and meshStudy.

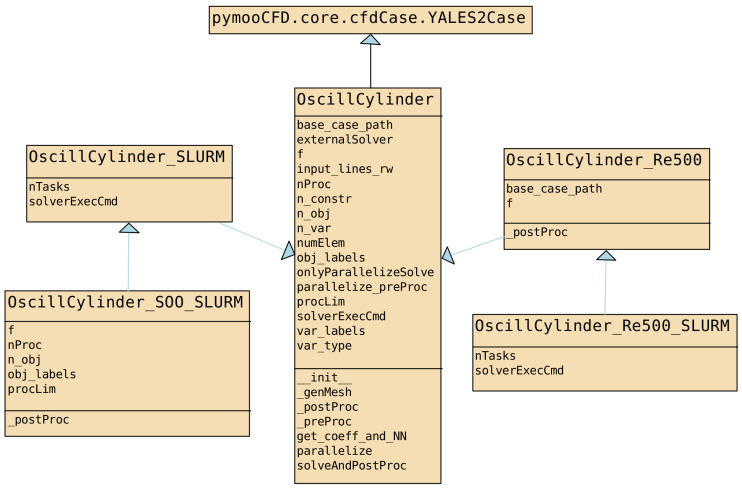


Figure 5.20: pymooCFD.problem.oscill_cyl Unified Modeling Language Class Diagram. Oscillating Cylinder Problem Module

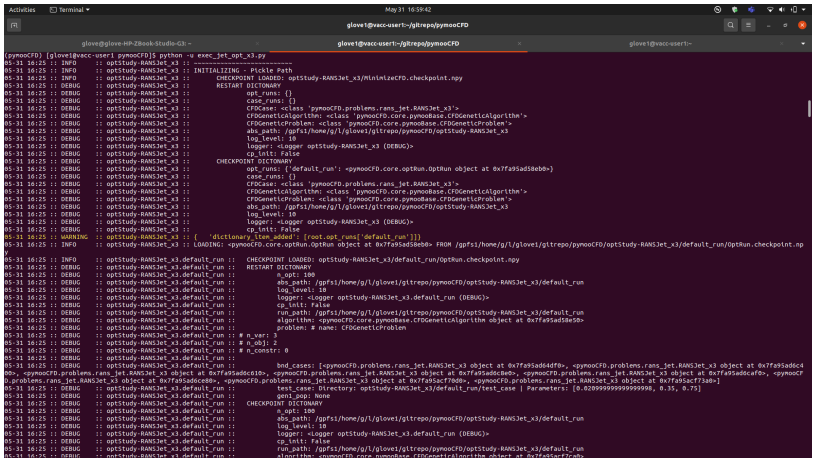


Figure 5.21: Screenshot of pymooCFD in use.

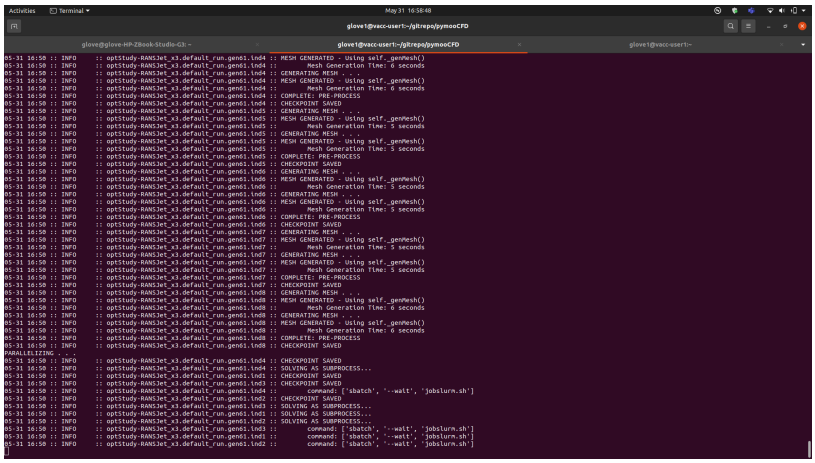


Figure 5.22: Screenshot of pymooCFD in use.

Please see https://github.com/gmclove/pymooCFD for the full codebase.