THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF MECHANICAL ENGINEERING

Optimal Design of Additively Manufactured Fin Topology for a Two-Phase Flow Channel Using Multi-Objective Genetic Algorithms

ANTHONY TUONG LUU
SPRING 2022

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Mechanical Engineering
with honors in Mechanical Engineering

Reviewed and approved* by the following:

Mary I. Frecker
Department Head of Mechanical Engineering
Thesis Co-Supervisor

Matthew J. Rau
Assistant Professor of Mechanical Engineering
Thesis Co-Supervisor

Anne E. Martin
Professor of Mechanical Engineering
Honors Adviser

* Electronic approvals are on file.

ABSTRACT

Through development of additive manufacturing, objects of certain materials are now able to be designed with increased complexity. As a result, applications in which traditional structures consisting of simple geometries can now be re-examined and optimized. This study examines a two-phase flow heat exchanger motivated by a potential of increased performance as well as lack of research present in the field. Specifically, this study examines the optimal design characteristics of an annularly finned heat exchanger in a uniformly cooled crossflow environment.

To optimize the design of a heat exchanger for competing objectives, heat transfer and pressure drop, a multi-objective genetic algorithm is employed. This methodology was selected due to the low number of variables used, its likelihood to converge to the global optima, its ability to support multiple objectives, and its ease of implementation in MATLAB. A compact heat exchanger correlation was used to evaluate the pressure drop of the system, and a flat plate correlation was used in combination with a cylinder in cross flow correlation to evaluate the heat transfer of the system.

This study is organized into 3 case studies that investigates the optimal designs of the heat exchanger with different aspects of design freedom. The results of these case studies have revealed that different variables have a larger impact than others when selecting the optimal design for a user's specified performance. It is also found that increasing the design freedom considered does not have a significant impact on performance. Additionally, it was found that introducing a temperature profile does not have an effect on the placement of fin mass along a channel. Finally, it was observed that for various levels of constant pressure drop, the total heat

transfer of the system was not affected by the mass flux of a surrounding fluid. Despite these findings, interest in introducing additional design variables for a more optimal performance and unrestricting experimental constraints for a wider range of results provides interest in future work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am extremely grateful to have gotten the opportunity to work with an amazing research team on this complex and interesting project during the development of my thesis. Dr. Mary Frecker and Dr. Matthew Rau have provided me with all the support I needed throughout my time as a Schreyer scholar, not only within the aspects of this project, but academically and professionally, and I could not be more thankful. I would also like to thank Nicholas Evich, a colleague and previous Schreyer scholar, who has helped guide me as I struggled throughout this project and acted as a great think partner. Through the contributions of my amazing research team over the past year and a half, I have gained much more confidence in my ability as a researcher and am excited to see how my experience working on this project makes me a better engineer in the future.

I would also like to thank my family and partner, Meghan, for supporting me throughout my undergraduate career. By being there for me both emotionally and financially, I felt confident that I could succeed professionally, academically, and as a leader. Without them, I would not be anywhere near the person I am today, and I am grateful for the part they played in me getting where I am.

Chapter 1

Literature Review

Introduction, Context, and Motivation

Additive manufacturing is the process of creating products by building them layer by layer. Because material is added layer-wise, additive manufacturing allows engineers to create products with much more design freedom while reducing waste as opposed to traditional methods like subtractive manufacturing. With increased design freedom, possibilities for the optimization of designs unrestricted by geometric constraints are now available. The type of additive manufacturing that will specifically be utilized in this study is laser metal powder bed fusion because it is amenable to metals of interest for cooling applications.

Currently, the design of thermal management systems and heat exchangers consist of engineers employing the use of extended surfaces or fins to sufficiently achieve the amount of heat transfer necessary to cool a fluid or a heat generating object. However, designs of these systems have been relatively limited in shape due to manufacturing constraints imposed by traditional manufacturing processes. Two competing objectives are often desired by engineers when designing thermal management systems. Maximizing the heat transfer by the device is desired to efficiently cool a heat generating object. However, engineers also aim to minimize the pressure drop across the system because the pressure drop corresponds to the amount of pumping power needed to cool the device. This study analyzes a system in which a channel is cooled through externally forced convection from a fan. For an air-side application, the pumping power exhibited by the fan is the product of the mass flow rate and pressure drop of the flow. This

relationship is also highlighted in fan curves which illustrates the dependence between the pumping power of a fan and the pressure drop of the flow [1]. An engineer is often able to achieve a lower pressure drop at the expense of the heat transfer and vice versa. By applying design optimization and additive manufacturing to these devices, unique shapes can be generated in which the tradeoff between heat transfer and pressure drop are optimized.

An additional consideration in the design of thermal management systems is the type of flow that travels through it. Single-phase flows are those that remain as a liquid or gas as they travel through a system. Conversely, two-phase flows feature a flow that transitions from liquid to gas due to evaporation. The interest in two-phase flows arises due to its ability to better transfer heat compared to single-phase flows [2]. However, the complex behavior of two-phase flows has made it difficult for researchers to accurately model their thermohydraulic performance causing the amount of research in this field to be relatively limited.

A previous study [3] investigated the thermodynamic properties of an optimized additively manufactured two-phase channel with nonuniform cross sections. In his study, Evich used a multi-objective genetic algorithm to create a set of optimal designs for a given two-phase flow. This work was intended to contribute to the development of a design tool that efficiently produces a set of optimal designs for two-phase flows given a design space and fluid properties. Currently, no design tools exist for two-phase flows due to their complex nature [3]. Although creating a fully functional design tool for two-phase flows is out of the scope of this project, the proposed project will seek to apply a multi-objective genetic algorithm optimization to the design of external annular fins on the surface of a non-uniform two-phase flow channel. Specifically, this study will investigate the effects of fin geometry on the convection heat transfer and the pressure loss of an external cooling fluid. Although the flow inside the channel will

consist of a two-phase fluid for increased heat transfer properties, the flow outside of the channel cooling the fins will be a single-phase flow such as water or air. At the end of development, the program created in this study will be incorporated into an optimization program currently in development for two-phase channels with nonuniform cross sections. Together, these design optimization codes will allow for the design of non-traditional evaporative heat exchangers that can be made with additive manufacturing. Although the purpose of developing this algorithm is intended for a two-phase flow heat exchanger design tool, the investigation of this study can be expanded to the design of annular fins for many other applications.

The use of fins or extended surfaces is a common method for cooling and heating applications because, by increasing the exposed area of a specimen to a fluid, the amount of convection on the specimen increases [2]. Traditional fins and extended surfaces consist of simple geometries for ease of manufacturing. In Figure 1, traditional fin and extended surface configurations are shown on the top and bottom, respectively. Although these geometries are currently prevalent in industry, by applying additive manufacturing, complex geometries and varying channel cross sections can be achieved. Although traditional designs accomplish the goal of increasing heat transfer to the surrounding fluid, this often comes at the expense of inefficiently increasing the pumping power needed to circulate the fluid across the thermal management system. As a result, this study investigates the use of design optimization tools to create a set of fin designs that best maximizes the heat transfer performance of the channel and minimizes the pressure loss across an external fluid as it travels through the system.

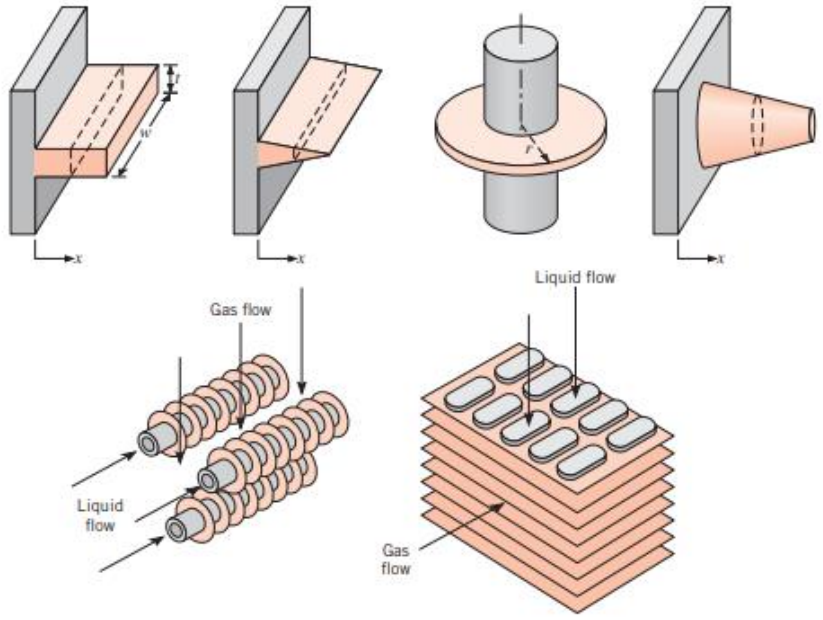

Figure 1. Schematic of typical fin configurations (top) and extended surfaces (bottom) used to increase heat transfer [2].

Introduction to Fins and Relevant Research

Fins or extended surfaces are common tools used by designers to enhance heat transfer between a solid with a surrounding fluid. Fins can come in all shapes and sizes depending on the application. Specifically, an annular fin, the type investigated in this study, is one that is attached circumferentially to a cylindrical channel [2]. Although the shape of annular fins is traditionally circular in shape with a constant thickness, using additive manufacturing allows designers to recreate these fins into virtually any desired shape. As a result, this study aims to investigate the optimal design of annular fins of varying length, thickness, and spacing.

Research on fins in single-phase flows is extensive due to their wide range of applications. For example, a study conducted by L. H. Tang *et al* [4]. investigated the air-side performance of fin and tube heat exchangers with five different patterns: crimped spiral fins, plain fins, slit fins, vortex generator fins, and mixed fins. Similar to this study, heat transfer and pumping power were the objective functions, and they concluded that, based on the application, certain designs had better thermohydraulic performances compared to the others. This study provides evidence that, depending on the thermal properties of a channel application, the optimal design for external fins should vary.

Another study conducted by Sebastian Unger *et al.* [5] investigated the thermohydraulic performance of additively manufactured fins compared to a plain annular fin. The two additively manufactured shapes used in this study were circular integrated pin fins and serrated integrated pin fins. It was found that the circular integrated pin fins were better than the other two designs if heat transfer performance was the only consideration. However, the serrated integrated pin fins were better than the other designs if physical properties such as cost, surface area, and weight

were objectives. This study provides evidence that the best fin design can also vary based on the relative importance of the heat transfer and pressure drop.

To calculate the thermohydraulic parameters in this study, convection correlations and/or models for heat transfer and pressure drop must be used. Correlations are equations for variables that were found experimentally under a set of given assumptions and conditions. As such, it is important to build a database of correlations of wide ranges of validity to increase the versatility of the proposed design tool. Kays and London [6] compiled a collection of experimental data for various conditions in a textbook where relevant correlations for this study will be utilized. Specifically, the correlation that will be used in this study is a general equation to calculate the pressure drop in a gas-flow heat exchanger application. However, an additional model will be needed to calculate the heat transfer of the system. Because the application in this study utilizes a cylindrical channel with annular fins, various heat transfer models need to be tested for applicability. For example, the annular fins could be treated as flat plates in parallel flow because of their geometry. An additional assumption could be with the channel acting as a cylinder in cross flow. Heat transfer models for both cases are found in Chapter 13 of Bergman *et al's* heat transfer textbook [2]. Unfortunately, there is no single heat transfer model for a cylindrical channel with external annular fins. As a result, initial calculations for the heat transfer of the system will test both cases for applicability.

Overview of Multi-Objective Engineering Optimization

Engineering optimization is the process which seeks to find a single optimal solution based on one objective or a set of optimal solutions based on multiple objective functions. For the application proposed in this study, the design objectives consist of minimizing the pressure loss and maximizing the heat transfer. As a result, applying an optimization process will produce a set of solutions where designers will be able to select a design based on a desired level pressure loss or heat transfer coefficient. This set of optimal solutions is commonly referred to as the Pareto front in the performance space.

The performance space plot is defined as the plot in which competing objective functions, pressure drop and heat transfer for example, are each assigned to an axis. Each point on the plot represents a feasible solution that corresponds to a value of pressure drop and heat transfer. Because engineering optimization tools traditionally serve to minimize objectives, the reciprocal of heat transfer will be plotted. As a result, it is desired to find the points closest to each respective axis to find the designs that represent the best tradeoff between the competing objectives. The most straight forward method of accomplishing this is by performing an exhaustive search of the search space [7]. This is the process of analyzing the problem and manipulating every possible combination of design variables until every solution is numerated. However, in an application where there is so much design freedom and independent variables, this method is often computationally intensive and inefficient. As a result, formal optimization algorithms are often desired to find similar results at significantly less computation time.

Gradient based optimization is a common method that uses derivatives to find optimal solutions [7]. By calculating the derivative of an objective function at a given point, gradient optimization converges towards a desired maximum or minimum of an objective function.

However, a drawback of this process is that it requires calculation of derivatives with respect to each design variable. Additionally, for complex objective functions that feature nonlinear behavior, gradient based methods will often “get stuck” and converge on a local minima or maxima. The derivative approach also becomes problematic when an objective function does not have a distinct solution in which case the program will never converge. In these cases, the computational time would also be extreme. Because gradient-based optimization has the potential of “getting stuck” at a local minimum far from the global minimum or never converging at all, a more efficient optimization method is desired.

Genetic algorithms are a specific form of evolutionary computation that finds optimal solutions in a method that emulates natural selection [7]. A set of solutions is created using random values for independent variables to represent an initial population. Afterwards, a new population is generated using crossover or mutation reproduction operators from the parent population. A fitness function is used to evaluate parent and offspring populations and selects the best portion of the overall population to continue in the optimization process. Each design has a corresponding pressure loss and heat transfer, and the set of designs are evolved to gradually improve both objectives. This process is repeated until a desired spread change between iterations is achieved. The spread change is the user-specified value for how much change a population should experience from one iteration to another until the program converges. Once the spread change criterion is met, the optimal population is presented. By starting with a random population and iteratively improving until only small changes are present, genetic algorithms can reliably find a Pareto-optimal set of solutions close to the actual optimal set of solutions in a fraction of the computation time needed for an exhaustive search. It is the ability to account for

multiple objectives, reduced computational time, and accuracy that has motivated the use of genetic algorithms for optimization of external annular fins in this study.

The specific type of genetic algorithm that will be utilized in this study is the NSGA-II algorithm published by Deb *et al* [8]. This method presents a nondominated multi-objective optimization process that produces a large spread of equally valid solutions. As a result, solutions found by the genetic algorithms can prevent clustering on specific combinations of the objective functions. In turn, the converged population is more likely to represent a wider spread of optimal solutions. The NSGA-II algorithm is embedded in the multi-objective genetic algorithm in MATLAB, which will be utilized in this study.

With every optimization method, design variables must be identified. Fin parameters such as fin height, fin thickness, number of fins, and fin spacing will be investigated. Additionally, constraint functions are defined to keep the search space from being too large. As a result, minimum feature thickness for additive manufacturing and a user inputted control volume will be used to constrain the design variables.

Research Goals and Overview of Thesis

The primary goal of this study is to investigate the optimal design of fins under various conditions and design flexibility. To achieve this goal, a design tool will be developed using the NSGA-II algorithm to create a set of optimal designs that minimizes the pressure drop and maximizes the heat transfer of a finned channel. Multiple case studies will be evaluated using different design parameters and temperature profiles in which a number of parameter studies and Pareto-optimal designs from the genetic algorithm will be compared. Currently, the shape and placement of the annular fins is expected to vary as different design parameters and temperature profiles are introduced. It is also expected that the shape of annular fins will vary as the designer selects different tradeoff values for the objective functions.

An additional goal of this thesis is to incorporate the process of optimizing fin designs to the optimization code for two-phase flow channels that is currently in development. To verify integration, the fin optimization code will be tested independently of the two-phase flow channel's heat transfer calculations and compared to the results found when including the channel's calculations. By the end of this thesis, I hope to have a better understanding of how the optimal design of fins varies with different design parameters and be equipped to make suggestions for the future development of the two-phase flow design tool and optimization of annular fins.

Chapter 2

Methods

Multi-Objective Optimization

To find a set of optimal designs based on user preferences, a multi-objective genetic algorithm (GA) optimization method is employed. This optimization method was selected due to the small number of variables, its likelihood to converge to the global optima, its ability to support multiple objectives, and its ease of implementation in MATLAB. Because of the range of design freedom, there are many possible combinations of objectives and performance. Instead of searching the entire design space to evaluate the performance of each design, the GA conducts an optimization process that emulates reproduction to find the set of Pareto optimal designs in a more computationally efficient way. Figure 2 shows an example Pareto set that may be presented by the GA as well as the remaining number of designs that may lie in the design space. In this example, the objectives on the plot are the inverse heat transfer coefficient and pressure drop [3]. Each point in the performance space represents a design with a different set of performance based on the objectives and geometric design. Although the intention for the optimization is to maximize the heat transfer of the system and minimize the pressure drop of the system, the standard method of optimization seeks to minimize all objectives. As a result, the fitness function evaluates the inverse of the heat transfer so that both objectives may be minimized. Due to the variability of the objective ranges, the objectives are also normalized to create a more uniformly distributed plot of results. Figure 2's plot is also an example of minimizing the normalized objectives of the inverse heat transfer and pressure drop.

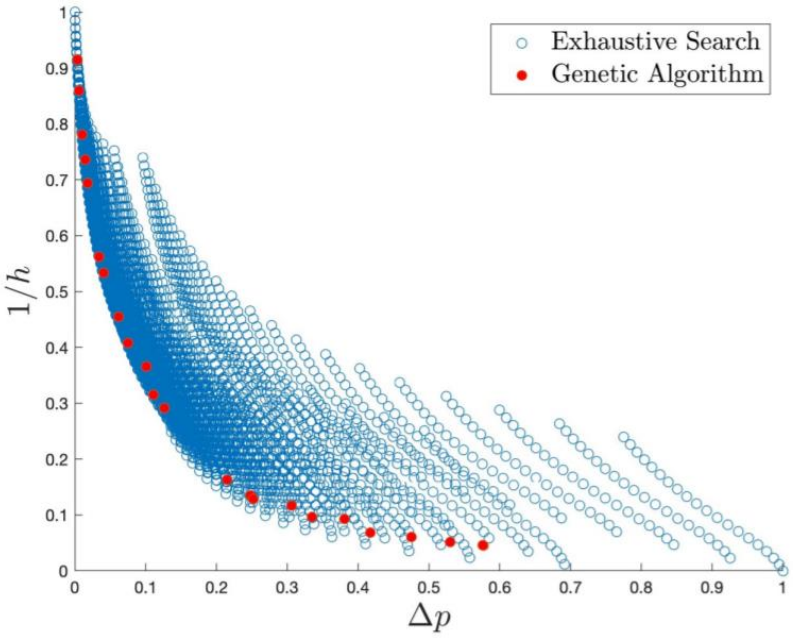

Figure 2. An example Pareto front presented by a genetic algorithm that minimizes the objectives of the inverse heat transfer coefficient and pressure drop [3]**.**

Because the number of designs is so large, the GA uses an iterative process to converge towards a set of optimal designs. Figure 3 shows a process flow chart that illustrates the process the program undergoes when searching for a Pareto optimal set of designs. During the optimization process, the GA will create an initial population of designs based on the designer's inputs, constraints, and randomized values for specified variables. After creating the first population, each design is evaluated with a fitness function that calculates each design's objectives.

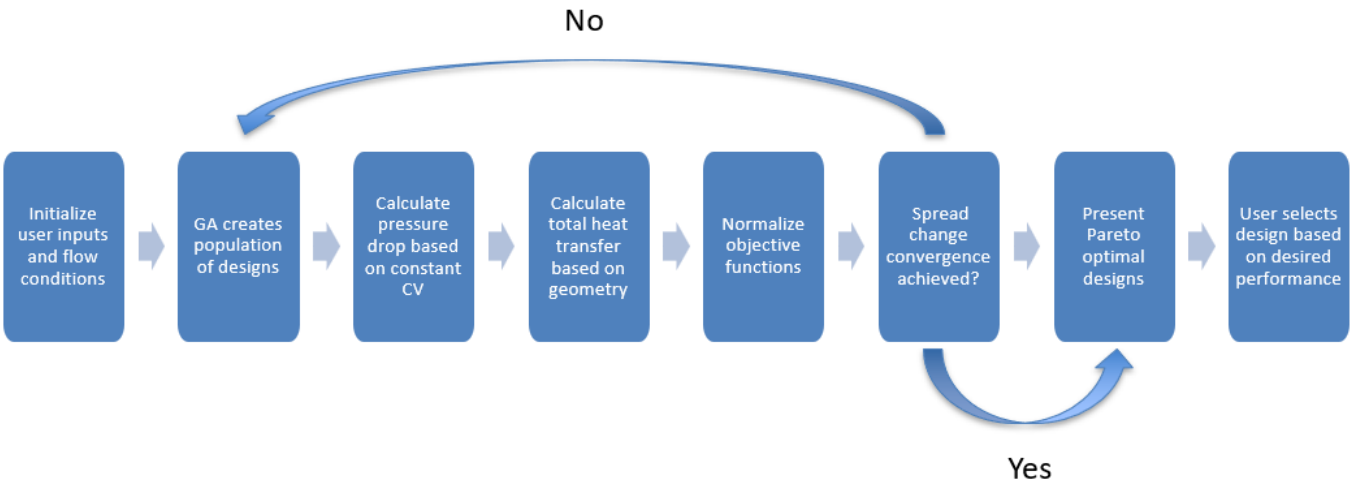

Figure 3. Process flow chart illustrating iterative process of the multi-objective genetic algorithm.

To simulate an environment of interest, user inputs are collected based on intended application of the heat exchanger. These values are held constant during the simulation and may be changed based on different application conditions. The constraints of the program ensure that the GA only produces designs that are feasible for the user's application. Furthermore, the GA prevents designs that are not feasible from propagating in future generations by penalizing their calculated fitness value if one parameter is infeasible, i.e., lies outside of the specified range in the constraint. To vary the performance of one generation to another, values of specified variables are altered with the goal of increasing fitness. After calculating the fitness of the first generation, the designs within the population will undergo crossover reproduction in which the values of the variables will be altered to create a new population of designs. This process is supplemented by mutations built into the GA that introduce diversity into the new population after reproduction. Then, the performance is recalculated, and a spread change is measured comparing the amount of change from the old generation to the new generation. As the designs

continue to improve, the spread change is expected to decrease from each generation to the next. This process will continue until a sufficiently small spread change is achieved causing the program to converge on the most recent population. A lower spread change can be set to increase the chance the program converges to the Pareto optimal set of designs at the cost of computation time. Once the program converges, the user is able select a design amongst the Pareto set that satisfies their performance requirements. Equation 1 illustrates the optimization problem of the GA in standard form in which the objective to the program is to minimize the pressure drop dP and the inverse heat transferred Q_{inv}. The GA will be constrained by limits set onto the variables of the system, number of fins $(n_{fin,i})$, radius of fins $(r_{fin,i})$, and thickness of fins $(t_{fin,i})$, as shown below. Specifically, the radius of the fins has a lower bound of the channel diameter $(d_{channel})$, and the thickness of fins has an upper bound of the length of the channel $(L_{channel})$ These constraints were selected specifically for this study and may be altered in future applications of this work.

$$
\begin{aligned}
&Minimize: && (dP, Q_{inv}) && (1)\\
&Subject\ to: && 0 \leq n_{fin,i} \leq 99 &&\\
& && d_{channel} \leq r_{fin,i} \leq 0.1 &&\\
& && 0.001 \leq t_{fin,i} \leq L_{channel} &&
\end{aligned}
$$

Fin Geometry and Parameters

The application for this study analyzes a heat exchanger with circular tubes and annular fins. In practice, channels would likely be connected to maximize the performance of the system. However, for the simplicity of the study, a singular channel within the entire heat exchanger system will be analyzed. Figure 4 illustrates the control volume that will be drawn around a singular channel as opposed to analyzing the entire system.

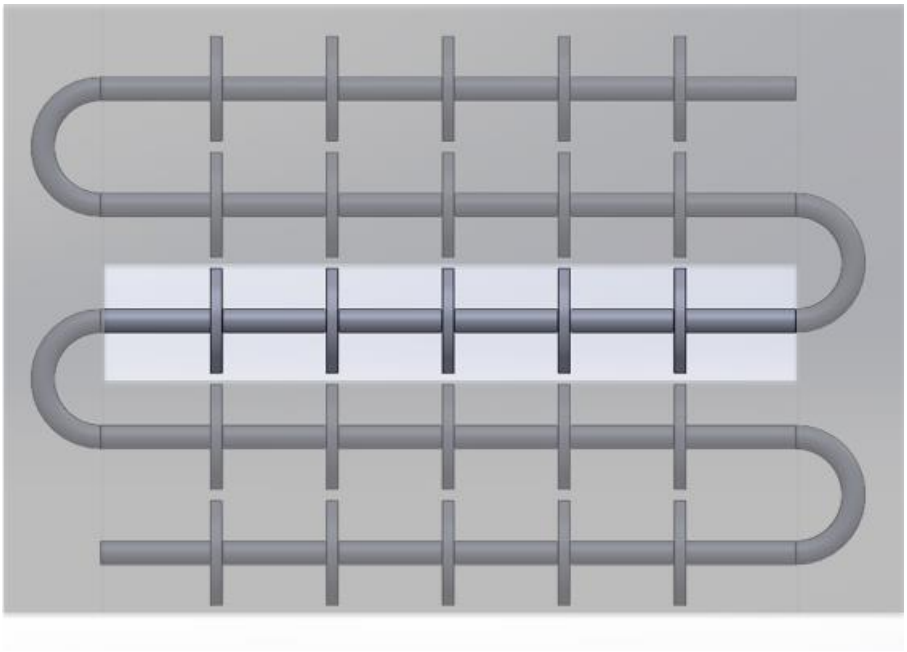

Figure 4. Heat exchanger system with control volume used for optimization highlighted.

To define the geometry of the channel, several parameters are introduced as constants, variables, and user inputs. In figures 5, 6, and 7, a frontal, side, and isometric view of the channel is shown. Shown in figure 6, the radius of the channel and fin (r_2) and the diameter of the channel ($d_{channel}$) are shown. Parameters such as the length of the channel ($L_{channel}$), diameter

of the channel ($d_{channel}$), and control volume size (depth and height defined as h_{CV}) are held as constants throughout the experiment. Additionally, the fin radius $(r_{fin,i})$, fin thickness $(t_{fin,i})$, and number of fins $(n_{fin,i})$ are the variables that will be manipulated from case to case. Finally, the user inputs include the fluid conditions of the airflow going into the heat exchanger system.

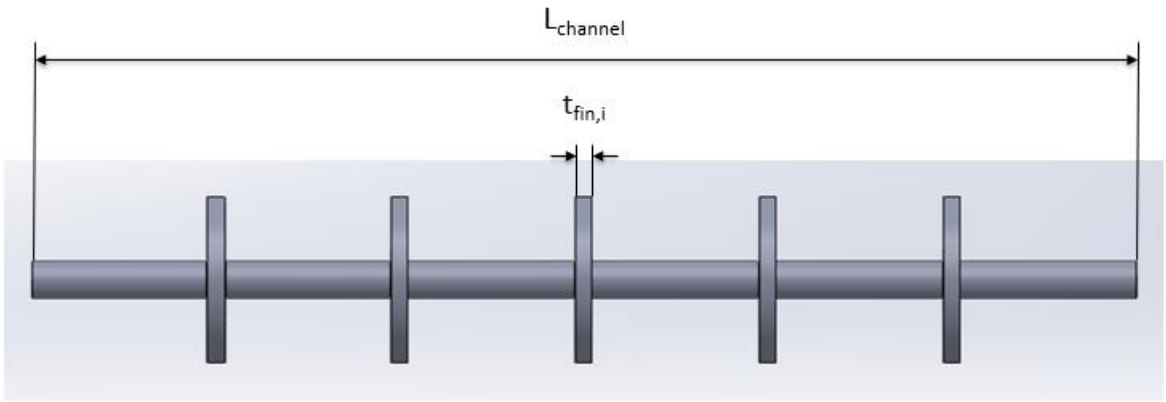

Figure 5. Frontal view of experimental heat exchanger channel.

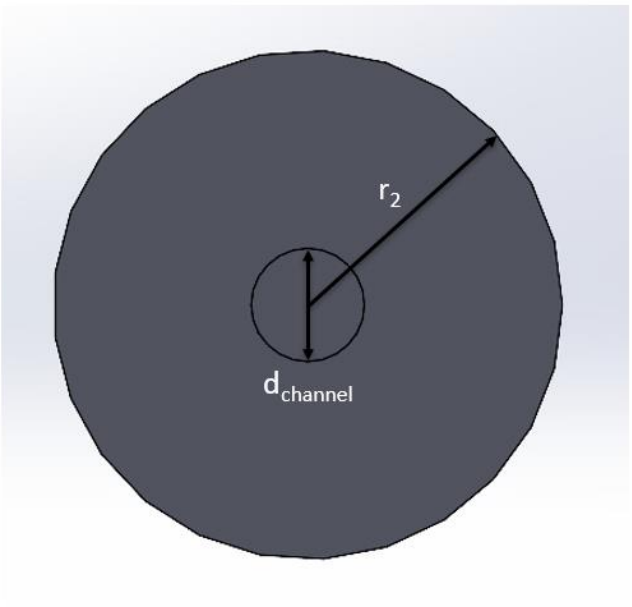

Figure 6. Side view of experimental heat exchanger channel.

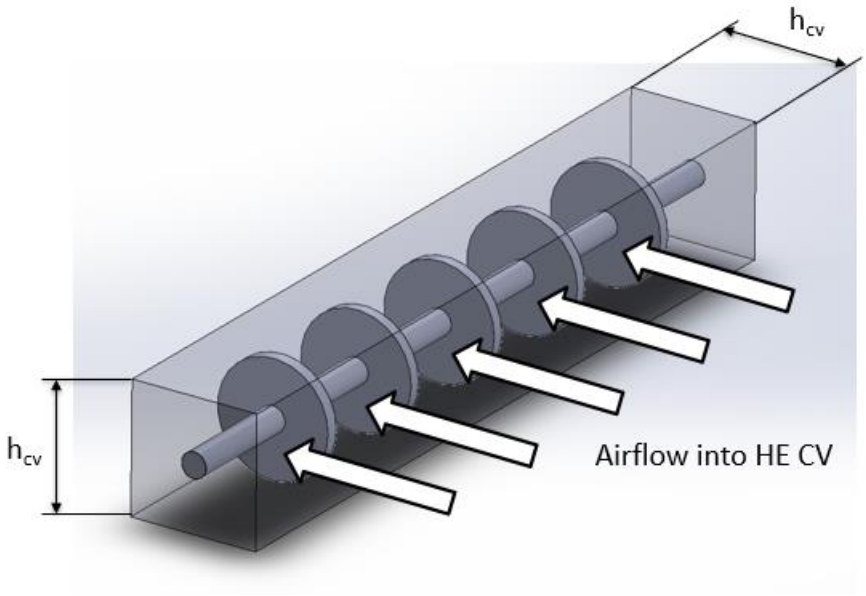

Figure 7. Isometric view of experimental heat exchanger channel.

Heat Transfer Models

To calculate the heat transferred from a given geometry, heat transfer correlations with specified ranges of applicability are employed. Because the geometry of the system includes both a cylindrical shaft and annular fins, two different correlations are used in conjunction to calculate the total heat transferred. To measure the performance of the channel, a model is used to approximate the convective heat transferred as a cylinder in cross flow. Additionally, to account for parallel flow on the annular fins, a flat plate correlation was applied. Although using these two models in conjunction is a relatively accurate method of estimating the heat transfer from this heat exchanger geometry, future applications of this work may benefit from using a correlation that accounts for the full geometry of annular fins on a cylindrical shaft.

The total heat transfer for a cylinder in cross flow is calculated using equation 2 [2]. This calculation is taken based on the diameter of the shaft, and accounts for the heat transfer from the portions of the channel that are unfinned. This correlation is valid under certain fluid conditions in which the Prandtl number is greater or equal to 0.7, and the Reynolds number (Re_D) is greater than 0.4 and less than 400,000 [2].

$$\overline{Nu_D} = \frac{\bar{h}D}{k} = C_{cyl} Re_D^{m_{cyl}} Pr^{1/3} \quad (2)$$

Although calculating the heat transfer from the channel is relatively straight forward, the approach to calculate the heat transfer from the fins is more complex. Equation 3 is used for calculating the total heat transfer for a flat plate with parallel laminar flow is valid for flows with Reynold's numbers (Re_x) less than 500,000 and Prandtl numbers greater or equal to 0.6 [2].

$$\overline{Nu_x} = \frac{\overline{h_x}x}{k} = 0.664Re_x^{1/2}Pr^{1/3} \quad (3)$$

The common application for this correlation is measuring the heat transfer for an infinitely wide plate that has a specified length of x. However, for a circular channel with annular fins, equating parameters of the fins to an equivalent theoretical length of a flat plate is not intuitive. A preliminary study was conducted to investigate how equating different fin parameters to the x term in the flat plate correlation affects the outputted heat transfer of the system. Figure 8 shows the 3 different parameters that were evaluated: the channel for the diameter ($d_{channel}$), half of the max finned diameter ($\frac{d_{max}}{2}$), and the max finned diameter (d_{max}). At the conclusion of the study, the heat transfer coefficient was found to have an inversely proportional relationship with the value substituted for x which was expected. As a result, a choice was made to consider ($d_{max} - d_{channel}$) as a sufficient intermediary value to approximate the fin geometry to a flat plate.

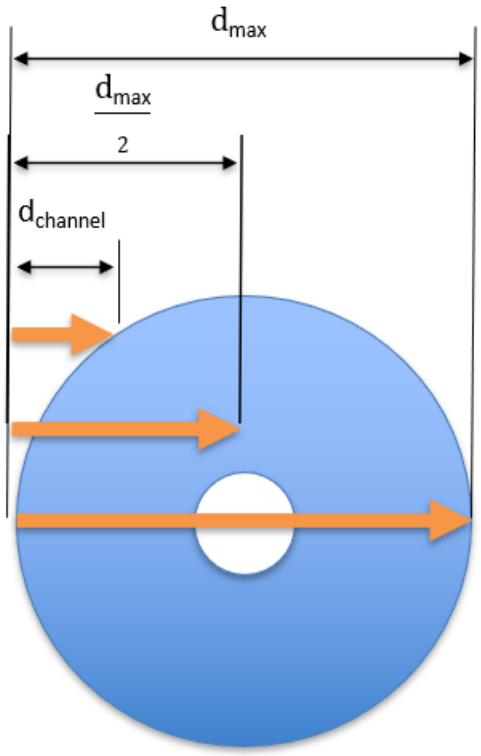

Figure 8. Tested finned channel parameters used to find appropriate length "x" for flat plate correlation.

To account for the heat transfer coefficient for the finned part of the channel, a fin efficiency equation must be used. Equations 4-8 are used to calculate the annular fin efficiency present on the channel. Figure 9 shows an illustration of the geometric parameters used in the fin efficiency equation [2]. r_1 and r_2 are defined as the channel radius and the channel and fin radius respectively. This equation also accounts for the heat transfer present on the tip of the fins, and if a user desires a different shape for the fin, this term would be replaced with the appropriate fin efficiency equation for a different geometry. After accounting for the heat transfer of the channel and the fins, the total heat transfer may be approximated by summing these two values.

$$r_{2c} = r_2 + \frac{t}{2} \tag{4}$$

$$A_f = 2\pi(r_{2c}^2 - r_1^2) \tag{5}$$

$$m = \sqrt{\frac{2h_{fin}}{t_{fin,i} * K}} \tag{6}$$

$$C_2 = \frac{\left(\frac{2r_1}{m}\right)}{(r_{2c}^2 - r_1^2)} \tag{7}$$

$$\eta_f = C_2 \frac{K_1(mr_1)I_1(mr_{2c}) - I_1(mr_1)K_1(mr_{2c})}{I_0(mr_1)K_1(mr_{2c}) + K_0(mr_1)I_1(mr_{2c})} \tag{8}$$

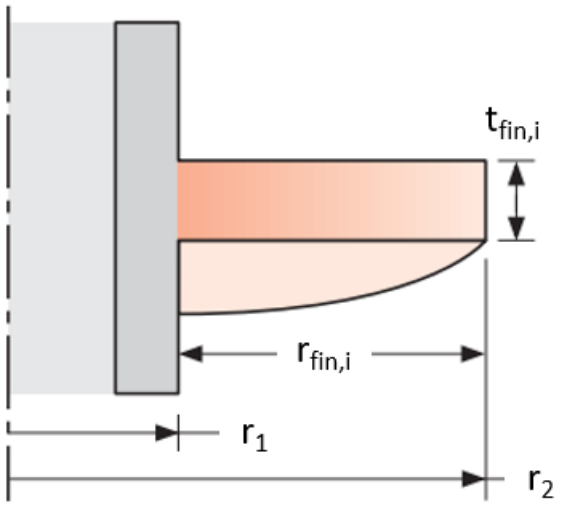

Figure 9. Illustration of parameters for the fin efficiency [2].

Heat Exchanger Pressure Drop Correlation

To approximate the pressure drop of air travelling through the heat exchanger, a compact heat exchanger correlation is used and shown in equation 9 [6].

$$\frac{dP}{P_1} = \frac{G^2}{2g_c}\frac{v_1}{P_1}\left[(1+\sigma^2)\left(\frac{v_2}{v_1}-1\right)+f\frac{A_s}{A_{free}}\left(\frac{v_m}{v_1}\right)\right] \quad (9)$$

To incorporate the correlation, several assumptions have been made in application during the experiment. Although the friction factor may vary for a given fluid under different conditions, a simplification was made in the model in which the friction factor is assumed to be a constant value for air at room temperature. Accuracy of the model may be improved in the future by incorporating a model that accurately identifies the value for the friction factor based on fluid conditions. Additionally, the $\frac{v_2}{v_1}$ term accounts for entrance effects and differences in density as the fluid flows through the system. However, because the application for the designs do not present a significant change in density as the fluid flows through the heat exchanger, the density fluid is assumed to be constant, resulting in $\frac{v_2}{v_1} = 1$. Another assumption that was made during the experiment was to hold the control volume constant as the genetic algorithm iterates designs. Because there is a $\frac{A_S}{A_{free}}$ term that describes the ratio of the surface area of the heat exchanger to the free flow area, the control volume is held constant so that the pressure drop of the different designs can be compared one to one.

Chapter 3

Results

Initial Parameter Studies

To verify that the algorithm is behaving properly, several parameter studies were conducted. During this process, it was found that as the number of fins, radius of fins, and thickness of fins, were increased, the total heat transferred and the pressure drop were found to also increase, which is expected. Figure 10 shows a plot that illustrates a parameter study used to verify the effects of changing each variable on the performance of the system. In each variation, a single variable is held is measured through its entire feasible range while the other two variables are held constant. The values the variables were held constant at were as follows: $n_{fin,i} = 10, r_{fin,i} = 0.05m, t_{fin,i} = 0.001m$. It is notable that, although all variables have a direct correlation with the performance of the system, different variables have a larger impact at different levels of performance. For example, varying fin radius (grey) has a large impact on increasing the heat transfer of the system as opposed to varying the fin thickness (yellow) due the larger of range of change seen by each parameter.

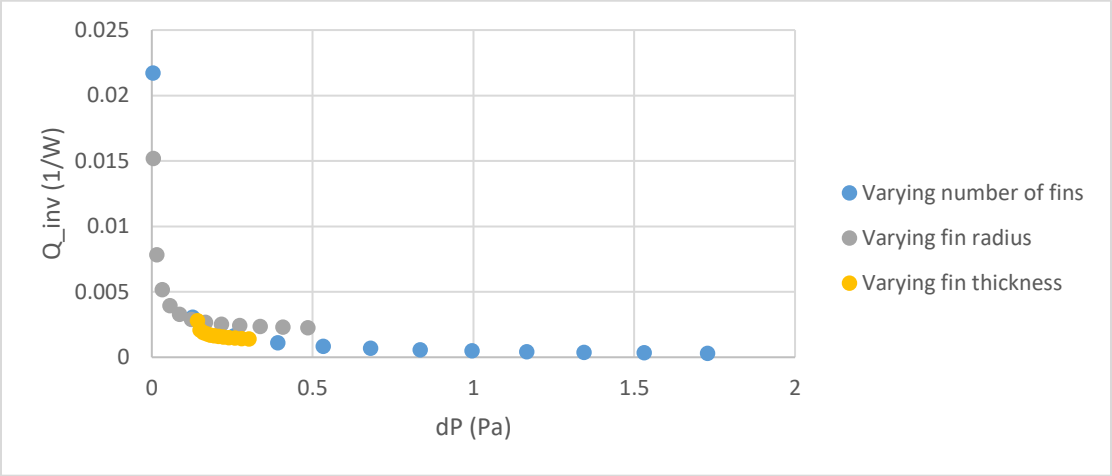

Figure 10. Plot of parameter study for geometric variables.

However, when implementing the iterative process of the genetic algorithm, convergence issues were identified in which the entire Pareto set of designs would converge to have extremely short fins with varying quantity. This caused issues with the Pareto set as all designs essentially simulated having no fins while retaining a nonzero value for the radius of fins. This is likely due to the methodology used to evaluate the designs being based on a finned channel geometry, thus an application range must be introduced. As a result, a constraint was introduced to restrict the design space to have a fin radius of at least 10% of the diameter of the channel to prevent this error from occurring. Applying the constraint, however, still allows the program to analyze the trivial solution since the minimum number of fins is still zero. After verifying the effect of variables through a parameter study and accounting for small fin errors, a number of case studies were established to investigate optimal designs of the system.

Summary of Case Studies

To investigate the optimal design of the size and number of fins, several case studies were conducted. Table 1 lists each case study according to number along with the associated design variables and temperature profile used along the channel. In cases 1 and 2, a constant temperature profile is used. The primary purpose of these two cases is to observe the optimal design based on geometric variables only. It is expected that when these variables are at their highest values, it will result in the highest heat transferred whereas when they are lowest, the lowest pressure drop will be produced. Case 3 introduces a varying temperature profile where the shape of the fins along the length of the channel will be investigated as the temperature varies. Thickness of the fins was not considered as a variable to facilitate simplicity for those complex cases.

Table 1. List of case studies with corresponding variables and temperature profiles.

Case Study Number	Design Variables	Type of Temperature Profile
1	$n_{fin,i}, r_{fin,i}$	Constant
2	$n_{fin,i}, r_{fin,i}, t_{fin,i}$	Constant
3	$n_{fin,i}, r_{fin,i}$	Linear

Appendix A lists all the parameters used in this study. These parameters are differentiated by constants, inputs, and variables. For simplicity and consistency of the experiment, the inputs have been held constant throughout the case studies. Table 2 includes the values for all the constants and inputs used in this study. Each constant is defined and listed as an input in Appendix A.

Table 2. Values for constants used in experiment.

Symbol	Value	Units
g	9.81	$\frac{m}{s^2}$
C	0.193	
m	0.618	
v_2	0.816	$\frac{m^3}{kg}$
$t_{fin,min}$	0.001	m
MP	144	
P_1	39,600,000	Pa
f	0.0064	
v_1	0.816	$\frac{m^3}{kg}$
G	10	$\frac{kg}{(s * m^2)}$
Pr	0.7	
T_b	373.15	K
T_f	293	K
K_f	0.0263	$\frac{W}{m * K}$
K	237	$\frac{W}{m * K}$
μ	0.0000185	$\frac{N * s}{m^2}$
$L_{channel}$	0.2	m
$d_{channel}$	0.01	m
h_{CV}	0.23	m

As case studies were conducted, a visualization code was developed to illustrate selected designs in the Pareto set. The script developed for the visualizations is included in Appendix B.

Case Study 1: Varying Fin Radius and Number of Fins

Case study 1 investigates the effect of varying the fin radius and number of fins on the pressure drop and heat transfer of an annularly finned heat exchanger. Figure 11 shows the results for the set of pareto optimal designs produced in this case study. The top plot represents the designs in the performance space in which each design has a corresponding pressure drop and heat transfer. The bottom plot is the spread change of each generation in the GA as the program iterates through each generation. It is observed that the spread change generally decreases as expected and converges once achieving a specified threshold. The convergence tolerance used in this thesis was the default value (0.001) of the multi-objective GA in MATLAB. It is also observed that majority of the designs lie closer to the x axis of the plot showing there to be a small difference in heat transfer while still having a large amount of difference in the pressure drop.

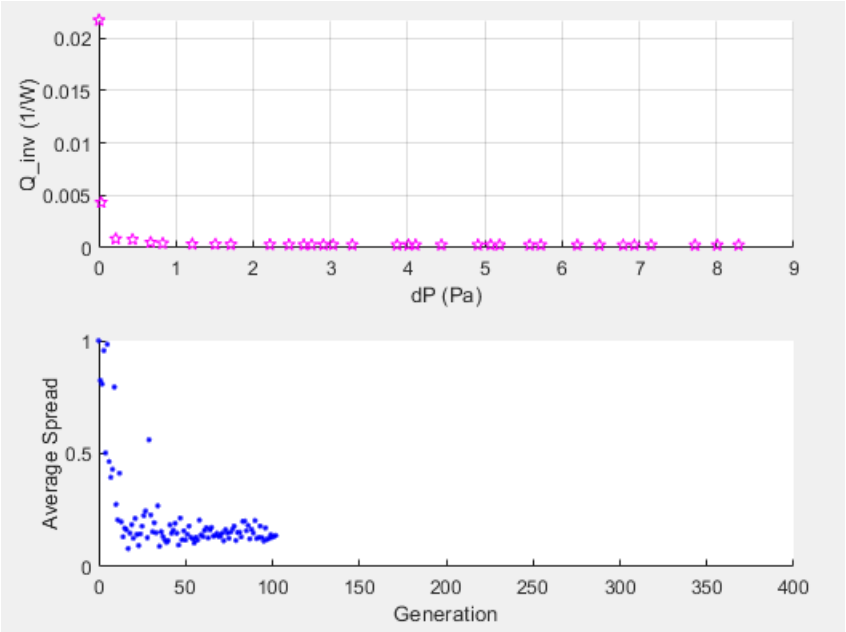

Figure 11. Pareto front and spread change results for case study 1.

To show a more even distribution of the designs, the problem is solved using normalized objectives as in equations 10 and 11. Each parameter, has a corresponding set of minimal and maximal values $(dP_{min}, dP_{max}, Q_{inv,min}, Q_{inv,max})$ feasible within the design space to normalize the results of a given design. Figure 12 show the results of running the program again while normalizing the output after it is calculated in the fitness function. As a result, the GA iterates based on the normalized objective values. It can be seen that a more evenly distributed Pareto set is achieved after normalization, and it is concluded that the normalized objectives allow the GA to consider the entire design space without having one objective dominate the other one due to large differences in their magnitudes.

$$dP_{norm} = \frac{dP - dP_{min}}{dP_{max} - dP_{min}} \tag{10}$$

$$Q_{inv,norm} = \frac{Q_{inv} - Q_{inv,min}}{Q_{inv,max} - Q_{inv,min}} \tag{11}$$

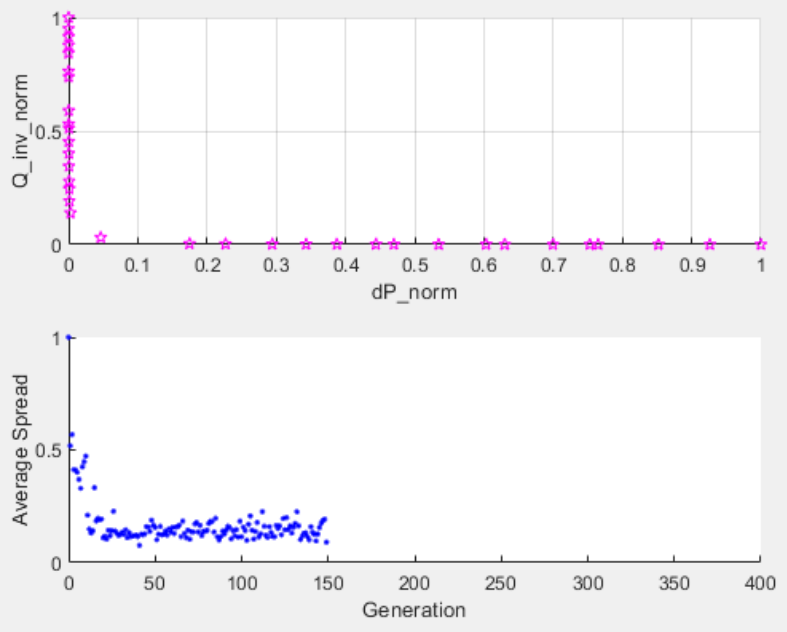

Figure 12. Normalized Pareto front and spread change results for case study 1.

Figure 13 shows a representation of selected designs associated with the results found from case study 1. It can be observed that as designs move from the top left of the Pareto front to the bottom right, both the radius of fins and number of fins increase. For example, in figure 13, the design in the top left of the Pareto front has no fins, and as you move to the right of the plot, the optimal designs have 38 and 99 fins. This observation coincides with the previous expectation that as the number of fins and radius of fins increases, the total heat transfer and pressure drop will increase. Table 3 lists the values of the parameters and objectives at the extremes and design closest to the utopia in the Pareto set. In conclusion, this case served as a valuable confirmation of the best designs obtained intuitively before moving on to the other case studies.

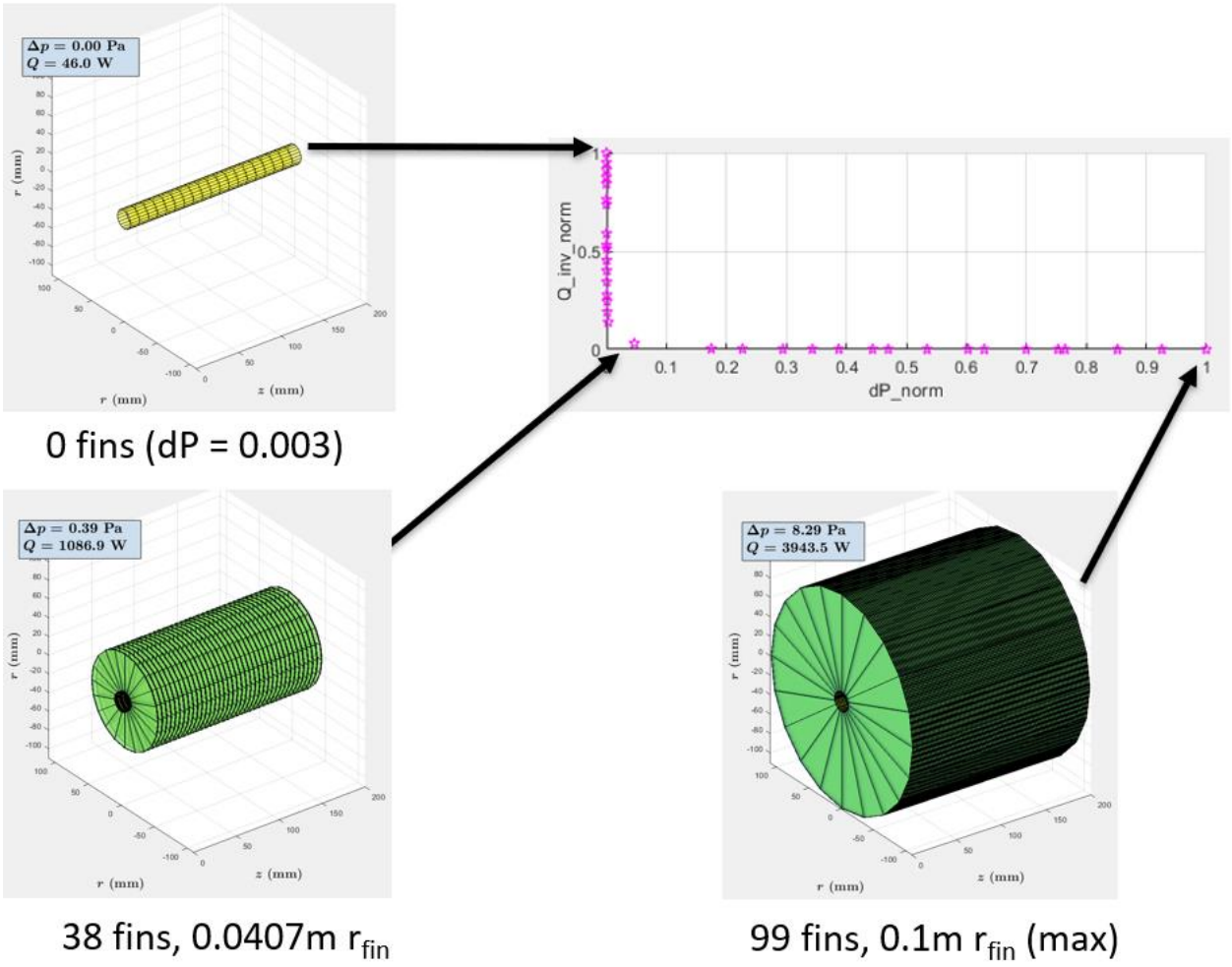

Figure 13. Diagram showing visual representation of designs for case study 1.

Table 3. Parameters of designs at extremes and closest to utopia in case study 1.

$n_{fin,i}$	$r_{fin,i}$	$dP\ (Pa)$	$Q\ (W)$	Distance to utopia point
0	0.001	0.003	46	1
38	0.0407	0.39	1086.9	0.0559
99	0.1	8.29	3943.5	1

Case Study 2: Varying Fin Thickness, Fin Radius, and Number of Fins

This case study investigated the effects of increasing the geometric freedom considered by the GA. However, because of the additional variables, the design space becomes restricted due to feasibility of designs in application. For example, if a channel has a single fin, that fin could be as long as the length of the channel. However, if there are more than one fin on the channel, the max thickness of the fin must be constrained to remain in the bounds of the length of the channel while maintaining minimum spacing between them. Because both the number of fins and fin thickness determine how much space is taken up by the fins along the length of the channel, an issue of interdependent variables arises.

To consider only feasible designs in the GA, a penalization algorithm is implemented in which, when the design constraints are violated, the fitness value of an infeasible design will be penalized and therefore be unlikely to propagate into future generations. Because the goal of the GA is to minimize the value of the objectives, the fitness value of an infeasible design is penalized by setting the value of the objectives to infinity when the geometric constraint is violated. Equation 12 shows the upper bound for each variable independent of the other. Equation 13 is used to calculate the maximum value of the interdependent variables. The top of figure specifies the upper bound of each variable when considered independent of the other. However, as fin thickness is introduced as a variable to the GA a new upper bound must be considered for the number of fins and thickness of fins.

$$Upper\ bound\ for\ each\ variable\ independently$$

$$n_{fin,i} \leq 99 \tag{12}$$

$$t_{fin,i} \leq L_{channel}$$

$$Actual\ upperbound\ with\ interdependent\ variables\ (t_{min} = 0.001m)$$

$$n_{fin,i} \leq n_{fin,max} = \frac{\left(L_{channel} - \left(\left(n_{fin,i} + 1\right)\right) * t_{min}\right)}{t_{fin,i}} \tag{13}$$

$$t_{fin,i} \leq t_{fin,max} = \frac{\left(L_{channel} - \left(\left(n_{fin,i} + 1\right)\right) * t_{min}\right)}{n_{fin,i}}$$

Figure 14 shows a plot of the design space in which the constraint functions for the maximum number of fins and thickness of fins is plotted. By plotting these constraints, one can observe how much the design space has been constrained now that the thickness of fins has been added as a variable (only designs on or under the curve are feasible). Had thickness been removed as a variable, the entire design space presented in figure 14 would be feasible.

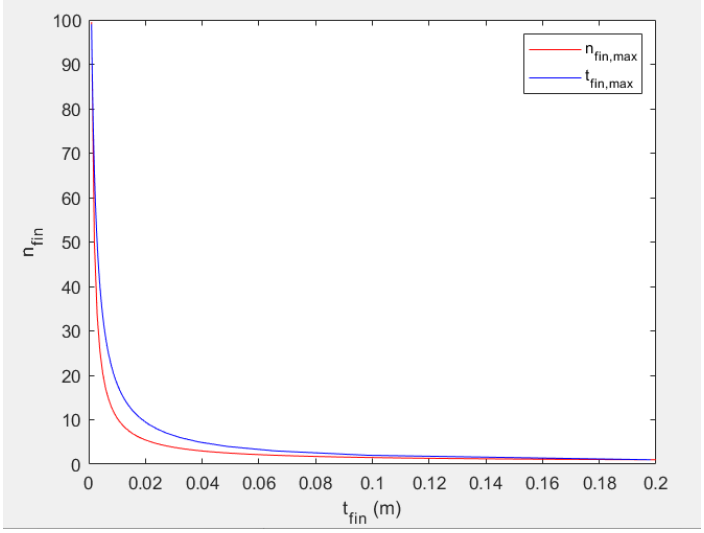

Figure 14. Plot of design space restricted by maximum geometric constraints (only designs under the curve are feasible).

Figure 15 shows a plot of the results from introducing fin thickness as a variable to the GA. Once again, the majority of the designs have similar levels of heat transfer performance whereas there is more difference in the amount of pressure drop across the Pareto set. The spread change is found generally to decrease until convergence, as expected.

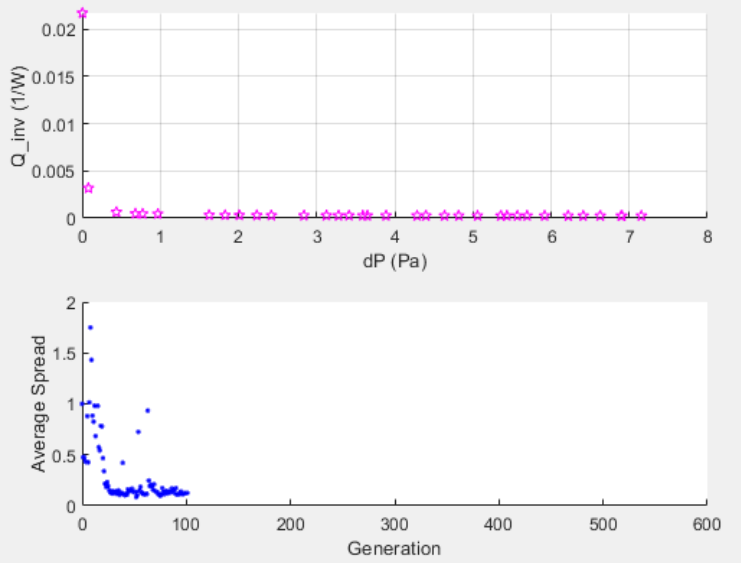

Figure 15. Pareto front and spread change results for case study 2.

Figure 16 shows the results from case study 2 with normalized objectives. It is notable that the optima used for normalization in this case varies from the previous case due to increased design freedom. As a result, when normalizing the results, the maximum and minimum objective values used must be adjusted to values corresponding to a system with thickness as an added parameter. A key takeaway from this plot is that, although the designs in the pareto set have a wide range of performance for the heat transferred, the design with the largest pressure drops only produces 76% of the maximum pressure drop. This issue likely arises due to the large reduction in feasible designs from introducing the penalty function.

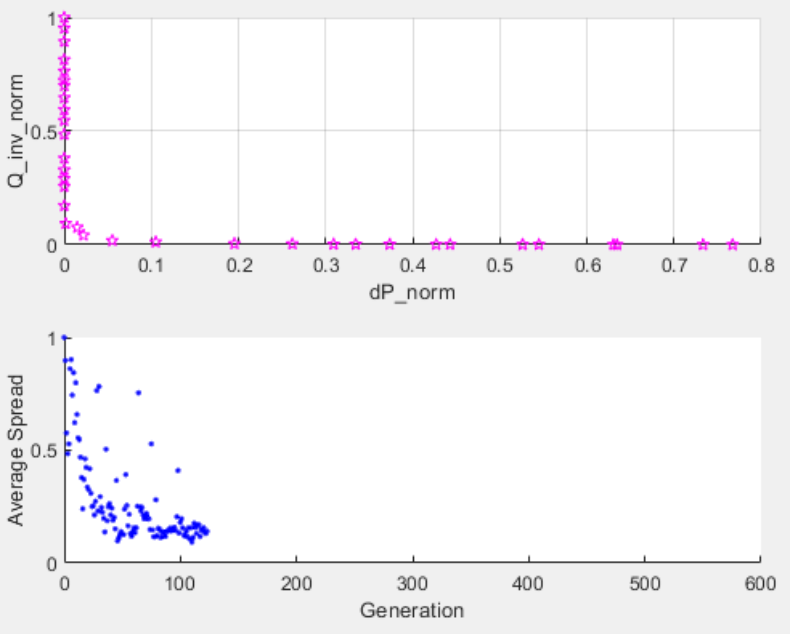

Figure 16. Normalized Pareto front and spread change results for case study 2.

Figure 17 illustrates the selected designs at the extremes and the design closest to the utopia point for the results of case study 2. It is observed that as one moves from the top left to the bottom right of the plot, the number of fins increases as the radius and thickness varies. However, it is still found that as designs move to the right, the mass of the channel increases due to increased number of fins, fin radius, or fin thickness which is expected. Table 4 lists the parameter and objective values for designs at the extremes and closest to the utopia point in the pareto set.

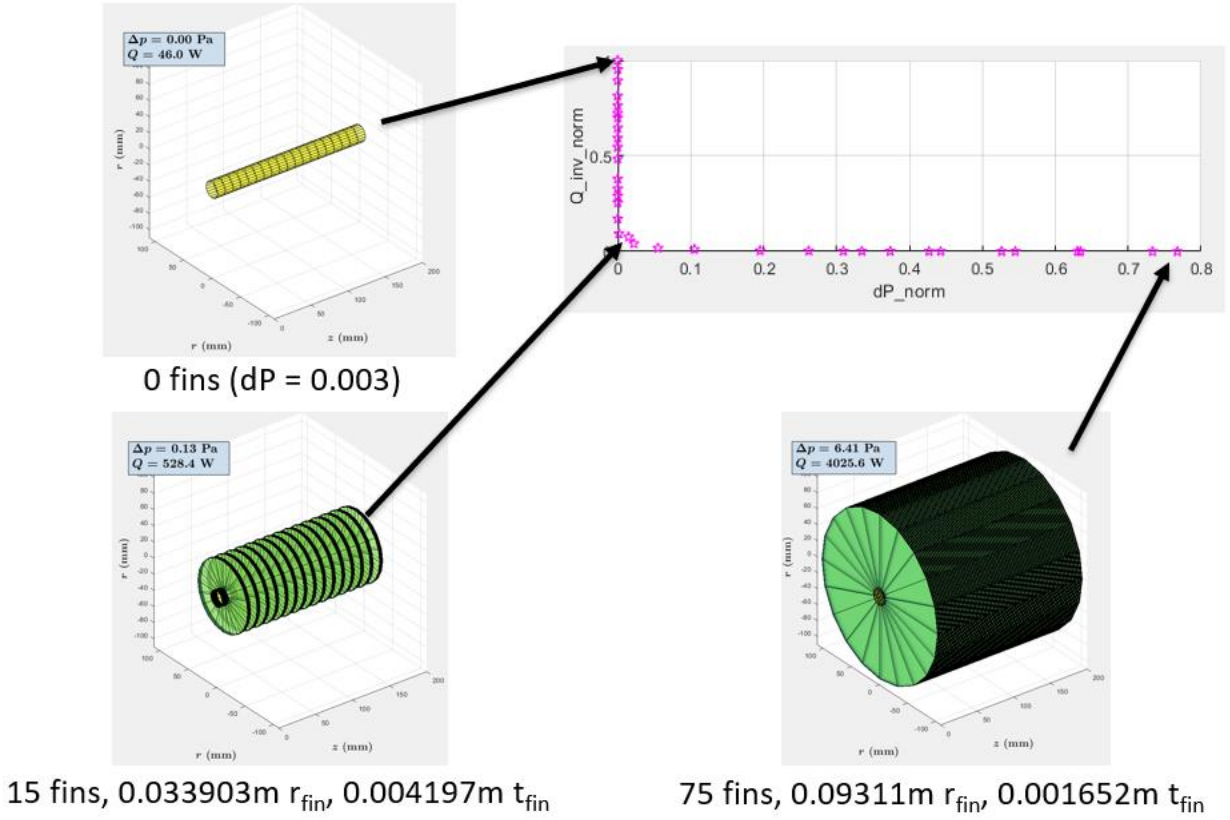

Figure 17. Diagram showing visual representation of designs for case study 2.

Table 4. Parameters of designs at extremes and closest to utopia in case study 2.

$n_{fin,i}$	$r_{fin,i}$	$t_{fin,i}$	$dP\ (Pa)$	$Q\ (W)$	Distance to utopia point
0	0.001	0.001	0.003	46	1
38	0.03390	0.004197	0.13	528.4	0.00778
75	0.09311	0.001652	6.41	4025.6	1

To compare the results from case study 1 and 2, figures 18 and 19 were used to plot the results from each simulation. By looking at figure 18, table 3, and table 4, one can see that there

is no significant increase in performance by adding fin thickness as a variable. Although the design with the highest number of fins has a higher value for heat transfer and lower value for pressure drop, it is found that, for the design closest to the utopia point, the pressure drop and heat transfer is lower for case 2 than case 1 This is likely because the GA is already selecting the most optimal ratio of fin parameters for performance. Additionally, figure 19 shows the effect of the interdependent variable constraints by illustrating the fact that case 2's results do not span across the entire range of pressure drop available to the systems design potential.

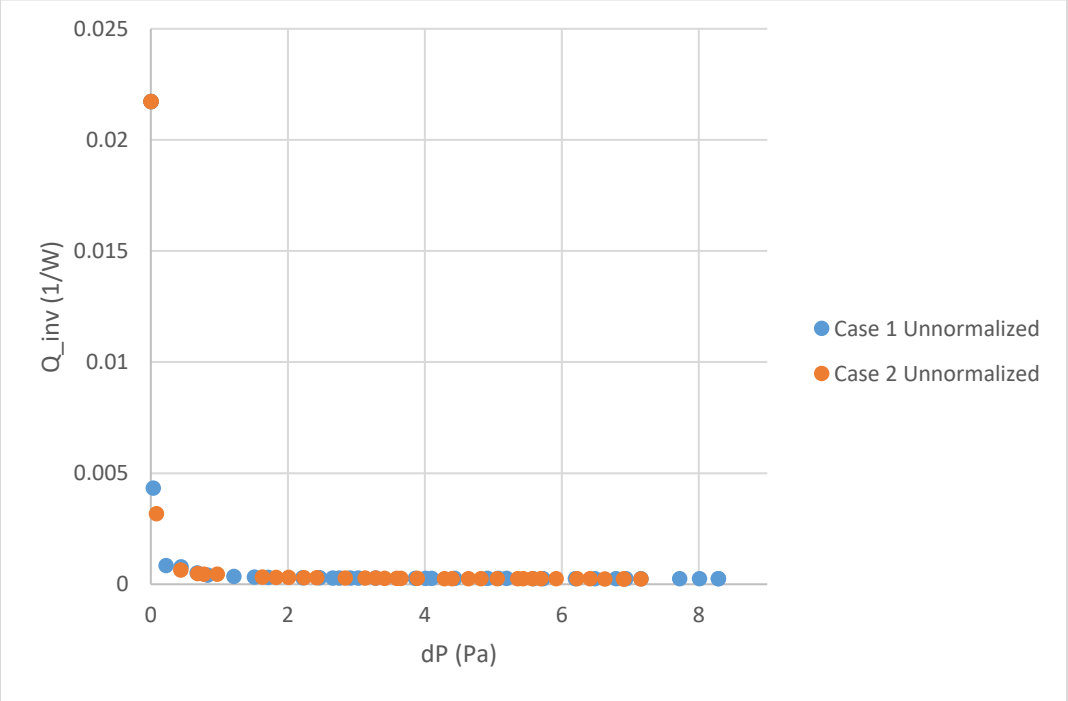

Figure 18. Pareto front results comparing case study 1 and 2

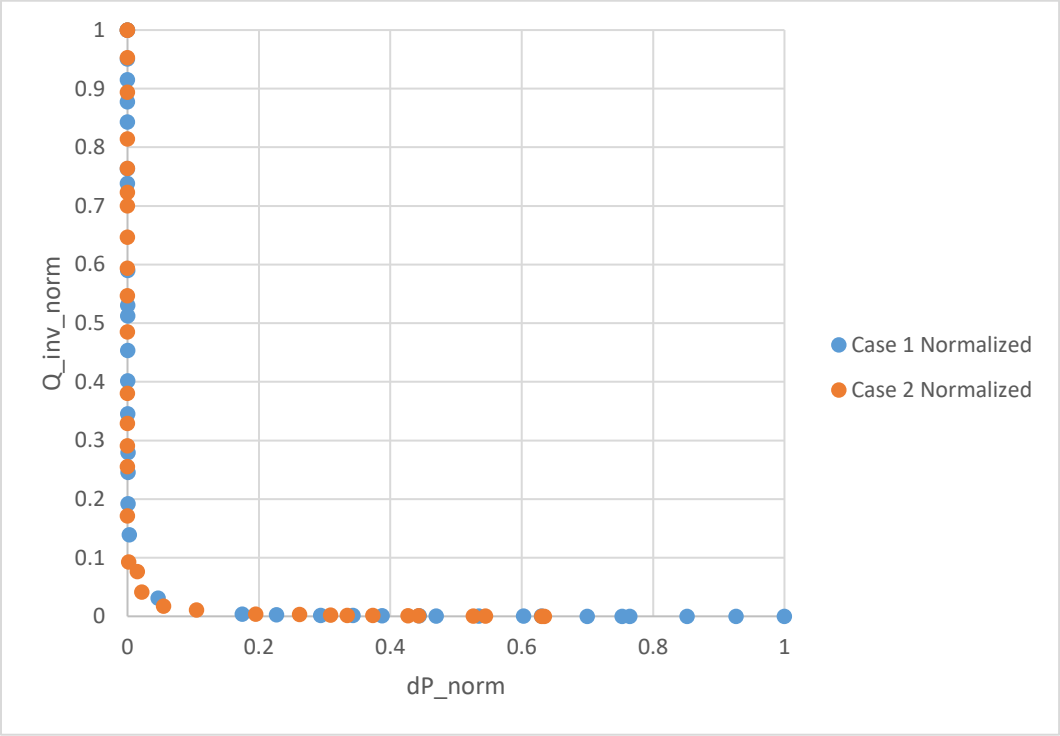

Figure 19. Normalized Pareto front results comparing case study 1 and 2.

Case Study 3: Varying Fin Radius and Number of Fins with Linear Temperature Profile

This case study was developed to investigate the optimal design of fins along the length of the channel under a varying temperature profile. To accomplish this, the channel is broken up into discretized sections, and a linear temperature profile is applied along the length of the channel. To approximate a varying temperature along the length of the channel, the average base temperature across a given section is applied as a constant base temperature. As a result, the previous methodology of calculating the fitness of each section may be used. Figure 20 shows an illustration that depicts the stepwise temperature profile used in case study 3.

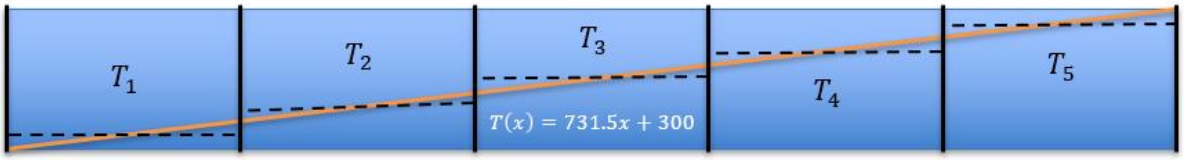

Figure 20. Discretized temperature profile of channel in case 3.

However, by introducing discretized sections of varying temperatures, the calculation of the pressure drop must be revised. Because the geometry along the length of the channel is expected to change, the pressure drop along the length of the channel will also vary from section to section. In reality, lateral pressure drop along the length of the channel would not be sustained, and the entire channel is expected to have a constant pressure drop. As a result, the pressure drop is assumed to be constant across all sections of the channel, and a new objective of mass flow rate (FR_i) is introduced to measure the performance of the channel. In addition to pressure drop, mass flow rate is a helpful measure of the amount of pumping power (P_{fan}) required in application. Equations 14 and 15 show the calculation of pumping power from a fan and mass

flux (G_i) previously used to measure the pressure drop. The mass flux of the system of a function of the free flow area available (A_{free}). Both equations are used case 3 to calculate the mass flow rate of the system. Because the mass flow rate is directly related to the fan power needed to cool the system, the GA will search for designs in which this parameter will be minimized. To the total flow rate of the system (FR_{system}), the individual mass flux (G_i) and flow rate (FR_i) must be calculated for each section of the channel must be calculated. Afterwards, the flow rates of each section are summed to calculate the total mass flow rate of the system as shown in equation 16. Because the mass flux must be altered for each section, both Reynold's numbers must be recalculated based on the section parameters before calculating the heat transfer of the system, as shown in equations 17 and 18. Finally, the heat transfer of the system ($Q_{inv,system}$) is calculated by summing the convective heat transfer of each section and taking the inverse of that sum, shown in equation 19.

$$P_{fan} = FR * dP \tag{14}$$

$$G_i = \frac{FR_i}{A_{free}} \tag{15}$$

$$FR_{system} = \Sigma FR_i \tag{16}$$

$$Re_{D,i} = \frac{G_i d_{channel}}{\mu} \tag{17}$$

$$Re_{x,i} = \frac{G_i x_i}{\mu} \tag{18}$$

$$Q_{inv,system} = \frac{1}{\Sigma Q_{conv,i}} \tag{19}$$

A summation of conditions for case study 3 is found in table 5. To evaluate the designs, the optimization is solved for a range of constant pressure drops. This range was determined based on the results of the pressure drop in case study one varying from 0-8 Pa. The temperature profile used was chosen such that the middle section of the channel was the same temperature as the base temperature used in cases 1 and 2. As a result, differences in performance using an approximately linear temperature profile may be compared.

Table 5. Conditions used in case study 3.

Number of sections	5
Variables	$n_{fin,i1}, r_{fin,i1}, n_{fin,i2}, r_{fin,i2}, n_{fin,i3}, r_{fin,i3}, n_{fin,i4}, r_{fin,i4}, n_{fin,i5}, r_{fin,i5},$
Constant Pressure Drops (Pa)	0.001, 0.25, 2, 4, 6, 8
Temperature profile (K)	$T(x) = 731.5x + 300$

Figures 21-26 present the results of case study 3 for each pressure drop level. It is observed that for the first 3 simulations, the program converges to one optimal design. It is also observed that for these cases, the spread change varies between 1 and 0 before converging to the final design. This observation is evidence that the algorithm thoroughly checks the design space before converging to the single optimal design. For the last 3 simulations, a more evenly distributed Pareto set was presented. In these cases, a typical spread change behavior is observed in which the spread change decreases as the number of generations increases. The reason why the first 3 simulations converge only one design is likely because, for low pressure drop conditions, the changes in mass flux have very little effect on the Reynold's number. As a result,

the program identifies the design with the maximum heat transfer to be the best design. However, as the pressure drop increases, the decrease in mass flux has a larger impact on the Reynold's number, thus creating a competing objective between mass flow rate and total heat transferred. As a result, the simulations with higher constant pressure drops observed an evenly distributed Pareto set instead of a single optimal design.

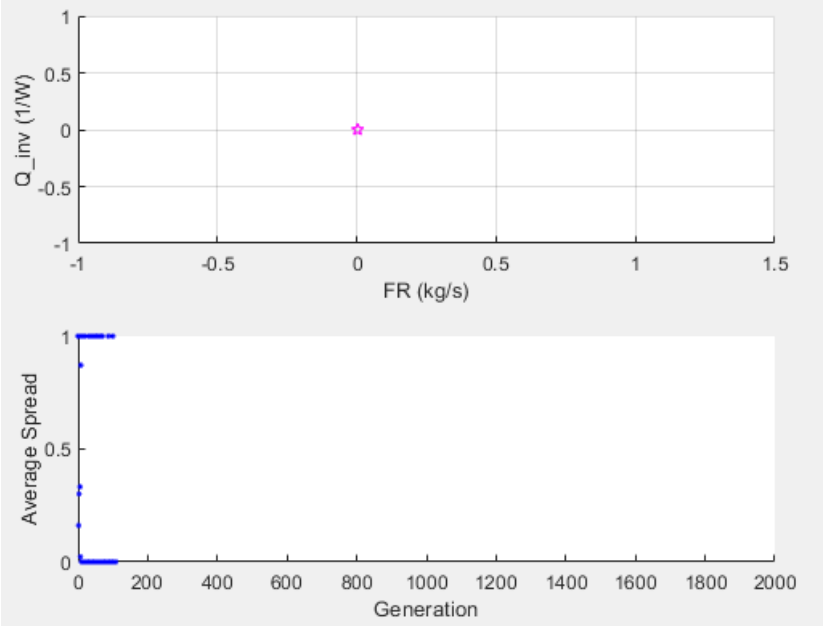

Figure 21. Pareto front and spread change results for case study 3 ($dP = 0.001Pa$).

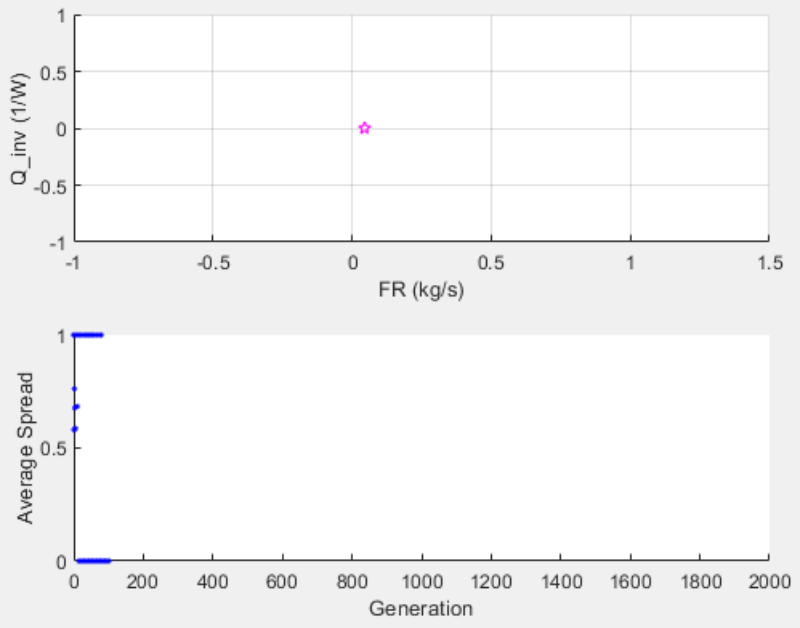

Figure 22. Pareto front and spread change results for case study 3 ($dP = 0.25Pa$).

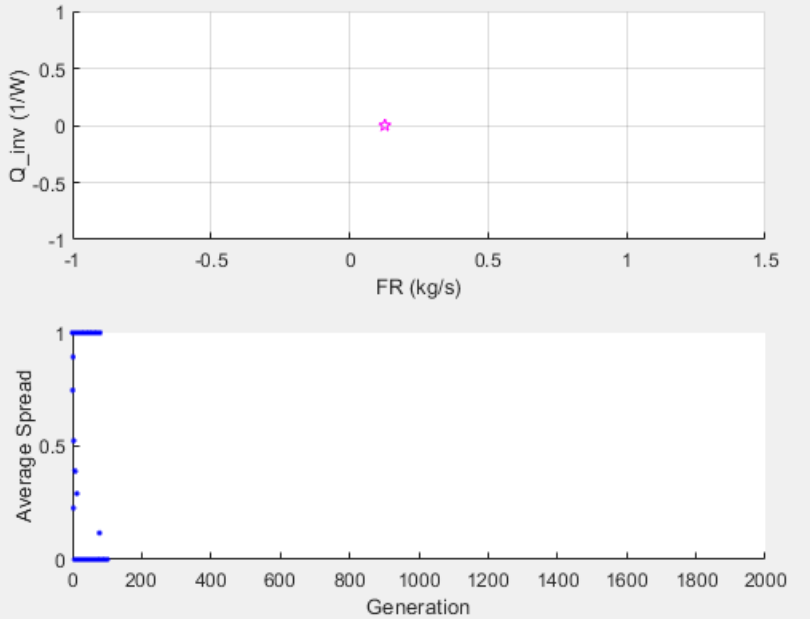

Figure 23. Pareto front and spread change results for case study 3 ($dP = 2Pa$).

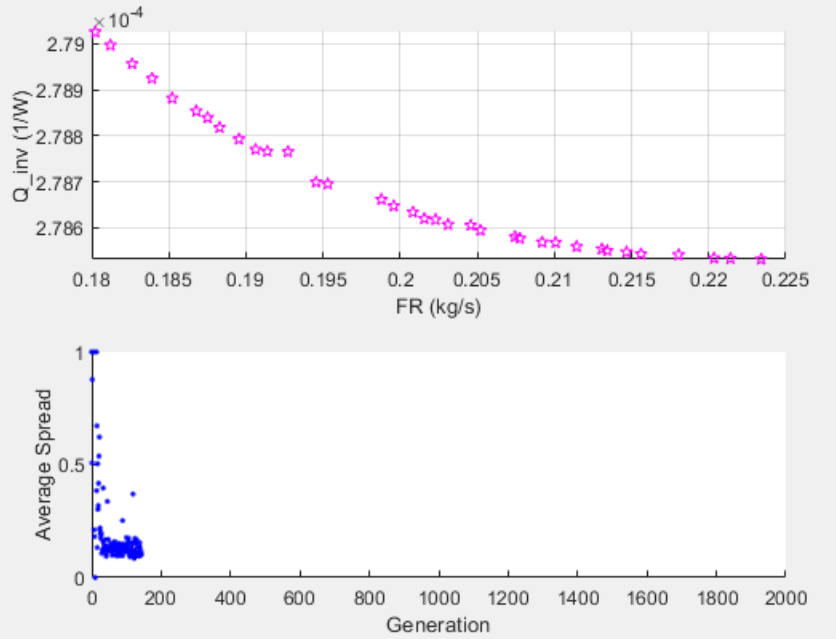

Figure 24. Pareto front and spread change results for case study 3 ($dP = 4Pa$).

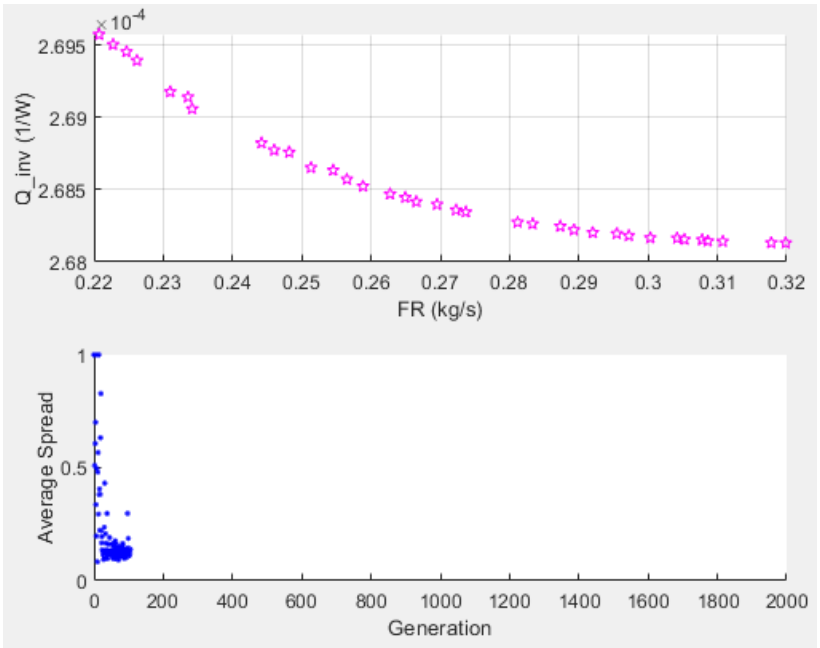

Figure 25. Pareto front and spread change results for case study 3 ($dP = 6Pa$).

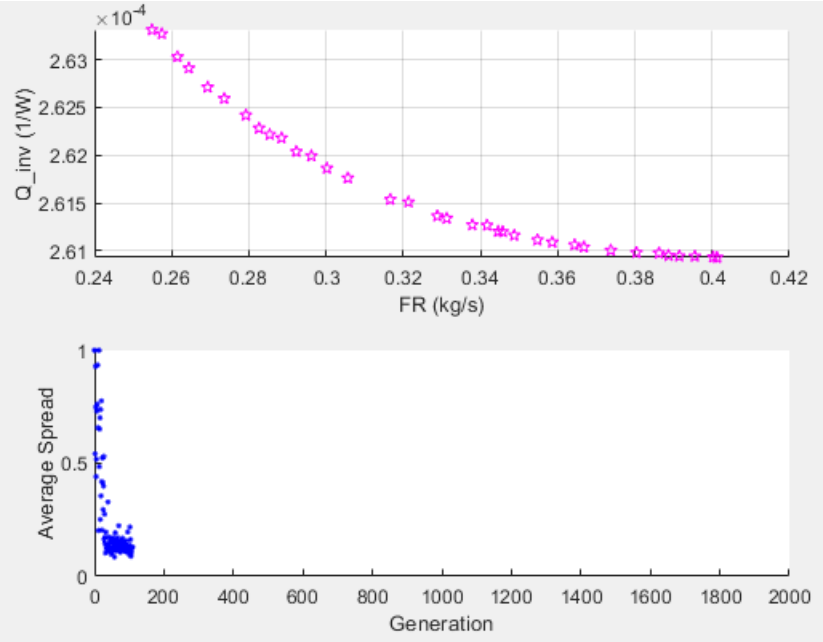

Figure 26. Pareto front and spread change results for case study 3 ($dP = 8Pa$).

To compare the results of case 3 for each pressure level, figures 27 shows the results aggregated onto a single plot. It is observed that as the constant pressure drop increases, the systems mass flow rate and total heat transfer also increase. This result makes sense because as the pressure drop increases, the flow rate of the system is expected to increase because of a higher mass flux. As a result, the total heat transfer of the system increases due to the increase in flow rate. It is also observed that for the different levels of pressure drop, the heat transfer of the system stays relatively constant as a function of mass flow rate. This result is not expected because the flow rate has a direct effect on the Reynold's number of the fluid. However, the range of parameters may be too small to illustrate a significant change. As a result, expanding the parameters of this study serves as a key area of interest for future investigation.

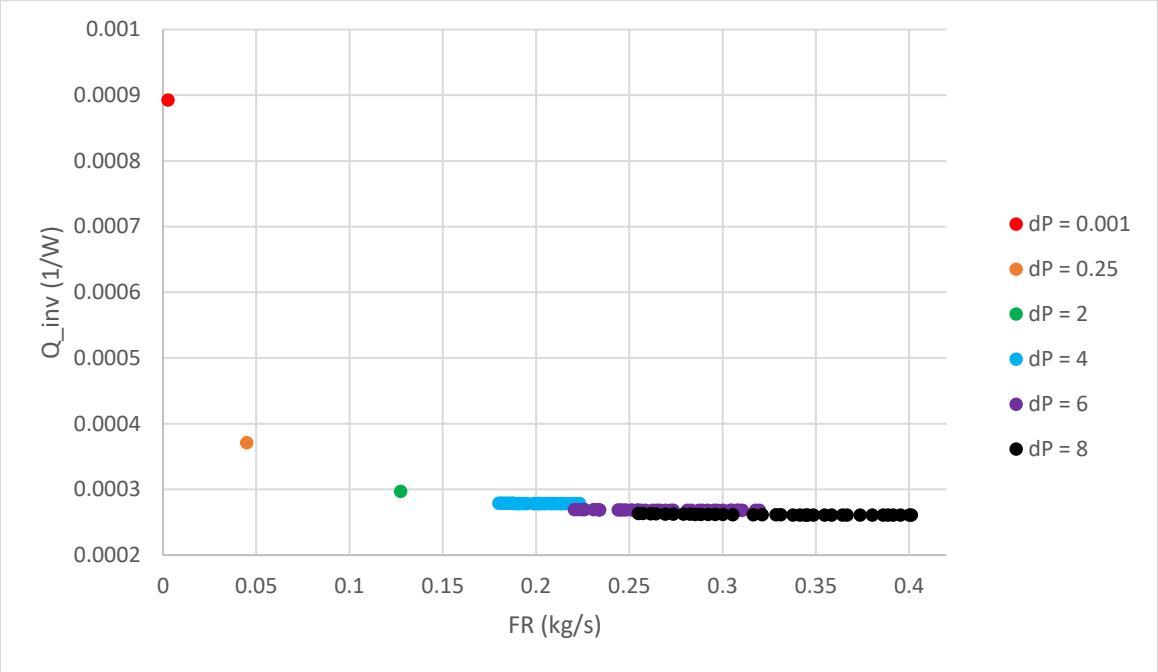

Figure 27. Aggregated Pareto front results for case study 3.

T compare the results of case study 3 to previous results, table 6 shows the parameters of designs closest to the utopia point at pressure drop levels of 0.001, 0.25, 6, and 8 Pa. Because previous design at the extreme points and closest to the utopia had similar levels of pressure drop, these results were selected to conduct a one-to-one comparison. For the cases with low pressure drop (0.001 and 0.25Pa), a significant increase in heat transfer was observed. However, because the mass flux was held constant for previous cases, the reason for this increase in performance is likely due to the optimal mass flow rate found in case study 3. As a result, this data provides evidence that by increasing the flow rate for a constant pressure drop system, the heat transfer is expected to increase significantly, which was expected. Furthermore, for the cases of higher pressure drop, the results from case 3 are consistent with the performance of previous cases. This is likely because at this level of pressure drop, the mass flux in these cases is

consistent to those presented previously. An examination of the effect of the mass flux and flow rate in combination with the system objectives serves as an additional area of interest for investigation. Finally, an observation made in this case study is that all the designs closest to the utopia point are identical. As a result, it is concluded that within the constraint of this study, a temperature profile does not significantly affect the placement of fin mass along the channel. However, by reducing the restrictions of the simulation and increasing the effect of the mass flux on the Reynold's number at higher level pressure drop, the objectives of mass flux and heat transfer would be expected to have a higher level of competition, and as result, a wider range of design diversity would be expected.

Table 6. Parameters of designs closest to utopia in case study 3.

$n_{fin,i}$	$r_{fin,i}$	$n_{fin,i}$	$r_{fin,i}$	$n_{fin,i}$	$r_{fin,i}$	$n_{fin,i}$	$r_{fin,i}$	$n_{fin,i}$	$r_{fin,i}$	$dP\ (Pa)$	$FR\ (kg/s)$	$Q\ (W)$	Distance to utopia point
19	0.1	19	0.1	19	0.1	19	0.1	19	0.1	0.001	0.00285	1119.979	0.002986
19	0.1	19	0.1	19	0.1	19	0.1	19	0.1	0.25	0.04505	2693.632	0.045210
19	0.1	19	0.1	19	0.1	19	0.1	19	0.1	6	0.220733	3709.513	0.220733
19	0.1	19	0.1	19	0.1	19	0.1	19	0.1	8	0.25489	3797.766	0.25489

Chapter 4

Discussion, Conclusions, and Future Work

This thesis investigates the optimal design of external annular fins on a circular channel using a multi-objective genetic algorithm. Through 3 case studies, it was found that increasing the design freedom of the simulation did not greatly increase the system's ability minimizing the pressure drop and maximizing the heat transfer of the system. Although adding fin thickness did not have a significant effect on the performance of the system, additional design objectives and parameters may be introduced to further investigate the effect of design freedom on heat exchanger performance. Additionally, it was found that introducing a nonuniform temperature profile did not shift the number of fins or fin radius toward sections of higher temperature. However, by reducing the constraints applied in this study, it is expected to see a higher level of design variance for optimal performance in a wider range of conditions. Finally, it was found that at varying levels of constant pressure drop, the heat transfer of the system was not significantly impacted by the mass flow rate. Although this is counter intuitive, the range of the parameters may have been too small to detect a significant change, and expanding the experimental conditions is desired. Despite these findings, there are multiple areas of investigation that may improve the fidelity of this work and future applications.

Looking at the methodology in this study, cases 1 and 2 assumed a constant mass flux for the air going into the system. Although this makes sense theoretically, in application, users would be interested in a constant mass flow rate value based on a given fan capacity. As a result, this study could be repeated using a mass flow rate to better simulate application of this work. Additionally, the accuracy of the flat plate correlation could be improved via a better way to approximate the distance (x) used to calculate the Reynold's number (Re_x) and Nusselt number

(Nu_x) in the model. Using the average flow length along the fin may be a potential solution. A more accurate but complex approach would be conducting further research into correlations that are designed for annular finned cylinders. This will allow the approximation for the heat transferred in the simulated system to be calculated more accurately compared to a physical system. An additional consideration for case studies is increasing the number of objectives of the algorithm to help differentiate the performance of the design parameters. One possible approach is varying the control volume of the channel. This would allow users to compare the effectiveness of short finned closely packed channels versus long finned sparsely packed channels.

Furthermore, several changes can be made to the algorithm to increase the accuracy of modeling various application environments. Assumptions about the fluid characteristics may be explored further. For example, the friction factor is assumed to be the values of air at room temperature, and the density of the fluid is assumed to remain constant as it travels through the system. Finding ways to approximate these values more precisely will produce more accurate results.

Appendix A

Nomenclature

Tables 7-9 define the nomenclature used for the constants, inputs, and variables used for the genetic algorithm, respectively. Each symbol has corresponding description and units associated with it. Table 10 lists the intermediate variables used in the code.

Table 7. Nomenclature defined for constants of algorithm.

Symbol	Description	Units
g	Gravitational acceleration	$\frac{m}{s^2}$
C_{cyl}	Constant for cylinder in cross flow (based on Reynold's number)	
m_{cyl}	Constant for cylinder in cross flow (based on Reynold's number)	
v_2	Specific volume of outlet	$\frac{m^3}{kg}$
$t_{fin,min}$	Minimum fin thickness	m
h_{CV}	Constant height and depth of control volume	m

Table 8. Nomenclature defined for inputs of algorithm.

Symbol	Description	Units
MP	Molar mass of fluid	
P_1	Inlet pressure	Pa
f	Fluid friction factor	
v_1	Specific volume of inlet	$\frac{m^3}{kg}$
G	Fluid mass flux	$\frac{kg}{s * m^2}$
Pr	Prandtl number of fluid	
T_b	Base temperature of channel	K
T_f	Fluid temperature	K
K_f	Thermal conductivity of fluid	$\frac{W}{m * K}$
K	Thermal conductivity of channel and fins	$\frac{W}{m * K}$
μ	Dynamic viscosity of fluid	$\frac{N * s}{m^2}$
$L_{channel}$	Length of channel	m
$d_{channel}$	Diameter of channel	m

Table 9. Nomenclature defined for variables of algorithm.

Symbol	Description	Units
$n_{fin,i}$	Number of fins	
$r_{fin,i}$	Radius of fins	m
$t_{fin,i}$	Thickness of fins	m

Table 10. Nomenclature defined for intermediate and final values of algorithm.

Symbol	Description	Units
d_{max}	Diameter of fins and channel	m
$A_{frontal}$	Frontal area of channel	m^2
A_{free}	Free flow area of channel	m^2
σ_A	Ratio of free flow area to frontal area	
$A_{s,channel}$	Surface area of channel only	m^2
A_s	Surface area of entire channel	m^2
P	Perimeter of fin	m
r_1	Radius of channel	m
r_2	Radius of channel and fin	m
r_{2c}	Radius of channel and fin with half of fin thickness	m
A_f	Surface area of fins	m^2
x	Distance used for flat plat Reynold's number	m
Re_D	Reynold's number based on diameter of channel	
Re_x	Flat plate Reynold's number	
dP	Pressure drop across channel	Pa
$h_{channel}$	Heat transfer coefficient of channel	$\frac{W}{m^2K}$
h_{fin}	Heat transfer coefficient of fin	$\frac{W}{m^2K}$
m	Constant for fin efficiency	
C_2	Constant for fin efficiency	
η	Fin efficiency	
$Q_{conv,fins}$	Convection heat transferred from fins	W
Q_{conv}	Convection heat transferred from system	W

Q_{inv}	Inverse heat transferred	$\frac{1}{W}$
dP_{min}	Minimum pressure drop in design space	Pa
dP_{max}	Maximum pressure drop in design space	Pa
$Q_{inv,min}$	Minimum inverse heat transferred in design space	$\frac{1}{W}$
$Q_{inv,max}$	Maximum inverse heat transferred in design space	$\frac{1}{W}$
dP_{norm}	Normalized pressure drop	
$Q_{inv,norm}$	Normalized heat transferred	
FR	Mass flow rate	$\frac{kg}{s}$
FR_{system}	Mass flow rate of whole system	$\frac{kg}{s}$
P_{fan}	Power used by the fan	W
$Q_{inv,system}$	Inverse heat transferred from entire system	

Appendix B

MATLAB Programs

This appendix presents the MATLAB script that was used to produce the results in this study. Multiple scripts were written to create functions that would be called on by the driver script. The following is the order of scripts: driver scripts and fitness function for each case followed by the visualization script and its driver code.

```
% This code investigates the optimal fin design for two-phase flow channel
% Written by: Anthony Luu

clear all
close all
clc

nvars = 2; %number of variables


% Constants
f = 0.0064; % Air side friction factor (from example)
v_1 = 0.816; %m^3/kg; specific volume of inlet (generic air)
v_2 = 0.816; %m^3/kg; specific volume of outlet (generic air)
G = 10; %kg/(s*m^2); air side mass flux (arbitrary)

% % Application Range Pr >= 0.7
% Pr = input('Input value for Pr of fluid (equal or larger than 0.7) = ');
% while Pr < 0.7 % fixing the applicability error
%     Pr = input('Input a valid value for Pr of fluid (equal or larger than 0.7) = ');
% end

Pr = 0.7; %no units; Prandtl number of air (arbitrary)
T_b = 373.15; %K; base temperature of channel (arbitrary)
T_f = 293; %K; fluid temperature (arbitrary)
k_f = 26.3*(10^-3); %W/mK; thermal conductivity of fluid (arbitrary)
k = 237; %W/mK; thermal conductivity of fin (arbitrary Al)
mu = 185*(10^-7); %Ns/m^2; dynamic visocisty of fluid
t_fin = 0.001 ; %m; thickness of fins (arbitrary)
L_channel = 0.2; %m; length of channel (arbitrary)
d_channel = 0.01; %m; diameter of channel (arbitrary)

% % Application Range 0.4 < Re < 400000
% G = input('Input value for G of fluid = ');
% mu = input('Input value for mu of fluid = ');
% d_channel = input('Input value for d_channel of HE = ');
% d_max = d_channel; %m; smallest max diamater of fins on channel
```

```
% d_max2 = (2*0.1)+ d_channel; %m; largest max diamater of fins on channel
% x1 = d_max;
% x2 = d_max2-d_channel; %m; length for flat plate correlation
% %Reynolds number calculation
% Re_x1 = (G*x1)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
% Re_x2 = (G*x2)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
% while Re_x1 < 0.4 || Re_x1 > 400000 || Re_x2 < 0.4 || Re_x2 > 400000
%     G = input('Input valid value for G of fluid (Re is out of range) = ');
%     mu = input('Input valid value for mu of fluid (Re is out of range) = ');
%     d_channel = input('Input valid value for d_channel of HE (Re is out of range) = ');
%     d_max = d_channel; %m; smallest max diamater of fins on channel
%     d_max2 = (2*0.1)+ d_channel; %m; largest max diamater of fins on channel
%     x1 = d_max;
%     x2 = d_max2-d_channel; %m; length for flat plate correlation
%     %Reynolds number recalculation
%     Re_x1 = (G*x1)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
%     Re_x2 = (G*x2)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
% end

%asking for normalized or unormalized results)
norm = input("Present results normalized (1) or unnormalized (2)? ");
if norm ~= 1 && norm ~=2
    norm = 2;
end


consts = [
    f
    v_1
    v_2
    G
    Pr
    T_b
    T_f
    k_f
    k
    mu
    t_fin
    L_channel
    d_channel
    norm
    ];


% Define values for consts
fitnessfcn = @(params) Case1_Fin_fitness(params, consts);
options = optimoptions('gamultiobj','PlotFcn',{@gaplotpareto,@gaplotspread},"PopulationSize",100,
'FunctionTolerance',0.0001); %,@gaplotscores
% Variables
% params = [
%    n_fin_i %number of fins
%    r_fin_i %length of fins
%    ];
```

```
% Constraints
A = [];
%A = [1 0;
%     0 1];
%b = [
%    10*d_channel;
%     L_channel/(2*t_fin)];
b = [];
Aeq = [];
beq = [];
lb = [0 0.1*d_channel]; %lower bound constraint, fin radius set to one channel diameter
ub1 = (L_channel - t_fin)/(2*t_fin);
ub2 = 10*d_channel; %10*d_channel
ub = [ub1 ub2]; %upper bounder constraint
nonlcon = [];
intcon = [1]; %constaining nfin to integers
[x,fval] = gamultiobj(fitnessfcn,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options);

P1 = subplot(2,1,1);
P2 = subplot(2,1,2);
% P3 = subplot(3,1,3);

%normalzation condition for axis
if norm == 1
    xlabel(P1,'dP_norm')
    ylabel(P1,'Q_inv_norm')
else
    xlabel(P1,'dP (Pa)') %units for unnormalized results
    ylabel(P1,'Q_inv (1/W)')  %units for unnormalized results
end
title(P1,'')
title(P2,'')
```

```
% This is a function to evaluate the parameters of the genetic algorithm as
% the variables are manipulated. The two objectives of the program consists
% of the inverse of total heat transferred by the fins and the external
% pressure drop of the system

%Every parameter that says in example or arbitrary must be changed for
%application results

function [out] = Case1_Fin_fitness(params, consts)



format long

% System constants
g = 9.81; %m/(s^2); gravitational acceleration
C_constant = 0.193; %no units; cylinder cross flow constant (arbitrary)
m_constant = 0.618; %no units; cylinder cross flow constant (arbitrary)
%MP_1 and P_1 are irrelevant
```

```
MP_1 = 144; %no units; molar mass of fluid (in example)
P_1 = 275000*MP_1; %Pa = N/m^2; Inlet pressure multiplied by MP1 (in example)

% Fluid properties
f = consts(1); % Air side friction factor (from example)
% TO DO:
%1: for specific volume, find way to incorporate iterative solver
% assuming value for pressure at end of channel
%2: incorporate equations to calculate specific volume (assuming same
%specific volume for now)
%3: find better way of approximating v_m
v_1 = consts(2); %m^3/kg; specific volume of inlet (generic air)
v_2 = consts(3); %m^3/kg; specific volume of outlet (generic air)
v_m = (v_2+v_1)/2; %m^3/kg; application specific volume (approximated by log mean absolute
temperature) (in example)
G = consts(4); %kg/(s*m^2); air side mass flux (in example)
%A/Ac = L/r_h; --> not valid for our application
Pr = consts(5); %no units; Prandtl number of air (arbitrary)
T_b = consts(6); %K; base temperature of channel (arbitrary)
T_f = consts(7); %K; fluid temperature (arbitrary)
k_f = consts(8); %W/mK; thermal conductivity of fluid (arbitrary)
k = consts(9); %W/mK; thermal conductivity of fin (arbitrary Al)
mu = consts(10); %Ns/m^2; dynamic visocisty of fluid

% Channel geometric properties
t_fin = consts(11) ; %m; thickness of fins (arbitrary)
L_channel = consts(12); %m; length of channel (arbitrary)
d_channel = consts(13); %m; diameter of channel (arbitrary)

%normalization condition
norm = consts(14);

%n_fin_i = input('Number of fins: '); %no units; number of fins on channel (independent variable)
n_fin_i = params(1);
%if n_fin_i*t_fin*2 > L_channel %accounting for too many fins
%    n_fin_i = input('Input new value for number of fins: ');
%else
%    n_fin_i = n_fin_i;
%end
%L_fin_i = input('Length of fins(m)(multiplied by channel diameter): ')*d_channel; %m; length of
fins (independent variable)
r_fin_i = params(2);
if n_fin_i == 0 || r_fin_i == 0 % To correct for having no fins
    d_max = d_channel; %m; max diamater of fins on channel
else
    d_max = (2*r_fin_i)+ d_channel; %m; max diamater of fins on channel
end
% Adjust this late for varying CV size
%A_frontal = L_channel*d_max; %m^2; frontal area
%if A_frontal < L_channel*3*d_channel % adjusting for minimum CV area
%    A_frontal = L_channel*3*d_channel;
%else
%    A_frontal = A_frontal;
```

```
%end
% sub in 0.23 for d_max CV variables (high case for parameter test)
A_frontal = L_channel*((2*11*d_channel)+d_channel); %m^2; constant frontal area (CV size)
A_free = A_frontal-((n_fin_i*t_fin*d_max)+(((L_channel-(n_fin_i*t_fin))*d_channel))); %m^2; free
flow area
sigma_A = A_free/A_frontal; %no units; Free flow area / frontal area of (example)
if r_fin_i == 0 % fixing fin length error
    A_s_channel = pi*d_channel*(L_channel); %m^2; surface area of channel only (excluding fins)
else
    A_s_channel = pi*d_channel*(L_channel-(n_fin_i*t_fin)); %m^2; surface area of channel only
(excluding fins)
end
if r_fin_i == 0 % fixing fin length error
    A_surf = pi*d_channel*(L_channel); %m^2; surface area of channel only (including 0 fins)
else
    A_surf = (pi*d_channel*(L_channel-(n_fin_i*t_fin))) +
(n_fin_i*t_fin*pi*(d_channel+(2*r_fin_i))) + (2*n_fin_i*pi*(((r_fin_i + (d_channel))^2)-
((d_channel)^2))); %m^2; surface area of channel and fins (including fins)
end
P = 2*pi*d_max; %m; perimeter of a fin
r_1 = d_channel/2; %m; radius of channel
r_2c = (d_max/2) + (t_fin/2); %m; radius of channel including fins
A_f = 2*pi*((r_2c^2)-(r_1^2)); %m^2; surface area of fins
if d_max == d_channel %accounting if max diameter is same as channel diameter
    x = d_max;
else
    x = d_max-d_channel; %m; length for flat plate correlation
end

%Reynolds number calculation
Re_D = (G*d_channel)/mu; %no units; Reynolds number of fluid based on the diameter (arbitrary
laminar)
Re_x = (G*x)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)

%Pressure drop equation
%dP = deltaP/P1
dP =((G^2)/(2*g))*(v_1/P_1)*[(1+(sigma_A^2))*((v_2/v_1)-1)+ (f*(A_surf/A_free)*(v_m/v_1))]; %Pa;
same as P1
deltaP = abs(dP * P_1);

% Heat transfer equation
h_channel = (C_constant*(Re_D^m_constant)*(Pr^(1/3))*k_f)/d_channel; %W/m^2*K; average heat
transfer coefficient of channel in cross flow (excluding fins)
q_conv_channel = h_channel*A_s_channel*(T_b-T_f); %W; total heat transferred by channel
(excluding fins)
if r_fin_i == 0 %correcting for 0 fin length
    h_fin = 0;
else
    h_fin = (0.664*(Re_x^(1/2))*(Pr^(1/3))*k_f)/x;  %W/m^2*K; average heat transfer coefficient
for one fin in parallel plate flow
end
m = sqrt((h_fin*2)/(t_fin*k)); %no units; constant for fin efficiency
if r_1 == r_2c % fixing undefined error
```

```
    C_2 = 0;
else
    C_2 = (2*r_1/m)/((r_2c^2)-(r_1^2)); %no units; constant for fin efficiency
end
if C_2 == 0 || r_fin_i == 0 || n_fin_i == 0 % fixing undefined error
    eta_fin = 0;
else
    eta_fin = (C_2*((besselk(1,m*r_1)*besseli(1,m*r_2c))-
(besseli(1,m*r_1)*besselk(1,m*r_2c))))/((besseli(0,m*r_1)*besselk(1,m*r_2c))+(besselk(0,m*r_1)*be
sseli(1,m*r_2c))); %no units; fin efficiency
end
q_conv_fins = n_fin_i*eta_fin*A_f*h_fin*(T_b-T_f); %W; total heat transferred by fins
q_conv = q_conv_channel+q_conv_fins;
q_inv = 1/q_conv;


fval = [deltaP q_inv]; %outputs the pressure drop and inverse heat transferred (unnormalized
results)
P_min = 0.003801003955636; % Pa from no fins
P_max = 8.285497531658235; % Pa from max fins and max fin size
Q_inv_min = 2.535811493740514e-04; %1/W from max fins and max fin size
Q_inv_max =  0.021734500426815; %1/W from no fins
deltaPnorm = (deltaP - P_min)/ (P_max - P_min); %normalize dP
q_invnorm = (q_inv - Q_inv_min)/(Q_inv_max-Q_inv_min); %normalize Q_inv
outnorm = [deltaPnorm q_invnorm]; %outputs the pressure drop and inverse heat transferred
normalized results

%normalization condition
if norm == 1
    [out] = [outnorm];
else
    [out] = [fval];
end

%{
the following section is calculations for L_flow and r_h (only relavant for
rectangular flow channels)
if L_fin_i == 0 % Correcting for no fin length
    %P_wet = 2*(L_channel)+(2*(d_max-d_channel)); % m; wetting perimeter of the fluid flow cross
section
    P_wet = 2*(L_channel)+(2*(0.21-d_channel)); % m; wetting perimeter of the fluid flow cross
section
else
    %P_wet = (2*(L_channel-(n_fin_i*t_fin))+(L_fin_i*n_fin_i*4))+(2*(d_max-d_channel)); % m;
wetting perimeter of the fluid flow cross section
    P_wet = (2*(L_channel-(n_fin_i*t_fin))+(L_fin_i*n_fin_i*4))+(2*(0.21-d_channel)); % m;
wetting perimeter of the fluid flow cross section
end
r_h = A_free/P_wet; %m; flow passage hydraulic radius / diameter
%L_flow = d_max; %m; flow depth
L_flow = 0.21; %m; flow depth
%}
```

```
% This code investigates the optimal fin design for two-phase flow channel
% Written by: Anthony Luu

clear all
close all
clc

nvars = 3; %number of variables


% Constants
f = 0.0064; % Air side friction factor (from example)
v_1 = 0.816; %m^3/kg; specific volume of inlet (generic air)
v_2 = 0.816; %m^3/kg; specific volume of outlet (generic air)
G = 10; %kg/(s*m^2); air side mass flux (arbitrary)

% % Application Range Pr >= 0.7
% Pr = input('Input value for Pr of fluid (equal or larger than 0.7) = ');
% while Pr < 0.7 % fixing the applicability error
%     Pr = input('Input a valid value for Pr of fluid (equal or larger than 0.7) = ');
% end

Pr = 0.7; %no units; Prandtl number of air (arbitrary)
T_b = 373.15; %K; base temperature of channel (arbitrary)
T_f = 293; %K; fluid temperature (arbitrary)
k_f = 26.3*(10^-3); %W/mK; thermal conductivity of fluid (arbitrary)
k = 237; %W/mK; thermal conductivity of fin (arbitrary Al)
mu = 185*(10^-7); %Ns/m^2; dynamic visocisty of fluid
t_fin = 0.001 ; %m; thickness of fins (arbitrary)
L_channel = 0.2; %m; length of channel (arbitrary)
d_channel = 0.01; %m; diameter of channel (arbitrary)

% % Application Range 0.4 < Re < 400000
% G = input('Input value for G of fluid = ');
% mu = input('Input value for mu of fluid = ');
% d_channel = input('Input value for d_channel of HE = ');
% d_max = d_channel; %m; smallest max diamater of fins on channel
% d_max2 = (2*0.1)+ d_channel; %m; largest max diamater of fins on channel
% x1 = d_max;
% x2 = d_max2-d_channel; %m; length for flat plate correlation
% %Reynolds number calculation
% Re_x1 = (G*x1)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
% Re_x2 = (G*x2)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
% while Re_x1 < 0.4 || Re_x1 > 400000 || Re_x2 < 0.4 || Re_x2 > 400000
%     G = input('Input valid value for G of fluid (Re is out of range) = ');
%     mu = input('Input valid value for mu of fluid (Re is out of range) = ');
%     d_channel = input('Input valid value for d_channel of HE (Re is out of range) = ');
%     d_max = d_channel; %m; smallest max diamater of fins on channel
%     d_max2 = (2*0.1)+ d_channel; %m; largest max diamater of fins on channel
%     x1 = d_max;
%     x2 = d_max2-d_channel; %m; length for flat plate correlation
%     %Reynolds number recalculation
%     Re_x1 = (G*x1)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
```

```
%     Re_x2 = (G*x2)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
% end

%asking for normalized or unormalized results)
norm = input("Present results normalized (1) or unnormalized (2)? ");
if norm ~= 1 && norm ~=2
    norm = 2;
end


consts = [
    f
    v_1
    v_2
    G
    Pr
    T_b
    T_f
    k_f
    k
    mu
    L_channel
    d_channel
    norm
    ];


% Define values for consts
fitnessfcn = @(params) Case2_Fin_fitness(params, consts);
% options =
optimoptions('gamultiobj','PlotFcn',{@gaplotpareto,@gaplotspread},"PopulationSize",100,
'FunctionTolerance',0.0001); %,@gaplotscores
options = optimoptions('gamultiobj','PlotFcn',{@gaplotpareto,@gaplotspread},"PopulationSize",100,
'FunctionTolerance',0.0001,'CreationFcn',{@gacreationnonlinearfeasible,...
    'UseParallel',true,'NumStartPts',20}); %,@gaplotscores


% Constraints
A = [];
%A = [1 0;
%     0 1];
%b = [
%    10*d_channel;
%     L_channel/(2*t_fin)];
b = [];
Aeq = [];
beq = [];
tmin = 0.001; %1mm minimum thickness
lb = [0 0.1*d_channel tmin]; %lower bound constraint, fin radius set to one channel diameter,
thickness set to 1mm
ub1 = 99;
%ub1 = 10;
ub2 = 10*d_channel; %10*d_channel
```

```
ub3 = L_channel; %length of channel
%ub3 = L_channel/10; %length of channel
ub = [ub1 ub2 ub3]; %upper bounder constraint
nonlcon = [];
intcon = [1]; %constaining nfin to integers
[x,fval] = gamultiobj(fitnessfcn,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options);

P1 = subplot(2,1,1);
P2 = subplot(2,1,2);
% P3 = subplot(3,1,3);

%normalzation condition for axis
if norm == 1
    xlabel(P1,'dP_norm')
    ylabel(P1,'Q_inv_norm')
else
    xlabel(P1,'dP (Pa)') %units for unnormalized results
    ylabel(P1,'Q_inv (1/W)')  %units for unnormalized results
end

title(P1,'')
title(P2,'')
```

```
% This is a function to evaluate the parameters of the genetic algorithm as
% the variables are manipulated. The two objectives of the program consists
% of the inverse of total heat transferred by the fins and the external
% pressure drop of the system

%Every parameter that says in example or arbitrary must be changed for
%application results

function [out] = Case2_Fin_fitness(params, consts)



format long

% System constants
g = 9.81; %m/(s^2); gravitational acceleration
C_constant = 0.193; %no units; cylinder cross flow constant (arbitrary)
m_constant = 0.618; %no units; cylinder cross flow constant (arbitrary)
%MP_1 and P_1 are irrelevant
MP_1 = 144; %no units; molar mass of fluid (in example)
P_1 = 275000*MP_1; %Pa = N/m^2; Inlet pressure multiplied by MP1 (in example)

% Fluid properties
f = consts(1); % Air side friction factor (from example)
% TO DO:
%1: for specific volume, find way to incorporate iterative solver
% assuming value for pressure at end of channel
%2: incorporate equations to calculate specific volume (assuming same
%specific volume for now)
%3: find better way of approximating v_m
```

```
v_1 = consts(2); %m^3/kg; specific volume of inlet (generic air)
v_2 = consts(3); %m^3/kg; specific volume of outlet (generic air)
v_m = (v_2+v_1)/2; %m^3/kg; application specific volume (approximated by log mean absolute
temperature) (in example)
G = consts(4); %kg/(s*m^2); air side mass flux (in example)
%A/Ac = L/r_h; --> not valid for our application
Pr = consts(5); %no units; Prandtl number of air (arbitrary)
T_b = consts(6); %K; base temperature of channel (arbitrary)
T_f = consts(7); %K; fluid temperature (arbitrary)
k_f = consts(8); %W/mK; thermal conductivity of fluid (arbitrary)
k = consts(9); %W/mK; thermal conductivity of fin (arbitrary Al)
mu = consts(10); %Ns/m^2; dynamic visocisty of fluid

% Channel geometric properties
L_channel = consts(11); %m; length of channel (arbitrary)
d_channel = consts(12); %m; diameter of channel (arbitrary)

%normalization condition
norm = consts(13);

%Invepended variables
%n_fin_i = input('Number of fins: '); %no units; number of fins on channel (independent variable)
n_fin_i = params(1);
%if n_fin_i*t_fin*2 > L_channel %accounting for too many fins
%    n_fin_i = input('Input new value for number of fins: ');
%else
%    n_fin_i = n_fin_i;
%end
%L_fin_i = input('Length of fins(m)(multiplied by channel diameter): ')*d_channel; %m; length of
fins (independent variable)
r_fin_i = params(2);
t_fin_i = params(3) ; %m; thickness of fins (arbitrary)

% % Penalize designs that violate geometric constraints
% tfin_new=0;
% nfin_new=0;
% Convergence_tol=1e-20;
% Error=1;
% tmin=0.001;%(m)
% while Error>Convergence_tol
%     nfin_new=floor((L_channel-(n_fin_i+1)*tmin)/t_fin_i);
%     if nfin_new < 0
%         nfin_new = 0;
%     end
%     tfin_new=(L_channel-(n_fin_i+1)*tmin)/n_fin_i;
%     if tfin_new < 0.001
%         tfin_new = 0.001;
%     end
%     if nfin_new == 0
%         tfin_new = 0.2;
%     end
%     if nfin_new == 1
%         tfin_new = L_channel;
```

```
%     end
%     if nfin_new > 99
%         nfin_new = 99;
%     end
%     if tfin_new > 0.2
%         tfin_new = 0.2;
%     end
%     if nfin_new >= n_fin_i
%         nfin_new = n_fin_i;
%     end
%     if tfin_new >= t_fin_i
%         tfin_new = t_fin_i;
%     end
%     Error=sqrt((tfin_new-t_fin_i)^2+(nfin_new-n_fin_i)^2);
%
%     t_fin_i=tfin_new;
%     n_fin_i=nfin_new;
% end

if n_fin_i == 0 || r_fin_i == 0 % To correct for having no fins
    d_max = d_channel; %m; max diamater of fins on channel
else
    d_max = (2*r_fin_i)+ d_channel; %m; max diamater of fins on channel
end

% Adjust this late for varying CV size
%A_frontal = L_channel*d_max; %m^2; frontal area
%if A_frontal < L_channel*3*d_channel % adjusting for minimum CV area
%    A_frontal = L_channel*3*d_channel;
%else
%    A_frontal = A_frontal;
%end
% sub in 0.21 for d_max CV variables (high case for parameter test)
A_frontal = L_channel*((2*11*d_channel)+d_channel); %m^2; constant frontal area (CV size)
A_free = A_frontal-((n_fin_i*t_fin_i*d_max)+(((L_channel-(n_fin_i*t_fin_i))*d_channel))); %m^2;
free flow area
sigma_A = A_free/A_frontal; %no units; Free flow area / frontal area of (example)
if r_fin_i == 0 % fixing fin length error
    A_s_channel = pi*d_channel*(L_channel); %m^2; surface area of channel only (excluding fins)
else
    A_s_channel = pi*d_channel*(L_channel-(n_fin_i*t_fin_i)); %m^2; surface area of channel only
(excluding fins)
end
if r_fin_i == 0 % fixing fin length error
    A_surf = pi*d_channel*(L_channel); %m^2; surface area of channel only (including 0 fins)
else
    A_surf = (pi*d_channel*(L_channel-(n_fin_i*t_fin_i))) +
(n_fin_i*t_fin_i*pi*(d_channel+(2*r_fin_i))) + (2*n_fin_i*pi*(((r_fin_i + (d_channel))^2)-
((d_channel)^2))); %m^2; surface area of channel and fins (including fins)
end
P = 2*pi*d_max; %m; perimeter of a fin
r_1 = d_channel/2; %m; radius of channel
r_2c = (d_max/2) + (t_fin_i/2); %m; radius of channel including fins
```

```
A_f = 2*pi*((r_2c^2)-(r_1^2)); %m^2; surface area of fins
if d_max == d_channel %accounting if max diameter is same as channel diameter
    x = d_max;
else
    x = d_max-d_channel; %m; length for flat plate correlation
end

%Reynolds number calculation
Re_D = (G*d_channel)/mu; %no units; Reynolds number of fluid based on the diameter (arbitrary
laminar)
Re_x = (G*x)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)

%Pressure drop equation
%dP = deltaP/P1
dP =((G^2)/(2*g))*(v_1/P_1)*[(1+(sigma_A^2))*((v_2/v_1)-1)+ (f*(A_surf/A_free)*(v_m/v_1))]; %Pa;
same as P1
deltaP = abs(dP * P_1);

% Heat transfer equation
h_channel = (C_constant*(Re_D^m_constant)*(Pr^(1/3))*k_f)/d_channel; %W/m^2*K; average heat
transfer coefficient of channel in cross flow (excluding fins)
q_conv_channel = h_channel*A_s_channel*(T_b-T_f); %W; total heat transferred by channel
(excluding fins)
if r_fin_i == 0 %correcting for 0 fin length
    h_fin = 0;
else
    h_fin = (0.664*(Re_x^(1/2))*(Pr^(1/3))*k_f)/x;  %W/m^2*K; average heat transfer coefficient
for one fin in parallel plate flow
end
m = sqrt((h_fin*2)/(t_fin_i*k)); %no units; constant for fin efficiency
if r_1 == r_2c % fixing undefined error
    C_2 = 0;
else
    C_2 = (2*r_1/m)/((r_2c^2)-(r_1^2)); %no units; constant for fin efficiency
end
if C_2 == 0 || r_fin_i == 0 || n_fin_i == 0 % fixing undefined error
    eta_fin = 0;
else
    eta_fin = (C_2*((besselk(1,m*r_1)*besseli(1,m*r_2c))-
(besseli(1,m*r_1)*besselk(1,m*r_2c))))/((besseli(0,m*r_1)*besselk(1,m*r_2c))+(besselk(0,m*r_1)*be
sseli(1,m*r_2c))); %no units; fin efficiency
end
q_conv_fins = n_fin_i*eta_fin*A_f*h_fin*(T_b-T_f); %W; total heat transferred by fins
q_conv = q_conv_channel+q_conv_fins;
q_inv = 1/q_conv;

%Penalizing violating the geometric bounds.
t_fin_min = 0.001; %m; minimum fin thickness
if n_fin_i > floor((L_channel-((n_fin_i+1)*t_fin_min))/t_fin_i)
    deltaP = inf; %penalizing pressure drop
    q_inv = inf; %penalizing inverse heat transfer
end
```

```
if t_fin_i > (L_channel-((n_fin_i+1)*t_fin_min))/n_fin_i
    deltaP = inf; %penalizing pressure drop
    q_inv = inf; %penalizing inverse heat transfer
end

fval = [deltaP q_inv]; %outputs the pressure drop and inverse heat transferred (unnormalized
results)
P_min = 0.003801003955636; % Pa from no fins
P_max = 8.355240195140221; % Pa from max fins and max fin size
Q_inv_min = 2.518698602827820e-04; %1/W from max fins and max fin size
Q_inv_max =  0.021734500426815; %1/W from no fins
deltaPnorm = (deltaP - P_min)/ (P_max - P_min); %normalize dP
q_invnorm = (q_inv - Q_inv_min)/(Q_inv_max-Q_inv_min); %normalize Q_inv
outnorm = [deltaPnorm q_invnorm]; %outputs the pressure drop and inverse heat transferred
normalized results

%normalization condition
if norm == 1
    [out] = [outnorm];
else
    [out] = [fval];
end

%{
the following section is calculations for L_flow and r_h (only relavant for
rectangular flow channels)
if L_fin_i == 0 % Correcting for no fin length
    %P_wet = 2*(L_channel)+(2*(d_max-d_channel)); % m; wetting perimeter of the fluid flow cross
section
    P_wet = 2*(L_channel)+(2*(0.21-d_channel)); % m; wetting perimeter of the fluid flow cross
section
else
    %P_wet = (2*(L_channel-(n_fin_i*t_fin))+(L_fin_i*n_fin_i*4))+(2*(d_max-d_channel)); % m;
wetting perimeter of the fluid flow cross section
    P_wet = (2*(L_channel-(n_fin_i*t_fin))+(L_fin_i*n_fin_i*4))+(2*(0.21-d_channel)); % m;
wetting perimeter of the fluid flow cross section
end
r_h = A_free/P_wet; %m; flow passage hydraulic radius / diameter
%L_flow = d_max; %m; flow depth
L_flow = 0.21; %m; flow depth
%}
```

```
% This code investigates the optimal fin design for two-phase flow channel
% Written by: Anthony Luu

clear all
close all
clc

nvars = 10; %number of variables


% Constants
```

```
f = 0.0064; % Air side friction factor (from example)
v_1 = 0.816; %m^3/kg; specific volume of inlet (generic air)
v_2 = 0.816; %m^3/kg; specific volume of outlet (generic air)
G = 10; %kg/(s*m^2); air side mass flux (arbitrary)

% % Application Range Pr >= 0.7
% Pr = input('Input value for Pr of fluid (equal or larger than 0.7) = ');
% while Pr < 0.7 % fixing the applicability error
%     Pr = input('Input a valid value for Pr of fluid (equal or larger than 0.7) = ');
% end

Pr = 0.7; %no units; Prandtl number of air (arbitrary)
T_b1 = 314.62; %K; base temperature of channel (arbitrary)
T_b2 = 343.89; %K; base temperature of channel (arbitrary)
T_b3 = 373.15; %K; base temperature of channel (arbitrary)
T_b4 = 402.41; %K; base temperature of channel (arbitrary)
T_b5 = 431.67; %K; base temperature of channel (arbitrary)
T_f = 293; %K; fluid temperature (arbitrary)
k_f = 26.3*(10^-3); %W/mK; thermal conductivity of fluid (arbitrary)
k = 237; %W/mK; thermal conductivity of fin (arbitrary Al)
mu = 185*(10^-7); %Ns/m^2; dynamic visocisty of fluid
%FR_system = 0.39; %kg/s; flow rate of the system
t_fin = 0.001 ; %m; thickness of fins (arbitrary)
deltaP = 8; %Pa; Pressure drop of system
L_channel = 0.2; %m; length of channel (arbitrary)
d_channel = 0.01; %m; diameter of channel (arbitrary)

% % Application Range 0.4 < Re < 400000
% G = input('Input value for G of fluid = ');
% mu = input('Input value for mu of fluid = ');
% d_channel = input('Input value for d_channel of HE = ');
% d_max = d_channel; %m; smallest max diamater of fins on channel
% d_max2 = (2*0.1)+ d_channel; %m; largest max diamater of fins on channel
% x1 = d_max;
% x2 = d_max2-d_channel; %m; length for flat plate correlation
% %Reynolds number calculation
% Re_x1 = (G*x1)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
% Re_x2 = (G*x2)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
% while Re_x1 < 0.4 || Re_x1 > 400000 || Re_x2 < 0.4 || Re_x2 > 400000
%     G = input('Input valid value for G of fluid (Re is out of range) = ');
%     mu = input('Input valid value for mu of fluid (Re is out of range) = ');
%     d_channel = input('Input valid value for d_channel of HE (Re is out of range) = ');
%     d_max = d_channel; %m; smallest max diamater of fins on channel
%     d_max2 = (2*0.1)+ d_channel; %m; largest max diamater of fins on channel
%     x1 = d_max;
%     x2 = d_max2-d_channel; %m; length for flat plate correlation
%     %Reynolds number recalculation
%     Re_x1 = (G*x1)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
%     Re_x2 = (G*x2)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
% end

%asking for normalized or unormalized results)
norm = input("Present results normalized (1) or unnormalized (2)? ");
```

```
if norm ~= 1 && norm ~=2
    norm = 2;
end


consts = [
    f
    v_1
    v_2
    Pr
    T_b1
    T_b2
    T_b3
    T_b4
    T_b5
    T_f
    k_f
    k
    mu
    t_fin
    deltaP
    L_channel
    d_channel
    norm
    ];


% Define values for consts
fitnessfcn = @(params) Case3b_Fin_fitness(params, consts);
% options =
optimoptions('gamultiobj','PlotFcn',{@gaplotpareto,@gaplotspread},"PopulationSize",100,
'FunctionTolerance',0.0001); %,@gaplotscores
options = optimoptions('gamultiobj','PlotFcn',{@gaplotpareto,@gaplotspread},"PopulationSize",100,
'FunctionTolerance',0.0001,'CreationFcn',{@gacreationnonlinearfeasible,...
    'UseParallel',true,'NumStartPts',20}); %,@gaplotscores


% Constraints
A = [];
%A = [1 0;
%     0 1];
%b = [
%    10*d_channel;
%     L_channel/(2*t_fin)];
b = [];
Aeq = [];
beq = [];
lb1 = 0; %minimum number of fins
lb2 = 0.1*d_channel; % minimum radius of fins
lb = [lb1 lb1 lb1 lb1 lb1 lb2 lb2 lb2 lb2 lb2]; %lower bound constraint, fin radius set to one
channel diameter, thickness set to 1mm
ub1 = 19; %max number of fins
ub2 = 10*d_channel; %10*d_channel
```

```
ub = [ub1 ub1 ub1 ub1 ub1 ub2 ub2 ub2 ub2 ub2]; %upper bounder constraint
nonlcon = [];
intcon = [1 2 3 4 5]; %constaining nfin to integers
[x,fval] = gamultiobj(fitnessfcn,nvars,A,b,Aeq,beq,lb,ub,nonlcon,intcon,options);

P1 = subplot(2,1,1);
P2 = subplot(2,1,2);
% P3 = subplot(3,1,3);

%normalzation condition for axis
if norm == 1
    xlabel(P1,'FR_inv_norm')
    ylabel(P1,'Q_inv_norm')
else
    xlabel(P1,'FR_inv (s/kg)') %units for unnormalized results
    ylabel(P1,'Q_inv (1/W)')  %units for unnormalized results
end

title(P1,'')
title(P2,'')
```

```
% This is a function to evaluate the parameters of the genetic algorithm as
% the variables are manipulated. The two objectives of the program consists
% of the inverse of total heat transferred by the fins and the external
% pressure drop of the system

%Every parameter that says in example or arbitrary must be changed for
%application results

function [out] = Case3b_Fin_fitness(params, consts)

format long

% System constants
g = 9.81; %m/(s^2); gravitational acceleration
C_constant = 0.193; %no units; cylinder cross flow constant (arbitrary)
m_constant = 0.618; %no units; cylinder cross flow constant (arbitrary)
%MP_1 and P_1 are irrelevant
MP_1 = 144; %no units; molar mass of fluid (in example)
P_1 = 275000*MP_1; %Pa = N/m^2; Inlet pressure multiplied by MP1 (in example)

% Fluid properties
f = consts(1); % Air side friction factor (from example)
% TO DO:
%1: for specific volume, find way to incorporate iterative solver
% assuming value for pressure at end of channel
%2: incorporate equations to calculate specific volume (assuming same
%specific volume for now)
%3: find better way of approximating v_m
v_1 = consts(2); %m^3/kg; specific volume of inlet (generic air)
v_2 = consts(3); %m^3/kg; specific volume of outlet (generic air)
v_m = (v_2+v_1)/2; %m^3/kg; application specific volume (approximated by log mean absolute
temperature) (in example)
```

```
%A/Ac = L/r_h; --> not valid for our application
Pr = consts(4); %no units; Prandtl number of air (arbitrary)
T_b1 = consts(5); %K; base temperature of channel (arbitrary)
T_b2 = consts(6); %K; base temperature of channel (arbitrary)
T_b3 = consts(7); %K; base temperature of channel (arbitrary)
T_b4 = consts(8); %K; base temperature of channel (arbitrary)
T_b5 = consts(9); %K; base temperature of channel (arbitrary)
T_f = consts(10); %K; fluid temperature (arbitrary)
k_f = consts(11); %W/mK; thermal conductivity of fluid (arbitrary)
k = consts(12); %W/mK; thermal conductivity of fin (arbitrary Al)
mu = consts(13); %Ns/m^2; dynamic visocisty of fluid
t_fin = consts(14) ; %m; thickness of fins (arbitrary)
deltaP = consts(15); %Pa; Pressure drop of system

% Channel geometric properties
L_channel = consts(16); %m; length of channel (arbitrary)
d_channel = consts(17); %m; diameter of channel (arbitrary)

%normalization condition
norm = consts(18);

%Invependent variables
%n_fin_i = input('Number of fins: '); %no units; number of fins on channel (independent variable)
n_fin_i1 = params(1);
n_fin_i2 = params(2);
n_fin_i3 = params(3);
n_fin_i4 = params(4);
n_fin_i5 = params(5);
%if n_fin_i*t_fin*2 > L_channel %accounting for too many fins
%    n_fin_i = input('Input new value for number of fins: ');
%else
%    n_fin_i = n_fin_i;
%end
%L_fin_i = input('Length of fins(m)(multiplied by channel diameter): ')*d_channel; %m; length of
fins (independent variable)
r_fin_i1 = params(6);
r_fin_i2 = params(7);
r_fin_i3 = params(8);
r_fin_i4 = params(9);
r_fin_i5 = params(10);


% % Penalize designs that violate geometric constraints
% tfin_new=0;
% nfin_new=0;
% Convergence_tol=1e-20;
% Error=1;
% tmin=0.001;%(m)
% while Error>Convergence_tol
%     nfin_new=floor((L_channel-(n_fin_i+1)*tmin)/t_fin_i);
%     if nfin_new < 0
%         nfin_new = 0;
%     end
```

```
%     tfin_new=(L_channel-(n_fin_i+1)*tmin)/n_fin_i;
%     if tfin_new < 0.001
%         tfin_new = 0.001;
%     end
%     if nfin_new == 0
%         tfin_new = 0.2;
%     end
%     if nfin_new == 1
%         tfin_new = L_channel;
%     end
%     if nfin_new > 99
%         nfin_new = 99;
%     end
%     if tfin_new > 0.2
%         tfin_new = 0.2;
%     end
%     if nfin_new >= n_fin_i
%         nfin_new = n_fin_i;
%     end
%     if tfin_new >= t_fin_i
%         tfin_new = t_fin_i;
%     end
%     Error=sqrt((tfin_new-t_fin_i)^2+(nfin_new-n_fin_i)^2);
%
%     t_fin_i=tfin_new;
%     n_fin_i=nfin_new;
% end


%Breaking channel into 5 sections
L_channel = L_channel/5;

% Adjust this late for varying CV size
%A_frontal = L_channel*d_max; %m^2; frontal area
%if A_frontal < L_channel*3*d_channel % adjusting for minimum CV area
%    A_frontal = L_channel*3*d_channel;
%else
%    A_frontal = A_frontal;
%end
% sub in 0.21 for d_max CV variables (high case for parameter test)

%Pressure drop for section 1
if n_fin_i1 == 0 || r_fin_i1 == 0 % To correct for having no fins
    d_max1 = d_channel; %m; max diamater of fins on channel
else
    d_max1 = (2*r_fin_i1)+ d_channel; %m; max diamater of fins on channel
end
A_frontal = L_channel*((2*11*d_channel)+d_channel); %m^2; constant frontal area (CV size)
A_free1 = A_frontal-((n_fin_i1*t_fin*d_max1)+(((L_channel-(n_fin_i1*t_fin))*d_channel))); %m^2;
free flow area
sigma_A1 = A_free1/A_frontal; %no units; Free flow area / frontal area of (example)
if r_fin_i1 == 0 % fixing fin length error
    A_s_channel1 = pi*d_channel*(L_channel); %m^2; surface area of channel only (excluding fins)
```

```
else
    A_s_channel1 = pi*d_channel*(L_channel-(n_fin_i1*t_fin)); %m^2; surface area of channel only
(excluding fins)
end
if r_fin_i1 == 0 % fixing fin length error
    A_surf1 = pi*d_channel*(L_channel); %m^2; surface area of channel only (including 0 fins)
else
    A_surf1 = (pi*d_channel*(L_channel-(n_fin_i1*t_fin))) +
(n_fin_i1*t_fin*pi*(d_channel+(2*r_fin_i1))) + (2*n_fin_i1*pi*(((r_fin_i1 + (d_channel))^2)-
((d_channel)^2))); %m^2; surface area of channel and fins (including fins)
end
r_1 = d_channel/2; %m; radius of channel
r_2c1 = (d_max1/2) + (t_fin/2); %m; radius of channel including fins
A_f1 = 2*pi*((r_2c1^2)-(r_1^2)); %m^2; surface area of fins
if d_max1 == d_channel %accounting if max diameter is same as channel diameter
    x1 = d_max1;
else
    x1 = d_max1-d_channel; %m; length for flat plate correlation
end


%Pressure drop equation --> used for flow rate
%dP = deltaP/P1
% dP1 =((G1^2)/(2*g))*(v_1/P_1)*[(1+(sigma_A1^2))*((v_2/v_1)-1)+
(f*(A_surf1/A_free1)*(v_m/v_1))]; %Pa; same as P1
% deltaP1 = abs(dP1 * P_1);
% G1 = FR_i1/A_free1; %kg/(s*m^2); air side mass flux
dP1 = deltaP/P_1;
G1 = sqrt((dP1*(2*g))/((v_1/P_1)*[(1+(sigma_A1^2))*((v_2/v_1)-1)+
(f*(A_surf1/A_free1)*(v_m/v_1))])); %kg/(s*m^2); air side mass flux
FR_i1 = G1*A_free1; %flow rate in section 1

%Reynolds number calculation
Re_D1 = (G1*d_channel)/mu; %no units; Reynolds number of fluid based on the diameter (arbitrary
laminar)
Re_x1 = (G1*x1)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)

% Heat transfer equation for section 1
h_channel1 = (C_constant*(Re_D1^m_constant)*(Pr^(1/3))*k_f)/d_channel; %W/m^2*K; average heat
transfer coefficient of channel in cross flow (excluding fins)
q_conv_channel1 = h_channel1*A_s_channel1*(T_b1-T_f); %W; total heat transferred by channel
(excluding fins)
if r_fin_i1 == 0 %correcting for 0 fin length
    h_fin1 = 0;
else
    h_fin1 = (0.664*(Re_x1^(1/2))*(Pr^(1/3))*k_f)/x1;  %W/m^2*K; average heat transfer
coefficient for one fin in parallel plate flow
end
m1 = sqrt((h_fin1*2)/(t_fin*k)); %no units; constant for fin efficiency
if r_1 == r_2c1 % fixing undefined error
    C_21 = 0;
else
    C_21 = (2*r_1/m1)/((r_2c1^2)-(r_1^2)); %no units; constant for fin efficiency
```

```
end
if C_21 == 0 || r_fin_i1 == 0 || n_fin_i1 == 0 % fixing undefined error
    eta_fin1 = 0;
else
    eta_fin1 = (C_21*((besselk(1,m1*r_1)*besseli(1,m1*r_2c1))-
(besseli(1,m1*r_1)*besselk(1,m1*r_2c1))))/((besseli(0,m1*r_1)*besselk(1,m1*r_2c1))+(besselk(0,m1*
r_1)*besseli(1,m1*r_2c1))); %no units; fin efficiency
end
q_conv_fins1 = n_fin_i1*eta_fin1*A_f1*h_fin1*(T_b1-T_f); %W; total heat transferred by fins
q_conv1 = q_conv_channel1+q_conv_fins1;




%Pressure drop for section 2
if n_fin_i2 == 0 || r_fin_i2 == 0 % To correct for having no fins
    d_max2 = d_channel; %m; max diamater of fins on channel
else
    d_max2 = (2*r_fin_i2)+ d_channel; %m; max diamater of fins on channel
end
A_free2 = A_frontal-((n_fin_i2*t_fin*d_max2)+(((L_channel-(n_fin_i2*t_fin))*d_channel))); %m^2;
free flow area
sigma_A2 = A_free2/A_frontal; %no units; Free flow area / frontal area of (example)
if r_fin_i2 == 0 % fixing fin length error
    A_s_channel2 = pi*d_channel*(L_channel); %m^2; surface area of channel only (excluding fins)
else
    A_s_channel2 = pi*d_channel*(L_channel-(n_fin_i2*t_fin)); %m^2; surface area of channel only
(excluding fins)
end
if r_fin_i2 == 0 % fixing fin length error
    A_surf2 = pi*d_channel*(L_channel); %m^2; surface area of channel only (including 0 fins)
else
    A_surf2 = (pi*d_channel*(L_channel-(n_fin_i2*t_fin))) +
(n_fin_i2*t_fin*pi*(d_channel+(2*r_fin_i2))) + (2*n_fin_i2*pi*(((r_fin_i2 + (d_channel))^2)-
((d_channel)^2))); %m^2; surface area of channel and fins (including fins)
end
r_2c2 = (d_max2/2) + (t_fin/2); %m; radius of channel including fins
A_f2 = 2*pi*((r_2c2^2)-(r_1^2)); %m^2; surface area of fins
if d_max2 == d_channel %accounting if max diameter is same as channel diameter
    x2 = d_max2;
else
    x2 = d_max2-d_channel; %m; length for flat plate correlation
end

%Pressure drop equation --> for flow rate
%dP = deltaP/P1
% dP2 =((G2^2)/(2*g))*(v_1/P_1)*[(1+(sigma_A2^2))*((v_2/v_1)-1)+
(f*(A_surf2/A_free2)*(v_m/v_1))]; %Pa; same as P1
% deltaP2 = abs(dP2 * P_1);
% G2 = FR_i2/A_free2; %kg/(s*m^2); air side mass flux
dP2 = deltaP/P_1;
G2 = sqrt((dP2*(2*g))/((v_1/P_1)*[(1+(sigma_A2^2))*((v_2/v_1)-1)+
(f*(A_surf2/A_free2)*(v_m/v_1))])); %kg/(s*m^2); air side mass flux
```

```
FR_i2 = G2*A_free2; %flow rate in section 2

%Reynolds number calculation
Re_D2 = (G2*d_channel)/mu; %no units; Reynolds number of fluid based on the diameter (arbitrary
laminar)
Re_x2 = (G2*x2)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)

% Heat transfer equation for section 2
h_channel2 = (C_constant*(Re_D2^m_constant)*(Pr^(1/3))*k_f)/d_channel; %W/m^2*K; average heat
transfer coefficient of channel in cross flow (excluding fins)
q_conv_channel2 = h_channel2*A_s_channel2*(T_b2-T_f); %W; total heat transferred by channel
(excluding fins)
if r_fin_i2 == 0 %correcting for 0 fin length
    h_fin2 = 0;
else
    h_fin2 = (0.664*(Re_x2^(1/2))*(Pr^(1/3))*k_f)/x2;  %W/m^2*K; average heat transfer
coefficient for one fin in parallel plate flow
end
m2 = sqrt((h_fin2*2)/(t_fin*k)); %no units; constant for fin efficiency
if r_1 == r_2c2 % fixing undefined error
    C_22 = 0;
else
    C_22 = (2*r_1/m2)/((r_2c2^2)-(r_1^2)); %no units; constant for fin efficiency
end
if C_22 == 0 || r_fin_i2 == 0 || n_fin_i2 == 0 % fixing undefined error
    eta_fin2 = 0;
else
    eta_fin2 = (C_22*((besselk(1,m2*r_1)*besseli(1,m2*r_2c2))-
(besseli(1,m2*r_1)*besselk(1,m2*r_2c2))))/((besseli(0,m2*r_1)*besselk(1,m2*r_2c2))+(besselk(0,m2*
r_1)*besseli(1,m2*r_2c2))); %no units; fin efficiency
end
q_conv_fins2 = n_fin_i2*eta_fin2*A_f2*h_fin2*(T_b2-T_f); %W; total heat transferred by fins
q_conv2 = q_conv_channel2+q_conv_fins2;




%Pressure drop for section 3
if n_fin_i3 == 0 || r_fin_i3 == 0 % To correct for having no fins
    d_max3 = d_channel; %m; max diamater of fins on channel
else
    d_max3 = (2*r_fin_i3)+ d_channel; %m; max diamater of fins on channel
end
A_free3 = A_frontal-((n_fin_i3*t_fin*d_max3)+(((L_channel-(n_fin_i3*t_fin))*d_channel))); %m^2;
free flow area
sigma_A3 = A_free3/A_frontal; %no units; Free flow area / frontal area of (example)
if r_fin_i3 == 0 % fixing fin length error
    A_s_channel3 = pi*d_channel*(L_channel); %m^2; surface area of channel only (excluding fins)
else
    A_s_channel3 = pi*d_channel*(L_channel-(n_fin_i3*t_fin)); %m^2; surface area of channel only
(excluding fins)
end
if r_fin_i3 == 0 % fixing fin length error
```

```
    A_surf3 = pi*d_channel*(L_channel); %m^2; surface area of channel only (including 0 fins)
else
    A_surf3 = (pi*d_channel*(L_channel-(n_fin_i3*t_fin))) +
(n_fin_i3*t_fin*pi*(d_channel+(2*r_fin_i3))) + (2*n_fin_i3*pi*(((r_fin_i3 + (d_channel))^2)-
((d_channel)^2))); %m^2; surface area of channel and fins (including fins)
end
r_2c3 = (d_max3/2) + (t_fin/2); %m; radius of channel including fins
A_f3 = 2*pi*((r_2c3^2)-(r_1^2)); %m^2; surface area of fins
if d_max3 == d_channel %accounting if max diameter is same as channel diameter
    x3 = d_max3;
else
    x3 = d_max3-d_channel; %m; length for flat plate correlation
end

%Pressure drop equation --> for flow rate
%dP = deltaP/P1
% dP3 =((G3^2)/(2*g))*(v_1/P_1)*[(1+(sigma_A3^2))*((v_2/v_1)-1)+
(f*(A_surf3/A_free3)*(v_m/v_1))]; %Pa; same as P1
% deltaP3 = abs(dP3 * P_1);
% G3 = FR_i3/A_free3; %kg/(s*m^2); air side mass flux
dP3 = deltaP/P_1;
G3 = sqrt((dP3*(2*g))/((v_1/P_1)*[(1+(sigma_A3^2))*((v_2/v_1)-1)+
(f*(A_surf3/A_free3)*(v_m/v_1))])); %kg/(s*m^2); air side mass flux
FR_i3 = G3*A_free3; %flow rate in section 3

%Reynolds number calculation
Re_D3 = (G3*d_channel)/mu; %no units; Reynolds number of fluid based on the diameter (arbitrary
laminar)
Re_x3 = (G3*x3)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)

% Heat transfer equation for section 3
h_channel3 = (C_constant*(Re_D3^m_constant)*(Pr^(1/3))*k_f)/d_channel; %W/m^2*K; average heat
transfer coefficient of channel in cross flow (excluding fins)
q_conv_channel3 = h_channel3*A_s_channel3*(T_b3-T_f); %W; total heat transferred by channel
(excluding fins)
if r_fin_i3 == 0 %correcting for 0 fin length
    h_fin3 = 0;
else
    h_fin3 = (0.664*(Re_x3^(1/2))*(Pr^(1/3))*k_f)/x3;  %W/m^2*K; average heat transfer
coefficient for one fin in parallel plate flow
end
m3 = sqrt((h_fin3*2)/(t_fin*k)); %no units; constant for fin efficiency
if r_1 == r_2c3 % fixing undefined error
    C_23 = 0;
else
    C_23 = (2*r_1/m3)/((r_2c3^2)-(r_1^2)); %no units; constant for fin efficiency
end
if C_23 == 0 || r_fin_i3 == 0 || n_fin_i3 == 0 % fixing undefined error
    eta_fin3 = 0;
else
    eta_fin3 = (C_23*((besselk(1,m3*r_1)*besseli(1,m3*r_2c3))-
(besseli(1,m3*r_1)*besselk(1,m3*r_2c3))))/((besseli(0,m3*r_1)*besselk(1,m3*r_2c3))+(besselk(0,m3*
r_1)*besseli(1,m3*r_2c3))); %no units; fin efficiency
```

```
end
q_conv_fins3 = n_fin_i3*eta_fin3*A_f3*h_fin3*(T_b3-T_f); %W; total heat transferred by fins
q_conv3 = q_conv_channel3+q_conv_fins3;




%Pressure drop for section 4
if n_fin_i4 == 0 || r_fin_i4 == 0 % To correct for having no fins
    d_max4 = d_channel; %m; max diamater of fins on channel
else
    d_max4 = (2*r_fin_i4)+ d_channel; %m; max diamater of fins on channel
end
A_free4 = A_frontal-((n_fin_i4*t_fin*d_max4)+(((L_channel-(n_fin_i4*t_fin))*d_channel))); %m^2;
free flow area
sigma_A4 = A_free4/A_frontal; %no units; Free flow area / frontal area of (example)
if r_fin_i4 == 0 % fixing fin length error
    A_s_channel4 = pi*d_channel*(L_channel); %m^2; surface area of channel only (excluding fins)
else
    A_s_channel4 = pi*d_channel*(L_channel-(n_fin_i4*t_fin)); %m^2; surface area of channel only
(excluding fins)
end
if r_fin_i4 == 0 % fixing fin length error
    A_surf4 = pi*d_channel*(L_channel); %m^2; surface area of channel only (including 0 fins)
else
    A_surf4 = (pi*d_channel*(L_channel-(n_fin_i4*t_fin))) +
(n_fin_i4*t_fin*pi*(d_channel+(2*r_fin_i4))) + (2*n_fin_i4*pi*(((r_fin_i4 + (d_channel))^2)-
((d_channel)^2))); %m^2; surface area of channel and fins (including fins)
end
r_2c4 = (d_max4/2) + (t_fin/2); %m; radius of channel including fins
A_f4 = 2*pi*((r_2c4^2)-(r_1^2)); %m^2; surface area of fins
if d_max4 == d_channel %accounting if max diameter is same as channel diameter
    x4 = d_max4;
else
    x4 = d_max4-d_channel; %m; length for flat plate correlation
end

%Pressure drop equation --> for flow rate
%dP = deltaP/P1
% dP4 =((G4^2)/(2*g))*(v_1/P_1)*[(1+(sigma_A4^2))*((v_2/v_1)-1)+
(f*(A_surf4/A_free4)*(v_m/v_1))]; %Pa; same as P1
% deltaP4 = abs(dP4 * P_1);
% G4 = FR_i4/A_free4; %kg/(s*m^2); air side mass flux
dP4 = deltaP/P_1;
G4 = sqrt((dP4*(2*g))/((v_1/P_1)*[(1+(sigma_A4^2))*((v_2/v_1)-1)+
(f*(A_surf4/A_free4)*(v_m/v_1))])); %kg/(s*m^2); air side mass flux
FR_i4 = G4*A_free4; %flow rate in section 4

%Reynolds number calculation
Re_D4 = (G4*d_channel)/mu; %no units; Reynolds number of fluid based on the diameter (arbitrary
laminar)
Re_x4 = (G4*x4)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)
```

```
% Heat transfer equation for section 4
h_channel4 = (C_constant*(Re_D4^m_constant)*(Pr^(1/3))*k_f)/d_channel; %W/m^2*K; average heat
transfer coefficient of channel in cross flow (excluding fins)
q_conv_channel4 = h_channel4*A_s_channel4*(T_b4-T_f); %W; total heat transferred by channel
(excluding fins)
if r_fin_i4 == 0 %correcting for 0 fin length
    h_fin4 = 0;
else
    h_fin4 = (0.664*(Re_x4^(1/2))*(Pr^(1/3))*k_f)/x4;  %W/m^2*K; average heat transfer
coefficient for one fin in parallel plate flow
end
m4 = sqrt((h_fin4*2)/(t_fin*k)); %no units; constant for fin efficiency
if r_1 == r_2c4 % fixing undefined error
    C_24 = 0;
else
    C_24 = (2*r_1/m4)/((r_2c4^2)-(r_1^2)); %no units; constant for fin efficiency
end
if C_24 == 0 || r_fin_i4 == 0 || n_fin_i4 == 0 % fixing undefined error
    eta_fin4 = 0;
else
    eta_fin4 = (C_24*((besselk(1,m4*r_1)*besseli(1,m4*r_2c4))-
(besseli(1,m4*r_1)*besselk(1,m4*r_2c4))))/((besseli(0,m4*r_1)*besselk(1,m4*r_2c4))+(besselk(0,m4*
r_1)*besseli(1,m4*r_2c4))); %no units; fin efficiency
end
q_conv_fins4 = n_fin_i4*eta_fin4*A_f4*h_fin4*(T_b4-T_f); %W; total heat transferred by fins
q_conv4 = q_conv_channel4+q_conv_fins4;




%Pressure drop for section 5
if n_fin_i5 == 0 || r_fin_i5 == 0 % To correct for having no fins
    d_max5 = d_channel; %m; max diamater of fins on channel
else
    d_max5 = (2*r_fin_i5)+ d_channel; %m; max diamater of fins on channel
end
A_free5 = A_frontal-((n_fin_i5*t_fin*d_max5)+(((L_channel-(n_fin_i5*t_fin))*d_channel))); %m^2;
free flow area
sigma_A5 = A_free5/A_frontal; %no units; Free flow area / frontal area of (example)
if r_fin_i5 == 0 % fixing fin length error
    A_s_channel5 = pi*d_channel*(L_channel); %m^2; surface area of channel only (excluding fins)
else
    A_s_channel5 = pi*d_channel*(L_channel-(n_fin_i5*t_fin)); %m^2; surface area of channel only
(excluding fins)
end
if r_fin_i5 == 0 % fixing fin length error
    A_surf5 = pi*d_channel*(L_channel); %m^2; surface area of channel only (including 0 fins)
else
    A_surf5 = (pi*d_channel*(L_channel-(n_fin_i5*t_fin))) +
(n_fin_i5*t_fin*pi*(d_channel+(2*r_fin_i5))) + (2*n_fin_i5*pi*(((r_fin_i5 + (d_channel))^2)-
((d_channel)^2))); %m^2; surface area of channel and fins (including fins)
```

```
end
r_2c5 = (d_max5/2) + (t_fin/2); %m; radius of channel including fins
A_f5 = 2*pi*((r_2c5^2)-(r_1^2)); %m^2; surface area of fins
if d_max5 == d_channel %accounting if max diameter is same as channel diameter
    x5 = d_max5;
else
    x5 = d_max5-d_channel; %m; length for flat plate correlation
end

%Pressure drop equation --> for flow rate
%dP = deltaP/P1
% dP5 =((G5^2)/(2*g))*(v_1/P_1)*[(1+(sigma_A5^2))*((v_2/v_1)-1)+
(f*(A_surf5/A_free5)*(v_m/v_1))]; %Pa; same as P1
% deltaP5 = abs(dP5 * P_1);
% G5 = FR_i5/A_free5; %kg/(s*m^2); air side mass flux
dP5 = deltaP/P_1;
G5 = sqrt((dP5*(2*g))/((v_1/P_1)*[(1+(sigma_A5^2))*((v_2/v_1)-1)+
(f*(A_surf5/A_free5)*(v_m/v_1))])); %kg/(s*m^2); air side mass flux
FR_i5 = G5*A_free5; %flow rate in section 5

%Reynolds number calculation
Re_D5 = (G5*d_channel)/mu; %no units; Reynolds number of fluid based on the diameter (arbitrary
laminar)
Re_x5 = (G5*x5)/mu; %no units; Reynolds number of fluid based on x (arbitrary laminar)

% Heat transfer equation for section 5
h_channel5 = (C_constant*(Re_D5^m_constant)*(Pr^(1/3))*k_f)/d_channel; %W/m^2*K; average heat
transfer coefficient of channel in cross flow (excluding fins)
q_conv_channel5 = h_channel5*A_s_channel5*(T_b5-T_f); %W; total heat transferred by channel
(excluding fins)
if r_fin_i5 == 0 %correcting for 0 fin length
    h_fin5 = 0;
else
    h_fin5 = (0.664*(Re_x5^(1/2))*(Pr^(1/3))*k_f)/x5;  %W/m^2*K; average heat transfer
coefficient for one fin in parallel plate flow
end
m5 = sqrt((h_fin5*2)/(t_fin*k)); %no units; constant for fin efficiency
if r_1 == r_2c5 % fixing undefined error
    C_25 = 0;
else
    C_25 = (2*r_1/m5)/((r_2c5^2)-(r_1^2)); %no units; constant for fin efficiency
end
if C_25 == 0 || r_fin_i5 == 0 || n_fin_i5 == 0 % fixing undefined error
    eta_fin5 = 0;
else
    eta_fin5 = (C_25*((besselk(1,m5*r_1)*besseli(1,m5*r_2c5))-
(besseli(1,m5*r_1)*besselk(1,m5*r_2c5))))/((besseli(0,m5*r_1)*besselk(1,m5*r_2c5))+(besselk(0,m5*
r_1)*besseli(1,m5*r_2c5))); %no units; fin efficiency
end
q_conv_fins5 = n_fin_i5*eta_fin5*A_f5*h_fin5*(T_b5-T_f); %W; total heat transferred by fins
q_conv5 = q_conv_channel5+q_conv_fins5;
```

```
%summing total heat transfer
q_conv = q_conv1 + q_conv2 + q_conv3 + q_conv4 + q_conv5;
q_inv = 1/q_conv;

%summing flow rate
FR_system = FR_i1 + FR_i2 + FR_i3 + FR_i4 + FR_i5; % units: kg/s; flow rate of system


fval = [FR_system q_inv]; %outputs the pressure drop and inverse heat transferred (unnormalized
results)

%UPDATE THIS WHEN CHANGING dP
FR_min = ; % FR from no fins
FR_max = ; % FR from max fins and max fin size
Q_inv_min = ; %1/W from max fins and max fin size
Q_inv_max = ; %1/W from no fins
FRnorm = (FR_system - FR_min)/ (FR_max - FR_min); %normalize dP
q_invnorm = (q_inv - Q_inv_min)/(Q_inv_max-Q_inv_min); %normalize Q_inv
outnorm = [FRnorm q_invnorm]; %outputs the pressure drop and inverse heat transferred normalized
results
%normalization condition

if norm == 1
    [out] = [outnorm];
else
    [out] = [fval];
end









%{
the following section is calculations for L_flow and r_h (only relavant for
rectangular flow channels)
if L_fin_i == 0 % Correcting for no fin length
    %P_wet = 2*(L_channel)+(2*(d_max-d_channel)); % m; wetting perimeter of the fluid flow cross
section
    P_wet = 2*(L_channel)+(2*(0.21-d_channel)); % m; wetting perimeter of the fluid flow cross
section
else
    %P_wet = (2*(L_channel-(n_fin_i*t_fin))+(L_fin_i*n_fin_i*4))+(2*(d_max-d_channel)); % m;
wetting perimeter of the fluid flow cross section
    P_wet = (2*(L_channel-(n_fin_i*t_fin))+(L_fin_i*n_fin_i*4))+(2*(0.21-d_channel)); % m;
wetting perimeter of the fluid flow cross section
end
r_h = A_free/P_wet; %m; flow passage hydraulic radius / diameter
%L_flow = d_max; %m; flow depth
```

```
L_flow = 0.21; %m; flow depth
%}
```

```
%Fin figures driver script

clear all
clc
close all

A = Fin_Figures(); % creating instance of fin figure class
%A = Figures(); % creating instance of fin figure class
r1 = 0.1; %mm, fin radius
r = @(z) 0.01; % mm, channel diameter
L = 0.2; %m
deltaPnorm = 1; %INPUT FROM PARETO FOR NORMALIZED RESULTS
q_invnorm = 3.17933810049489E-06; %INPUT FROM PARETO FOR NORMALIZED RESULTS

% Change below depending on case
P_min = 0.003801003955636; % Pa from no fins
P_max = 6.286860542621986; % Pa from max fins and max fin size
Q_inv_min = 3.206305563582875e-04; %1/W from max fins and max fin size
Q_inv_max =  0.021734500426815; %1/W from no fins


deltaP = (deltaPnorm*(P_max - P_min)) + P_min; %unnormalize dP
q_inv = (q_invnorm*(Q_inv_max-Q_inv_min)) + Q_inv_min; %unnormalize Q_inv
%deltaP = 0.215272721; %unnormalize dP
%q_inv = 0.000838972; %unnormalize Q_inv
Q = 1/q_inv; % W
dP = deltaP; % Pa
c = @(z) 0;
n_fin = 33; %number of fins
r_fin = 1*r1;
t = 0.005; %mm
A.plot3Dchannel(r,c,L,n_fin,r_fin,dP,Q,t); % run class function (create a plot) using the dot
operator
set(gcf, 'Position',  [250, 250, 750, 750])
%A.plot3Dchannel(r,c,L,dP,h); % run class function (create a plot) using the dot operator
```

```
% Class for easily creating plots with a standardized format
% See MATLAB documentation for differences between handle and value classes
classdef Fin_Figures < handle

    % Feel free to change these properties as you see fit
    properties (Constant)
        colors = ['b','r','k','g']
        FontSize = 12
        lineWidth = 3
        xlab = 'x'
        ylab = 'y'

        paretoFormat = struct(...
            'xlabel','\DeltaP',...
```

```
            'ylabel','1/h')

        rad2DFormat = struct(...
            'xlabel','z (cm)',...
            'ylabel', 'r (cm)',...
            'LineWidth', 2)

        rad3DFormat = struct(...
            'xlabel','{\boldmath$z$} \bf{(mm)}',...
            'ylabel', '{\boldmath$r$} \bf{(mm)}',...
            'zlabel', '{\boldmath$r$} \bf{(mm)}',...
            'LineWidth', 3,...
            'opacity', 0.5)

        incrementalFormat = struct(...
            'xlabel', 'z (cm)',...
            'ylabelP', '\DeltaP (Pa)',... % Check units
            'ylabelh', 'h (W/m^2*K)',...
            'ylabelx', 'Flow Quality',...
            'FontSize',12,...
            'LineWidth',2)

    end

    methods
        % All properties are constant, so the constructor doesnt need to
        % set any data members
        function obj = Figures()
        end


        % ---------------------------------------------------------------
        % Function for easily producing Pareto plots
        % ---------------------------------------------------------------
        % norm_p, norm_h are either matrices or vectors containing all of
        % the normalized pressure drops and inverse heat transfer
        % coefficients for all of the evaluated designs
        % opts is an optional cell array containing formatting data. To see
        % what data must be included in this array, look at the constant
        % data member "paretoFormat"
        function [] = pareto(obj, norm_p, norm_h, opts)
            if nargin < 4 % Change this value if number of args changes
                opts = obj.paretoFormat;
            end

            if ~isvector(norm_p) && ~isvector(norm_h)
                norm_p = reshape(norm_p, 1, []);
                norm_h = reshape(norm_h, 1, []);
                fprintf('this function works\n')
            end

            if length(norm_p) == length(norm_h)
```

```
            figure;
            scatter(norm_p,norm_h);
            xlabel(opts.xlabel);
            ylabel(opts.ylabel);
            set(gca,'FontSize',obj.FontSize);
        else
            warning('Pareto plot vectors/matrices not the same size');
            return;
        end
    end


    % ---------------------------------------------------------------
    % Function for creating channel surface plots
    % ---------------------------------------------------------------
    % r = radius function or vector, c = centerline function or vector
    % dP = total pressure drop, hInv = inverse heat transfer coef.
    % Call "figure;" before running this function
    % opts = figure formatting options. If not specified, the function
    % will use the defaults listed at the top of this class file
    function [p] = plot3Dchannel(obj,r,c,L,n_fin,r_fin,dP,Q,t,opts)
        if nargin < 10 % Change this value if number of args changes
            opts = obj.rad3DFormat;
        end

        N = 20; % number of points to plot
        z = linspace(0,L,N);
        if isa(r,'function_handle')
            rp = zeros(N,1); % rp is the discrete radius function
            for i = 1:N
                rp(i) = r(z(i));
            end
        else
            rp = r;
        end

        if isa(c,'function_handle')
            cp = zeros(N,1); % cp is the discrete centerline function
            for i = 1:N
                cp(i) = c(z(i));
            end
        else
            cp = c; % Apply this
        end

        % 40 is number of points in each "circle"
        %constants
        t = t*10; % mm; fin thickness
        h = 1000*L; % mm; height
        r_fin = rp + r_fin; %radius of channel and fin

        [r_1channel,r_2channel,z_ind] = cylinder(1000*rp,20); %set size for diameter
```

```
            [r_1fin,r_2fin,z_ind] = cylinder(1000*r_fin,20); %set size for fin diameter
            i = 0; %indexing variable for surface plot
            z = zeros(2,21); %filling initial matrix with zeros
            r1 = zeros(2,21); %filling initial matrix with zeros
            r2= zeros(2,21); %filling initial matrix with zeros
            while i ~= n_fin %Loop to add sections to the channel for each number of fins
                b = [z(end,:)];
                z = [z; ((h-(100*t*n_fin))/((n_fin+1)*h)).*z_ind + b];
                b = [z(end,:)];
                z = [z; t.*z_ind/2 + b]; % dividing by 2 is to adjust for doublinmg channel
length
                r1 = [r1; r_1channel; r_1fin];
                r2 = [r2; r_2channel; r_2fin];
                i = i+1;
            end
            b = [z(end,:)];
            z = [z; ((h-(100*t*n_fin))/((n_fin+1)*h)).*z_ind + b]; %adding the end channel
            r1 = [r1; r_1channel];
            r2 = [r2; r_2channel];
            %removing intial zeros from matrix
            z(1,:) = [];
            z(1,:) = [];
            r1(1,:) = [];
            r1(1,:) = [];
            r2(1,:) = [];
            r2(1,:) = [];
            z_cen = h*z_ind;
            z = z.*h;

            figure;
            %%figureHandle = plot3(z_cen,cp,cp,'r-','LineWidth',opts.LineWidth); % This might
need to be changed later on
            figureHandle = surf(z,r1,r2);
            hold on;
            p = surf(z,r1,r2); % Assigning the surface plot to a variable
            rz = sqrt(r1.^2 + r2.^2);

            % Saving rz to the base workspace. This is needed in order to
            % carry out the next command, which searches the workspace for
            % the string 'rz'.
            % The 'set' and 'refreshdata' commands change the direction of
            % the surface plot in which the colormap varies. Now, color
            % only changes with rz, the distance from the channel
            % centerline (the radius). To see what this means, comment out
            % the next two lines and run this function again.
            assignin('base','rz',rz); set(p, 'CDataSource', 'rz'); % Assign colorbar as a
function of radius
            refreshdata;
            % Scaling the output space to show what the channel would
            % actually look like
            axis equal;
           % set(gca,'DataAspectRatio',[1 1/300 1/300])
```

```
            xlabel(opts.xlabel,'interpreter','latex','FontSize',16);
            ylabel(opts.ylabel,'interpreter','latex','FontSize',16);
            zlabel(opts.zlabel,'interpreter','latex','FontSize',16);
            %Adjust axis limits
            ylim([-110 110]);
            zlim([-110 110]);

            % set(gca,'FontSize',obj.FontSize);
            % hold on;
            alpha (opts.opacity); % Changes opacity of surface plot
%
%             % Commented out for poster presentation of results
%             % legend('Channel Centerline', 'Location',
'Northwest','interpreter','latex','FontSize',16);
%             hcb = colorbar;
%             caxis([1000*0.008/2 1000*0.0350/2]); % SET COLORBAR HERE
%             colorTitleHandle = get(hcb,'Title');
%             titleString = 'Channel Radius (mm)';
%             set(colorTitleHandle ,'String',titleString,'interpreter','latex','FontSize',14);

            % Location needs tweaked
            dim1 = [0.10 0.55 0.3 0.3]; % normalized location of textbox
            str1 = {
                sprintf('\\bf{{\\boldmath$\\Delta p = $} %3.2f Pa}',dP),... % Figure out delta
                sprintf('\\bf{{\\boldmath$Q = $} %3.1f W}',Q)
            };
            t =
annotation('textbox',dim1,'String',str1,'FontSize',18,'FitBoxToText','on','interpreter','latex');

            t.FontWeight = 'bold';
            t.BackgroundColor = (1/255)*[205,222,238]';

            %change color or surface plot
            %set(figureHandle,'FaceColor',[0 0 1],'FaceAlpha', 1);
        end


    end
end
```

Published with MATLAB® R2021b

BIBLIOGRAPHY

[1] W. Howarth, “Straightening Out Fan Curves,” *AMCA immotion magazine*, 2020. https://www.amca.org/educate/articles-and-technical-papers/amca-inmotion-articles/straightening-out-fan-curves.html (accessed May 01, 2021).

[2] T. L. Bergman and A. S. Lavine, *Fundamentals of Heat and Mass Transfer*, 7th ed., vol. 1, no. 1. John Wiley & Sons, 2018.

[3] N. A. Evich, “Optimal Design of Two-Phase Flow Channels for Additively Manufacturing Using Multi-Objective Genetic Algorithms,” Pennsylvania State University.

[4] L. H. Tang, M. Zeng, and Q. W. Wang, “Experimental and numerical investigation on air-side performance of fin-and-tube heat exchangers with various fin patterns,” *Exp. Therm. Fluid Sci.*, vol. 33, no. 5, pp. 818–827, Jul. 2009, doi: 10.1016/j.expthermflusci.2009.02.008.

[5] S. Unger, M. Beyer, S. Gruber, R. Willner, and U. Hampel, “Experimental study on the air-side thermal-flow performance of additively manufactured heat exchangers with novel fin designs,” *Int. J. Therm. Sci.*, vol. 146, p. 106074, Dec. 2019, doi: 10.1016/j.ijthermalsci.2019.106074.

[6] W. . Kays and A. L. London, *Compact Heat Exchangers*, 3rd ed. McGraw-Hill, 1984.

[7] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, vol. 53, no. 9. 2017.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

ACADEMIC VITA

Anthony Luu

www.linkedin.com/in/anthonyluu1589

EDUCATION

The Pennsylvania State University
Bachelor of Science in Mechanical Engineering — Anticipated: May 2022
Engineering Leadership Development Minor
Scholar of the Schreyer Honors College

PROFESSIONAL EXPERIENCE

Penn State University | Computational Tools Teaching Assistant | State College, PA — Aug 2021 – Present
- Assisted students' conduct finite element analysis studies for mechanical, thermal, and fluid properties
- Troubleshooted students' CAD models in SolidWorks in preparation for finite element analysis

Phillips 66 | Lubricants Intern | Houston, TX — May 2021 – Aug 2021
- Conducted an economic analysis to determine feasibility for a new packaging type while considering the business unit's production supply chain
- Collaborated with external suppliers and company subject experts to compile a database of information
- Communicated key findings to leadership group, highlighting key findings and recommendations

Guidedwave Ultrasonics Inc. | Mechanical Engineering Intern | Bellefonte, PA — Mar 2020 – Nov 2020
- Designed CAD models for varying applications, gaining proficiency in Creo Parametric and design fundamentals
- Manufactured parts and products through different processes to develop operating skills in the design lab
- Contributed to design processes by building CAD models and adjusting designs for ongoing projects

Penn State University |Additive Manufacturing Research Assistant | State College, PA — Jan 2020 – Apr 2020
- Utilized MATLAB to analyze the effectiveness of a network approach for spatial recurrence analysis
- Applied spatial analysis to medical CAT scans to identify porosities and cracking throughout their layers

PROJECTS

Undergraduate Thesis
- Employed genetic optimization to investigate optimal fin topology for two-phase flow heat exchangers
- Compiled research by highlighting relevant background information, methodologies, and results of the study

Mechanical Engineering Design Methodology
- Developed an autonomous drawing system for an Etch-a-Sketch with an analytical approach in MATLAB
- Designed a guided user interface to facilitate user interactivity and customization with the program
- Led a team in organizing a structured report to highlight the constraints and potential of the system's design

TECHNICAL SKILLS

- **Advanced:** SolidWorks, MATLAB
- **Intermediate:** Arduino IDE, Creo Parametric, Alteryx, Tableau, Microsoft Excel
- **Proficient:** myVGL, Python

Leadership & Involvement

Asian Undergraduate Student Association | President, Vice President, Secretary — Aug 2018 – Present
- Created a community for cultural expression and inclusion for Asian Americans on campus
- Coordinated with a team of 8 other officers to plan 3-4 weekly events for an average of 50-70 participants
- Contributed to the recruiting and retention of over new 50 members each year
- Raised over $2,500 to self-fund club activities with a limited initial budget