SIMULATION OF TAPERED PIN FINS AND CYLINDERS IN CROSSFLOW

by

Courtney Marie Leeds

A thesis submitted in partial fulfillment of

the requirements for the degree of:

Master of Science

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2018

Approved by:

Professor Gregory F. Nellis	Date

ABSTRACT

This document describes the definition, simulation, performance, and parametric investigation of axially-tapered pin-fin arrays and the process for conducting large parametric analyses using High-Throughput Computing (HTC).

Pin-fin arrays were defined by the layout, degree of taper, and aspect ratio of their pin-fins. Various pin-fin array geometries were modeled and meshed using ANSYS DesignModeler and Meshing. For each geometry, various flow conditions were simulated by using ANSYS Fluent to run Computational Fluid Dynamics (CFD) simulations. A total of 8,250 simulations were ran, which included each combination of 33 pin-fin layouts, 5 degrees of taper, 5 aspect ratios, and 10 flow conditions.

From simulation data, hydrodynamic and thermal performance parameters were calculated. These performance parameters were correlated with geometric parameters and flow conditions using Engineering Equation Solver (EES). The correlations were validated using experimental data from 3D-printed heat exchangers.

The large parametric analysis of axially-tapered pin-fin arrays was enabled by High-Throughput Computing (HTC). The HTC process included preparation, job testing, job submission, and data extraction and it was executed using CHTCondor, Python, and shell scripts. HTC was enabled by the University of Wisconsin-Madison's Center for High-Throughput Computing (CHTC).

ACKNOWLEDGEMENTS

The work in this document is possible due to the guidance, advice, and support and of the following individuals.

Greg, thank you for your guidance and patience the past couple of years. You've not only helped me grow my technical skill set, but you've also helped me grow as person. I would also like to thank Sandy, John, and Franklin for their technical guidance and feedback.

Jake, Evan, Chen, Uzo, and Cass, thank you for troubleshooting codes, mesh sensitivity studies, and other technical problems with me. I've enjoyed learning from each of you and sharing laughs along the way. I would also like to thank Rachel for providing guidance in the early stages of this project.

Amy, Ana, and Diego, thank you for all your personal and professional advice. Your wisdom has been invaluable to me.

Finally, I would like to thank the graduate students in the Solar Energy Laboratory (SEL) for the great company and community you've provided the past few years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Conventionally, pin-fin heat exchangers contain cylinder pin-fins sandwiched between parallel plates. As fluid passes through these channels, heat is transferred between the fluid and the walls of the plates and the cylinders. Advances in additive manufacturing allow detailed surface geometries to be implemented in many technologies, including pin-fin heat exchangers. These surface geometries can be optimized for increased heat exchanger performance.

One option for increasing the performance of pin-fin heat exchangers is to implement axially-tapered pin-fins instead of cylinders. Axially-tapered pin-fins require less material, provide a higher heat transfer coefficient (due to the small, on average, diameter), and provide less pressure drop (due to a larger open area for air flow). To investigate their performance, this work conducted a parametric investigation of axially-tapered pin-fin arrays. Hundreds of different axially-tapered pin-fin array geometries were simulated, and correlations were developed to characterize their hydrodynamic and thermal performance. To facilitate the parametric investigation of axially-tapered pin-fin arrays, High-Throughput Computing (HTC) was utilized.

Sections 2, 3, and 4 of this document discuss the definition, simulation, and performance of axially-tapered pin-fin arrays. Then Section 5 describes the HTC process used by this work to conduct the parametric investigation.

2. DEFINTION OF AXIALLY-TAPERED AND CYLINDRICAL PIN-FIN ARRAYS

2.1. Geometry

The geometry of a bank of an axially-tapered pin-fin array is defined by the transverse and longitudinal pitches (S_T and S_L) with the pin shape itself defined by base and minimum diameters (D and D_{min}) and height (h). These dimensions are labeled for the bank of tapered pins in Figure 1 with fluid entering the first row of pins from the left and exiting to the right. In this work, only banks with uniform pin shapes are investigated.

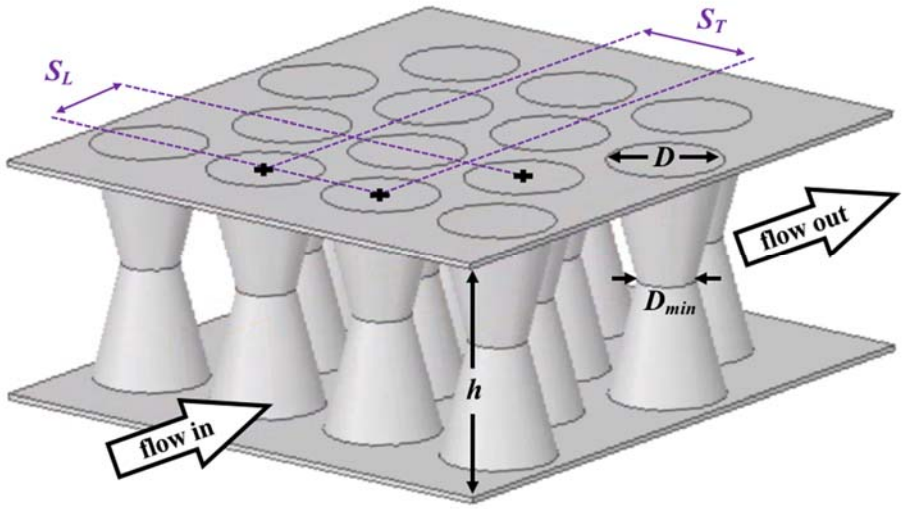


Figure 1: A bank of tapered pins in a staggered arrangement. Fluid enters the first row of pins from the left and exits to the right.

The geometric layout, or spatial arrangement, of pins can also be seen in Figure 2, which is a top view of the bank shown in Figure 1. S_T is the center-to-center distance between two neighboring pins in any given row and S_L is the longitudinal center-to-center distance. The pins

have a staggered geometric layout, meaning that the even rows (e.g., the second and fourth, etc.) are offset from the odd rows (the first and third, etc.) by a distance of $S_T/2$. The pins in all odd rows are in-line with the first-row pins and the pins in all even rows are in-line with the second-row pins.

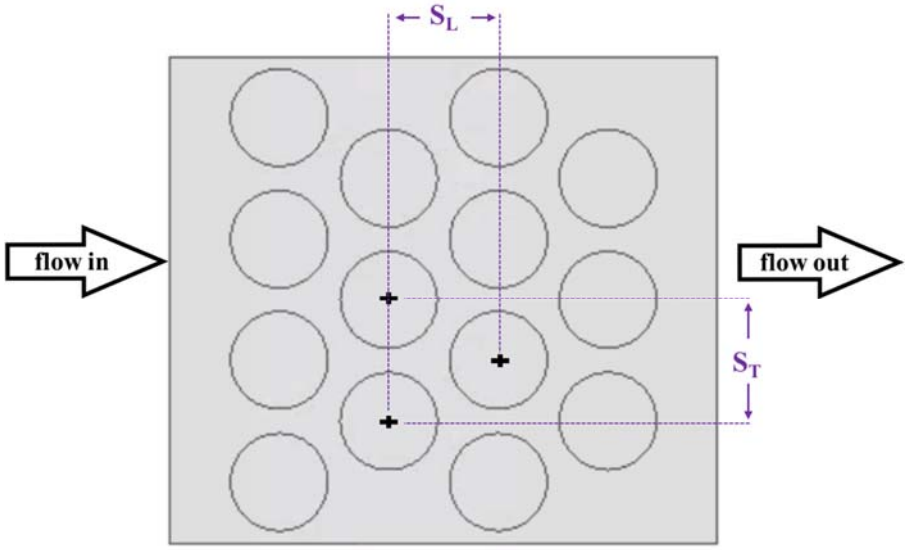


Figure 2: Geometric layout of a bank of pins.

The dimensional pitches (i.e., S_T and S_L) are normalized by the base diameter *D* to define the dimensionless transverse pitch $\overline{S_T}$ and dimensionless longitudinal pitch $\overline{S_L}$:

$$\overline{S_T} = \frac{S_T}{D} \tag{1}$$

$$\overline{S_L} = \frac{S_L}{D} \tag{2}$$

This work investigates values of dimensionless transverse pitch $\overline{S_T}$ that lie between 1.25 and 2.5 and values of dimensionless longitudinal pitch $\overline{S_L}$ that lie between 0.625 and 2.0. If either dimensionless pitch falls below those ranges (e.g., $\overline{S_T} < 1.25$ or $\overline{S_L} < 0.625$), then the pressure drop across the bank was found to be too large to be practically useful. If either dimensionless pitch exceeds those ranges ($\overline{S_T} > 2.5$ or $\overline{S_L} > 2.0$), then the flow across the bank approaches the behavior of flow past a single pin. The optimal combination of dimensionless pitches was found to lie within the simulated range for the heat exchangers considered in this work.

The shape of a fin is defined by two dimensionless parameters: the degree of taper *T* and the dimensionless height *H*. These parameters are defined as follow:

$$T = 1 - \frac{D_{\min}}{D} \tag{3}$$

$$H = \frac{h}{D} \tag{4}$$

Pins with various degrees of taper are shown in Figure 3. When $T = 0$, the pin has a straight-cylinder (i.e., tube) shape. As *T* increases, the minimum diameter of the pin decreases and the pin takes on an hourglass shape. Finally, when $T = 1$, D_{min} approaches 0 and the pin has a maximum taper. Degrees of taper *T* spanning from 0 to 1 are investigated in this work.

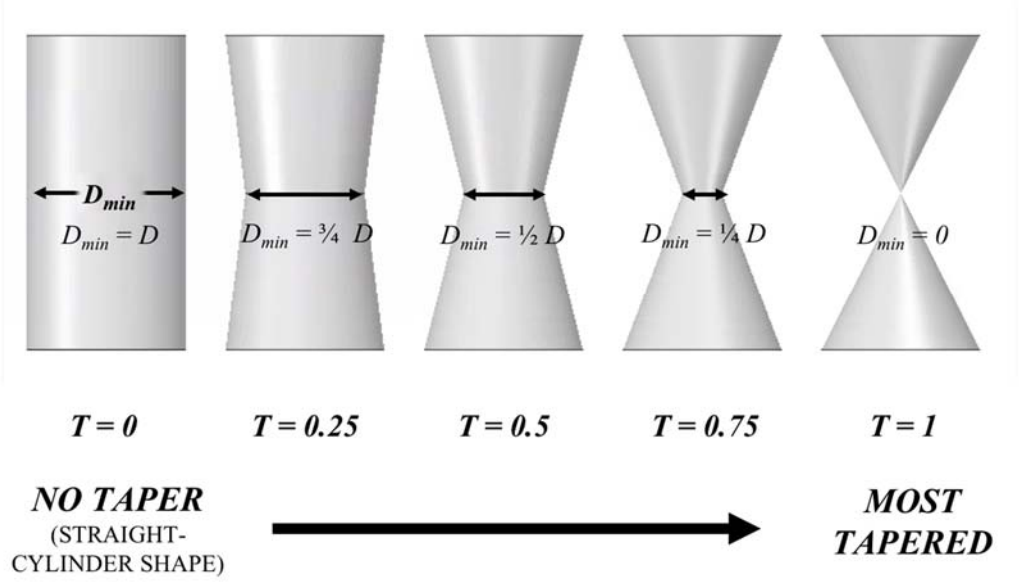


Figure 3: Pins with various degrees of taper

Pins with various dimensionless heights are shown in Figure 4. All three pins being shown have a degree of taper of $T = 0.5$. This work investigates dimensionless heights H between 0.5 and 6.

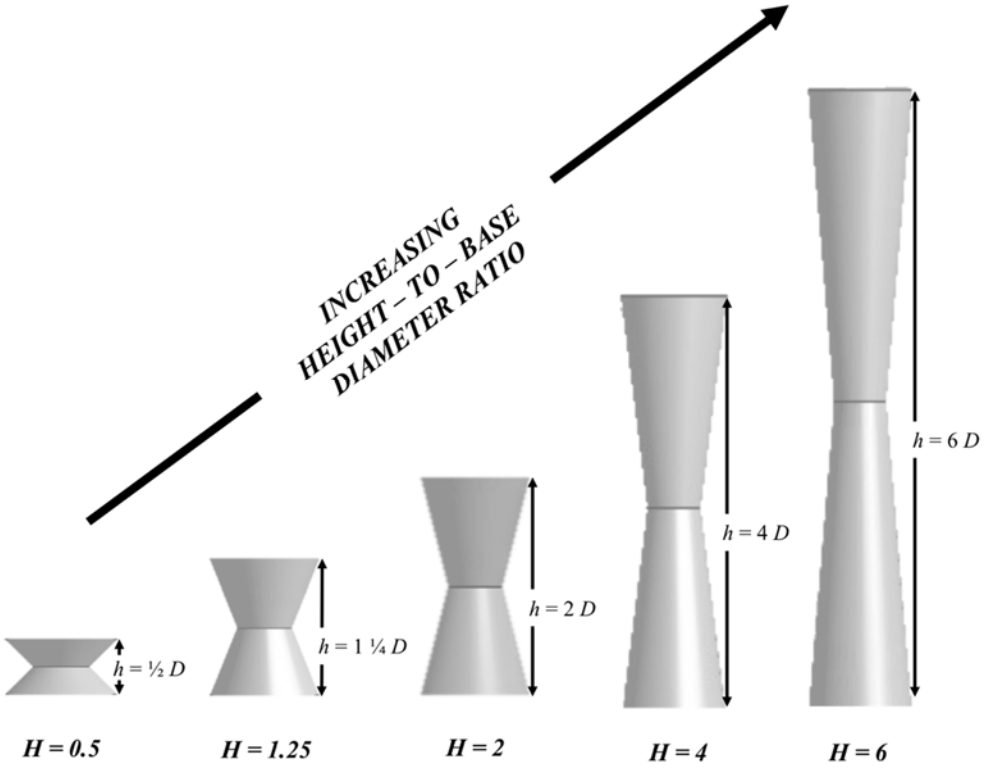


Figure 4: Pins with T = 0.5 and various dimensionless heights

2.2. Flow Conditions

When running simulations, the inlet velocity u_{inf} was varied to simulate a range of Reynolds numbers (Re) between 30 and 1000. This range has been found to be near optimal for the heat exchangers considered in this work. In this work, Re was defined by:

$$\text{Re} = \frac{u_{\text{inf}} D}{\nu} \tag{3}$$

where D is the base diameter of the pin and ν is the kinematic viscosity of the fluid ($\nu = \mu/\rho$)

2.3. Cylinder Correlations from Previous Works

2.3.1 Cylinder Flow Conditions

In previous works defining correlations for cylinders (i.e., $T = 0$), $\mathrm{Re}_{V_{\max}}$ was defined by:

$$\mathrm{Re}_{V_{\max}} = \frac{V_{\max} D}{\nu} \tag{4}$$

where D is the diameter and $V_{\max}$ is the maximum bulk velocity of the fluid.[1]

---

[1] When defining Re for banks of pins, it is important to understand why $V_{\max}$ was chosen by previous works and used in this work as the reference velocity. In reality, heat transfer and hydraulic drag are determined by the average velocity of the flow, integrated over the perimeter of the tube. In banks with wider spaces (which Žukauskas defines as when $\overline{S_T} \geq 1.25$), the average velocity is almost identical to $V_{\max}$. Therefore, the average velocity is justifiably approximated as $V_{\max}$, which is much simpler to calculate. However, in banks with close spaces (when $\overline{S_T} < 1.25$), Žukauskas advises using the average value of velocity as the reference velocity.

When developing correlations in this work, the reference velocity did not provide a significant improvement in velocity; therefore, to simplify the correlations, the inlet velocity was used as the reference velocity.

The physical meaning of V_{max} can be understood with a simple mass balance. For the fluid's mass to be conserved across the bank, the flow rate must increase as the cross-sectional area of the channel decreases. The location with the smallest cross-sectional area A_{min} must then be where the maximum velocity V_{max} occurs. Depending on the bank's geometry (specifically the values of D, S_T, and S_L), A_{min} occurs in one of two possible locations: the gap between neighboring pins in the same row or the gap between a pin and its closest neighbor in the row behind it. Both locations are identified in Figure 5.

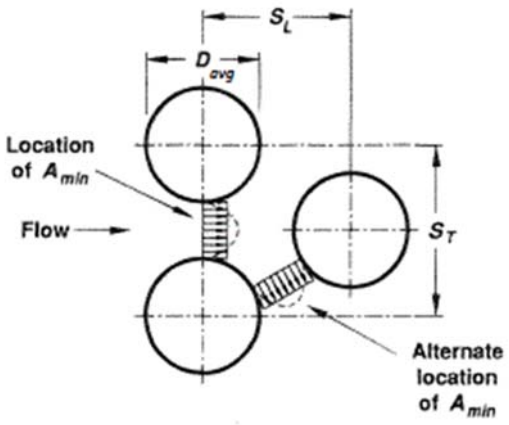


Figure 5: Location of minimum cross section for flow [1]

This maximum velocity V_{max} is expressed in one of two ways, depending on the location of A_{min} in the bank of pins:

$$V_{max} = \begin{cases} u_{in}\dfrac{S_T}{2(S_D - \overline{D})} & \text{if} \quad S_D < \dfrac{S_T + \overline{D}}{2} \\ u_{in}\dfrac{S_T}{(S_T - \overline{D})} & \text{otherwise} \end{cases} \tag{5}$$

where u_{in} is the inlet (i.e. free flow) velocity and S_D is the diagonal pitch, or the center-to-center distance between a pin in an odd row and its closest even row pin neighbor. Mathematically, S_D is defined by:

$$S_D = \sqrt{\left(S_T/2\right)^2 + S_L^2} \tag{6}$$

2.3.2 Pressure Drop and Friction Factor

The pressure drop of the fluid Δp defined by the Žukauskas relation [2], is:

$$\Delta p = N_L \chi \left(\frac{\rho u_{ref}^2}{2} \right) f \tag{7}$$

where N_L is the number of rows in the bank, χ is the correction factor, f is the friction factor, and $u_{ref} = V_{\max}$. Both f and χ can be interpolated from plotted data, shown in Figure 8. The friction factor f is plotted against Re for staggered arrays with an equilateral triangle arrangement (i.e. when $S_D = S_T$) for various dimensionless transverse pitches $\overline{S_T}$. The correction factor χ allows results to be applied to other staggered arrangements and is plotted against ratio of transverse to longitudinal pitch S_T / S_L for various Re. Data for f and χ are stored within and found using Engineering Equation Solver (EES) software [3].

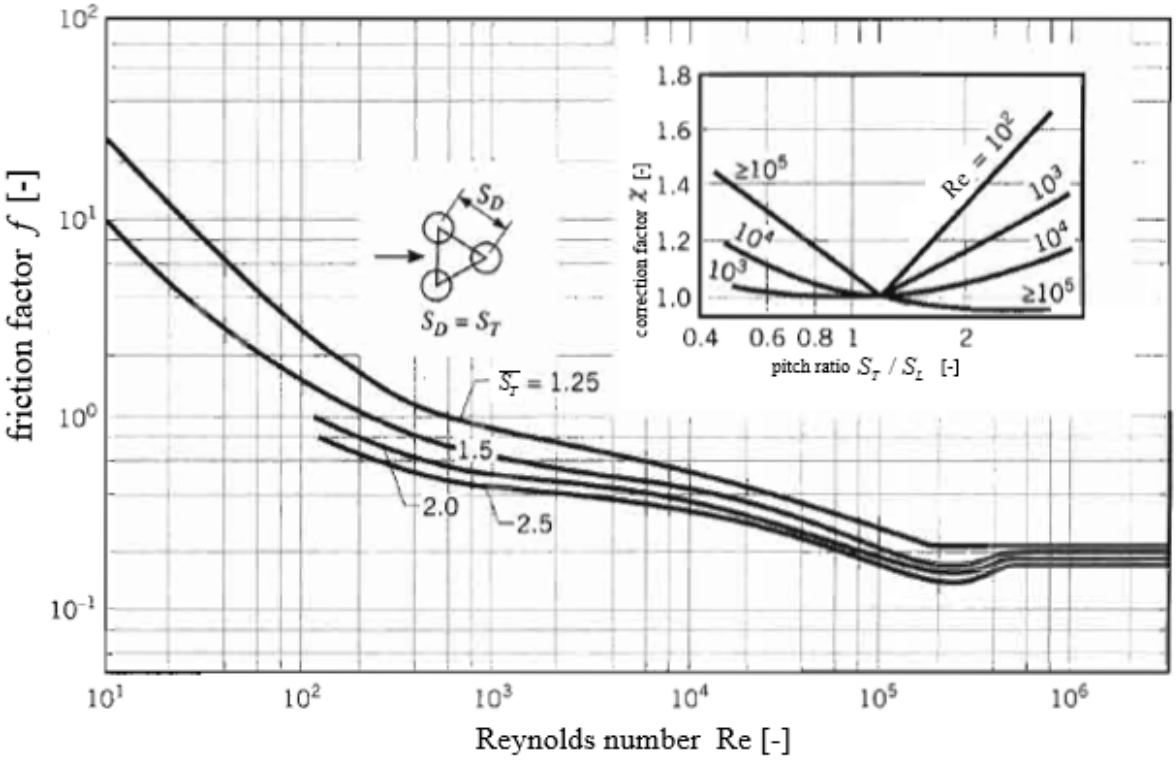


Figure 6: Friction factor f versus Reynolds number Re and correction factor χ versus pitch ratio S_T / S_L for external flow past a staggered bank of tubes [2]

2.3.3 Heat Transfer Coefficient and Nusselt Number

When simulating external flow across straight cylinder pins, the resulting average heat transfer coefficient $\bar{h}$ and pressure drop Δp were compared to accepted values, which were derived from previous publications by Žukauskas and Churchill and Bernstein. The remainder of this section will discuss how these accepted values were calculated. Since fluid properties (including thermal conductivity k, density ρ, and molecular viscosity μ) are held constant across simulations, they were set to the same constant values when deriving accepted values.

The accepted value for average heat transfer coefficient $\bar{h}$ was found using:

$$\bar{h} = \frac{\overline{\mathrm{Nu}}\ k}{D} \tag{8}$$

where $\overline{Nu}$ is the average Nusselt number derived from accepted relations and *D* is the diameter of the cylinders in the bank.

Method #1 (F. P. Incopera, D. P. DeWitt, et. al [4])

The accepted average Nusselt number $\overline{Nu}$ was calculated in one of two ways pending on Reynolds number. For low Reynolds numbers (that is, $\mathrm{Re} \le 100$), a modified version of Žukauskas's relation [4] for staggered banks was used, whereas higher Reynolds number ($100 < \mathrm{Re} \le 1000$) simulations used Churchill and Bernstein's equation [5] for flow across a single cylinder.

When $\mathrm{Re} \le 100$, $\overline{Nu}$ is calculated using Žukauskas's relation, which is defined as follows:

$$\overline{\mathrm{Nu}} = C\,\mathrm{Re}^{m}\,\mathrm{Pr}^{0.36}\left(\mathrm{Pr}/\mathrm{Pr}_s\right)^{0.25} C_2 \tag{9}$$

$$C = 0.90, m = 0.40$$
$$\mathrm{Pr}^{0.36}\left(\mathrm{Pr}/\mathrm{Pr}_s\right)^{0.25} = 0.88$$
$$C_2 = 0.89$$

where *C* and *m* are dimensionless constants, defined for ranges of *Re*, and C_2 is an additional dimensionless correction factor, defined by number of rows in the bank N_L, for banks with $N_L \le 20$. The constants are set to $C = 0.90$ and $m = 0.40$, as specified by the relation for low Re (within the $10 < \mathrm{Re} \le 100$ range) flow across banks with staggered configurations. For air,

$\mathrm{Pr}^{0.36}\left(\frac{\mathrm{Pr}}{\mathrm{Pr}_s}\right)^{0.25} = 0.88$. The correction factor is set to $C_2 = 0.89$, as specified by the relation for $N_L = 4$.

When $100 \le \mathrm{Re} \le 1000$, $\overline{\mathrm{Nu}}$ can be approximated using Churchill and Bernstein's equation [5] for flow across a single (isolated) cylinder. This comprehensive equation covers the entire range of *Re* and is given by:

$$\overline{\mathrm{Nu}} = 0.3 + \frac{0.62\,\mathrm{Re}_{u_{in}}^{1/2}\,\mathrm{Pr}^{1/3}}{[1+(0.4/\mathrm{Pr})^{2/3}]^{1/4}}\left[1+\left(\frac{\mathrm{Re}_{u_{\mathrm{inf}}}}{282,000}\right)^{5/8}\right]^{4/5} \tag{10}$$

where $\mathrm{Re}_{u_{in}}$ and *Pr* are evaluated using fluid properties that are held constant in this work (rather than properties evaluated at the film temperature) and $\mathrm{Re}_{u_{\mathrm{inf}}}$ is the Reynolds number calculated using u_{inf} (Eq. (3)).

Method #2 (Žukauskas)

Alternatively, the average Nusselt number can be calculated from the following relations for laminar flow in staggered banks, as defined directly by Žukauskas. At $1.6 < \mathrm{Re} \le 40$, $C = 1.04$ and $m = 0.4$, and at $40 < \mathrm{Re} \le 1000$, C=0.71 and m=0.5. By plugging these values into Eq. (9), we get:

$$\overline{\mathrm{Nu}} = \begin{cases} 1.04\,\mathrm{Re}^{0.4}\,\mathrm{Pr}^{0.36}\left(\mathrm{Pr}/\mathrm{Pr}_s\right)^{0.25} C_2 & \textit{for} \quad 1.6 < \mathrm{Re} \le 40 \\ 0.71\,\mathrm{Re}^{0.5}\,\mathrm{Pr}^{0.36}\left(\mathrm{Pr}/\mathrm{Pr}_s\right)^{0.25} C_2 & \textit{for} \quad 40 < \mathrm{Re} \le 1000 \end{cases} \tag{11}$$

$$\mathrm{Pr}^{0.36}\left(\mathrm{Pr}/\mathrm{Pr}_s\right)^{0.25} = 0.88$$

$$C_2 = 0.94 \pm 0.01$$

The correction factor was interpolated using an online plot digitizer from Figure 8, which plots C_2 versus N_L. As shown circled in Figure 8, $C_2 = 0.94 \pm 0.01$ when $N_L = 4$ (valid for all banks in this work) for most of the Re range investigated in this work.

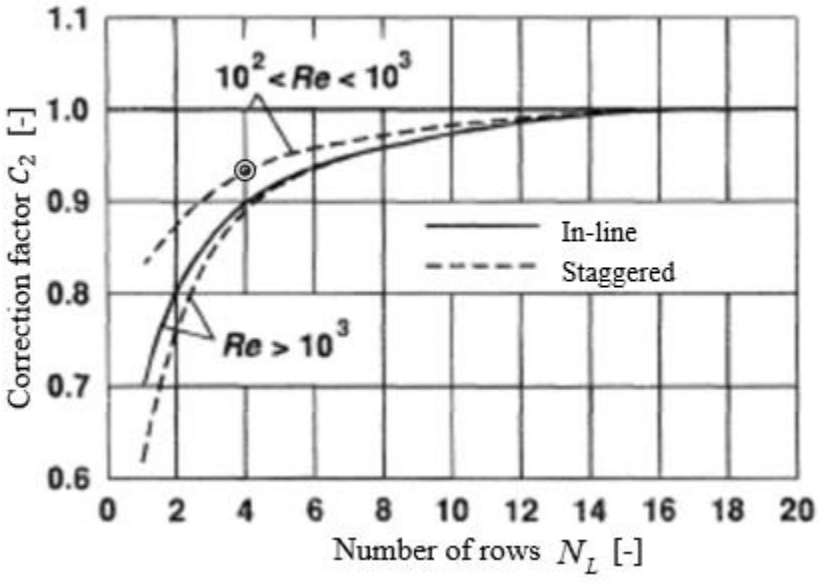


Figure 7: Correction factor C_2 versus number of rows N_L for in-line and staggered arrangements of pins at various ranges of Reynolds numbers [2]

3. SIMULATION OF AXIALLY-TAPERED PIN-FIN ARRAYS

To characterize the thermal performance of the pins and the base separately, two cases are investigated in this work: flow across heated pins sandwiched between adiabatic bases (from case 1) and flow across adiabatic pins sandwiched between heated bases (case 2). Any heated face zone was kept at a constant surface temperature T_s = 400 K. Both cases are shown in Figure 8.

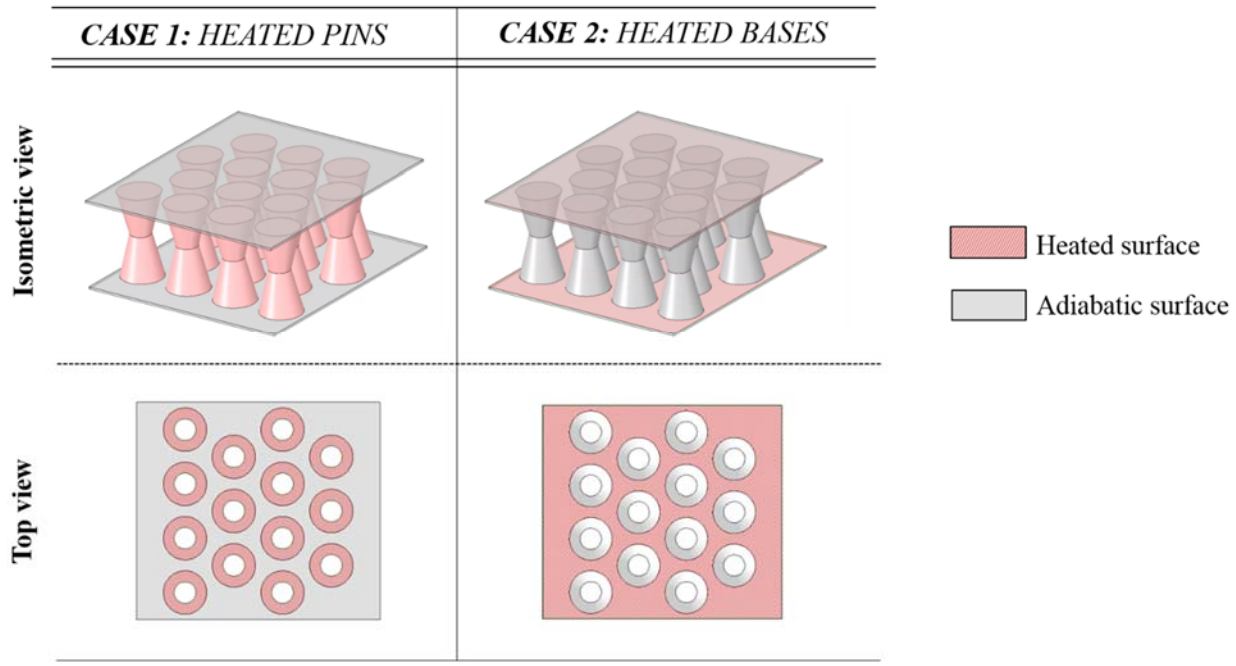


Figure 8: Cases being simulated: heated pins (case 1) on left and heated bases (case 2) on right

3.1. Simulated Region

When modeling each bank of pins, the goal was to accurately consider the defining features (i.e., the geometric layout and shape of its pins) while taking advantage of inherent symmetries to minimize the computational domain.

The banks of tubes were modeled under the assumption that each row extended infinitely. This assumption was based on the fact that banks of pins in practical engineering applications, like heat exchangers, have many cylinders in each row. This assumption allows for the identification of symmetry planes within the geometric layout. These planes of symmetry are coincident with all of the pin centers that fall in-line parallel to the flow. Using these lines of symmetry, the computational domain was reduced to a subsection sandwiched between neighboring symmetrical planes, as shown in Figure 9.

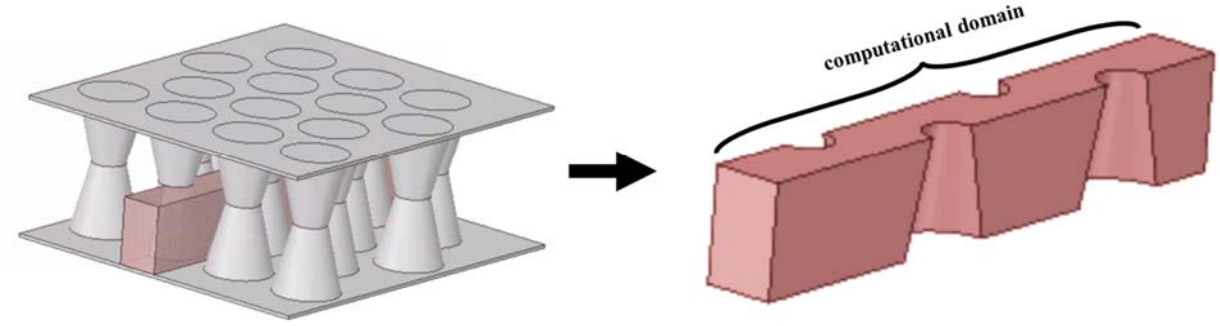


Figure 9: Simulated region (i.e. computational domain) of bank

3.2. Overview of Simulation Methods and Software Workflow

The following steps were used to setup and run a given simulation:

1. Model the geometry using DesignModeler within an ANSYS Workbench file.
2. Generate the mesh using Meshing within the ANSYS Workbench file.
3. Run CFD (Computational Fluid Dynamics) simulation using ANSYS Fluent.

To run the full array of simulations, UW-Madison's CHTC (Center for High-Throughput Computing) was utilized. Section 5 of this document describes how steps 1-3 were adapted for utilizing UW-Madison's CHTC (Center for High-Throughput Computing).

3.3. Geometry Generation

Each geometry was generated using ANSYS DesignModeler. Every geometry included inlet and outlet air blocks, in addition to the computational domain, so that simulation data is not impacted by any obscurities within the flow at the entrance or exit of the simulation. These obscurities, or non-physical flow behaviors, can occur in simulations attempting to balance momentum and energy and resolve fluid flow between 3-D computational domain elements and abstract 2-D inlet and outlet surface nodes. Obscurities in flow can occur at the entrance when solid obstructions are modeled too close to the inlet (which can cause highly-nonuniform stagnation properties across the inlet plane) and at the exit when the flow is more turbulent and chaotic (which can result in backflow at the outlet plane) [6]. The inlet air block provided space between the inlet and first pin to maximize uniformity of inflow stagnation properties at the entrance. Whereas, the outlet air block provided distance between the computational domain and outlet to help straighten exiting flow (i.e., minimize backflow) and provide a buffer for any backflow. The inlet and outlet air blocks and computational domain of a bank of tapered pins geometry are shown in Figure 10. The base diameter D, the minimum diameter D_{min}, the height h, the longitudinal pitch S_L , and transverse pitch S_T are also shown in Figure 10.

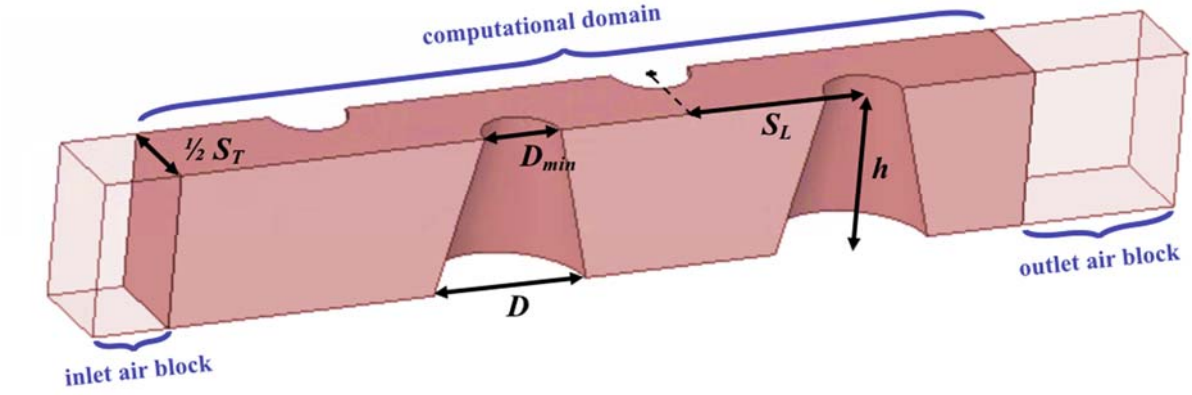


Figure 10: The inlet and outlet air blocks (shown as transparent) and computational domain (shown as a solid) of a bank of tapered pins

By parametrically varying D_{min}, h, S_L, and S_T within ANSYS DesignModeler, each geometry could be modeled. The combination of S_L and S_T determined the geometric layout whereas D_{min} and h determined the pin shape. This work generated geometries with the parameter values listed in Table 1. For all geometries, D = 2 mm.

Table 1: Values for Geometric Parameters.

		Transverse pitch S_T [mm]					
		1.25	1.5	1.75	2	2.25	2.5
Longitudinal pitch S_L [mm]	0.625			x	x	x	x
	0.75		x	x	x	x	x
	0.875	x	x	x	x	x	x
	1.125	x	x	x	x	x	x
	1.5	x	x	x	x	x	x
	2	x	x	x	x	x	x

Degree of taper T	0	0.25	0.5	0.75	1
D_{min} [mm]	2	1.5	1	0.5	0

Dimensionless height H	0.5	1.25	2	4	6
h [mm]	1	2.5	4	8	12

All combinations of S_L and S_T specified in Table 1 were modeled, which amounted to 33 unique geometric layouts. For each unique geometric layout, every combination of D_{min} and h were modeled. Therefore, a total of 825 unique geometries were modeled.

3.4. Mesh Generation

After the bank of pins geometry was modeled, it was imported into ANSYS Meshing, where a mesh was generated. In ANSYS Meshing, the Physics Preference was set to CFD and the Solver Preference was set to Fluent. The Relevance Center was set to Medium, the Max Face Size was set to 0.1 mm and the Max Size was set to 0.25 mm. Additionally, 6 Inflation layers were added to the surfaces highlighted in red in Figure 12 to capture boundary layer effects near the walls. All surfaces were inflated except for the inlet and outlet.

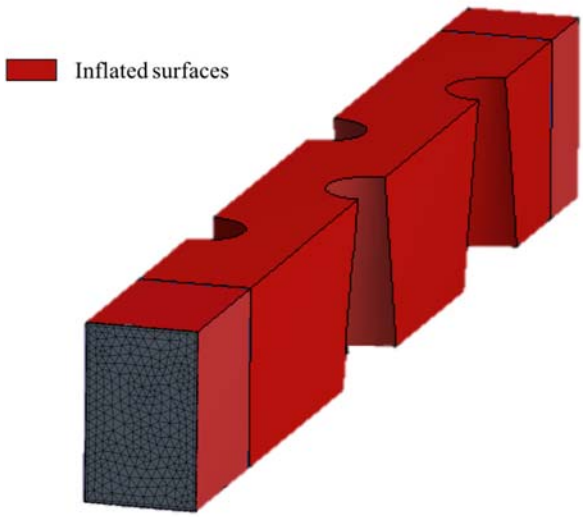


Figure 11: Inflation layers added to wall and symmetry surfaces of mesh of a bank of tapered pins.

The average orthogonal skew for all meshes in this work was ≤ 0.3 [-] where, on a scale between 0 and 1, 0 corresponds to lowest skew. Each mesh contains one interior fluid cell zone, which contains the computational domain and the inlet and outlet air blocks. The mesh also includes two interfaces (Start and End), which separate the computational domain from the inlet and outlet air blocks.

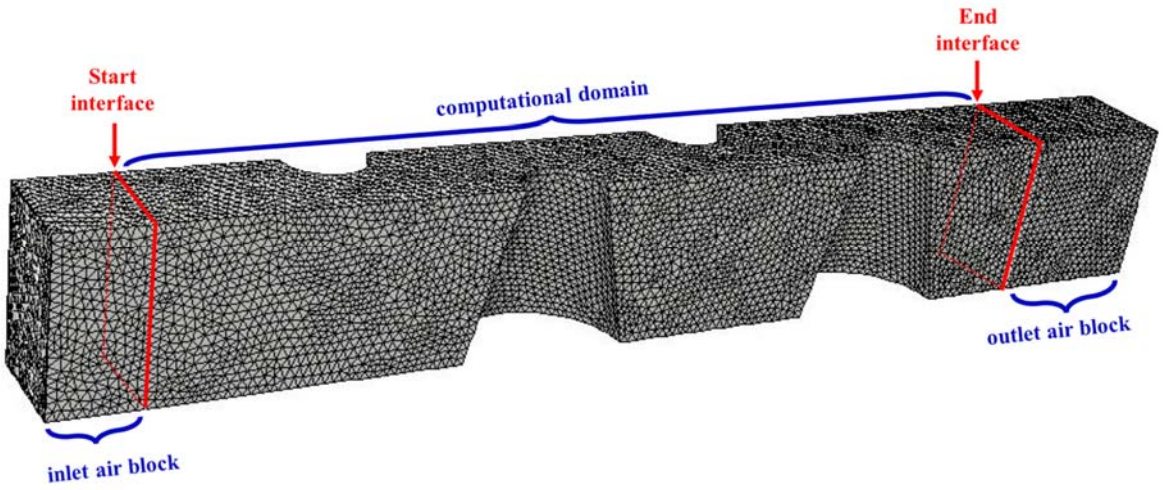


Figure 12: Mesh of a bank of tapered pins.

In addition to the interior fluid cell zone and Start and End interfaces, each mesh also included eight boundary face zones (In, Out, Pins, Sides, Top of Computational Domain, Top outside Computational Domain, Bottom of Computational Domain, and Bottom outside Computational Domain), which are labeled in Figure 13.

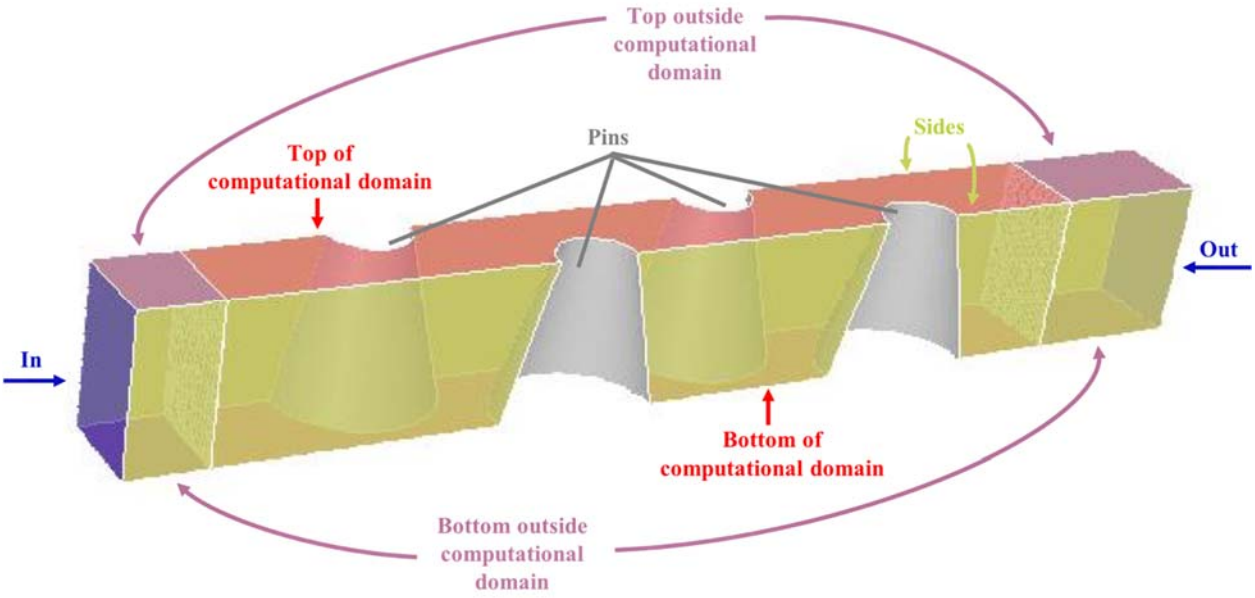


Figure 13: Boundary face zones of a bank of tapered pins.

The mesh's physical accuracy was ensured via simulating external flow across a bank of straight cylinder pins and comparing the results with predicted values from the correlations. The quality and resolution of the mesh also played a role in its generation. These considerations are discussed in Sections 3.4.1 and 3.4.2.

3.4.1 Comparing Simulation Results to Cylinder Correlations from Previous Works

To ensure the mesh (and simulation methods) were providing physically accurate results, external flow past a bank of cylinder pins was modeled and then compared to previous works using the relations provided in Section 2. Specifically, the values for pressure drop and heat transfer coefficient were compared across the full range of Reynolds numbers in this work. The results are shown in Figure 14.

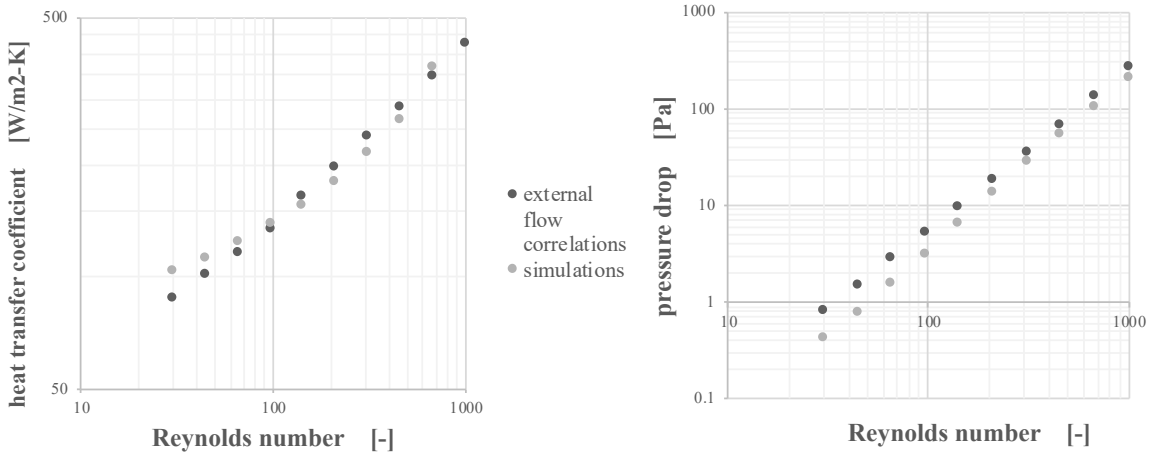


Figure 14: Pressure drop [Pa] and heat transfer coefficient [W/m^2-K] vs Reynolds number from simulation and from previous works.

3.4.2 Grid Independence Verification

The resolution of a mesh (i.e., how many elements are used to break up the computational domain) is an important consideration when running simulations because it influences computing time, file size, and the accuracy of results. A high resolution ensures accurate results but requires more computing resources – especially for 3-D simulations, like the simulations in this work. On the other hand, a low resolution reduces computing costs, but might fail to produce accurate results, thereby rendering them useless. The optimal resolution must be somewhere in the middle: fine enough to provide meaningful results but coarse enough to lessen computing costs.

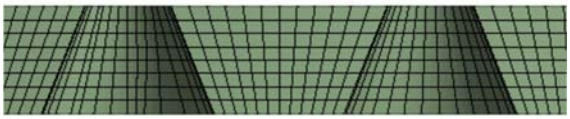
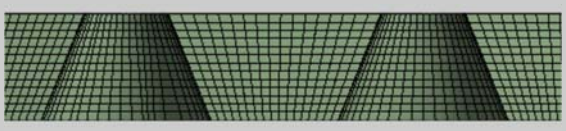
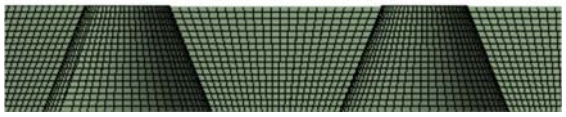
In addition to the global resolution of the mesh, the local resolution at the walls must also be considered for CFD simulations to ensure boundary layer effects are captured. If the mesh isn't fine enough at the wall, boundary layer effects won't be captured, and simulation results will be skewed. Inflation layers can be used to capture boundary layer effects.

To determine the optimal configuration of global and local mesh sizing, meshes with and without inflation were generated. For each case, the mesh resolution was incrementally increased until the values for pressure drop Δp and heat transfer coefficient $\overline{h}$ converged. Convergence was reached when the coarser mesh produced results that were less than 10% different than those from a very refined mesh. Using this process, the optimal resolution was determined for a bank with the following dimensionless parameters: $\overline{S_L}$ = 0.875, $\overline{S_T}$ = 1.75, T = 0.5, and H = 1.25.

A total of 14 meshes were generated (7 meshes with inflation and 7 meshes without it) spanning from low to high resolution were generated. The resolution of a mesh was determined by two input parameters: (1) the maximum face size of elements and (2) the maximum overall size of elements. As both input parameters decreased, the resolution increased and so did the

number elements of each mesh. The maximum face and overall element sizes and the resulting number of elements for all 14 meshes are presented in Table 2.

Table 2: The image, maximum face size, maximum size, number of elements, and average orthogonal quality of the meshes without inflation used to determine convergence.

			No Inflation	**With Inflation**
Mesh Image	Maximum Face Size [mm]	Maximum Size [mm]	Number of Elements	Number of Elements
	0.15	0.375	32,717	69,417
	0.1	0.25	90,642	182,757
	0.075	0.1875	196,551	366,055
	0.05	0.125	517,514	900,952
	0.0375	0.1	1,104,145	1,773,425
	0.025	0.0625	2,688,500	4,183,568

For each mesh described in

Table 2, the pressure drop and heat transfer coefficient were calculated. The results were then normalized by the values from the highest resolution mesh with inflation. Figure 15 and Figure 16 show the normalized pressure drop and normalized heat transfer coefficient plotted against the number of elements for meshes with and without inflation.

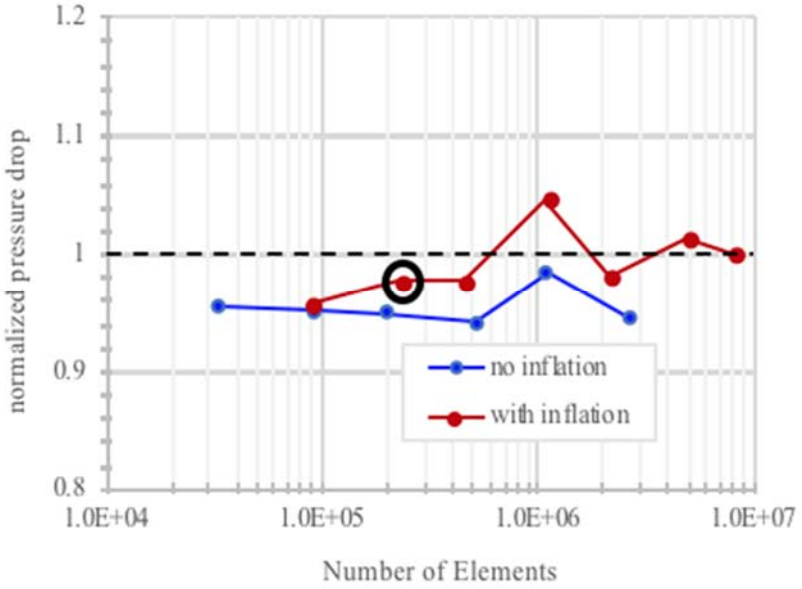


Figure 15: Normalized pressure drop [-] vs number of elements [-] for meshes with and without inflation. The values are normalized by the pressure drop from the highest resolution mesh with inflation.

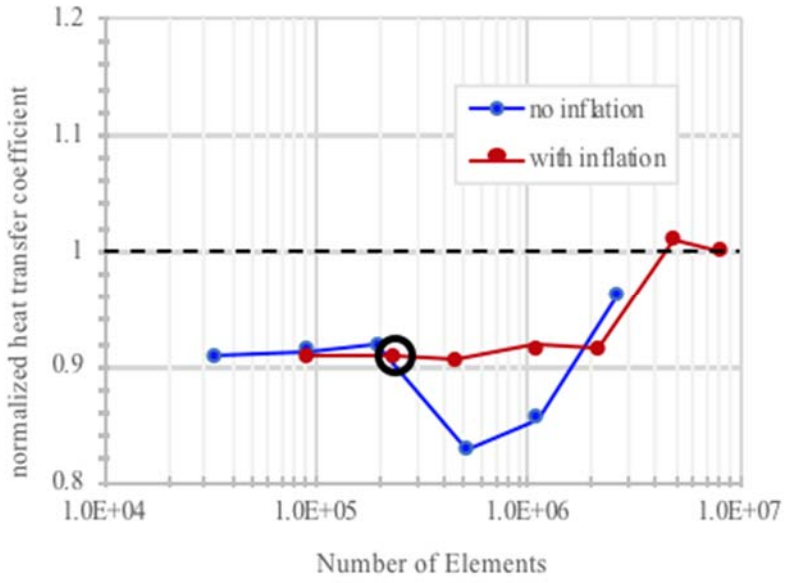


Figure 16: Normalized heat transfer coefficient [-] vs number of elements [-] for meshes with and without inflation. The values are normalized by the heat transfer coefficient from the highest resolution mesh with inflation.

As shown in Figure 15 and Figure 16, meshes with inflation (in red) provide more consistent results and converge with fewer elements than those without inflation (in blue). Figure 16 also shows a jump in heat transfer coefficient for meshes with greater than 1.0 x 10^6 elements, which could indicate that the inflation layers are not refined/thick enough to capture all boundary layer effects or the overall mesh is not fine enough. However, the meshes with inflation consistently simulated pressure drops and heat transfer coefficients within 5% and 10%, respectively, of the finest mesh. Therefore, meshes with inflation and Max Face Size of 0.1 mm and Max Size of 0.25 mm (circled in black in the plots) were used for all simulations in this work.

3.5. Computational Fluid Dynamics (CFD) Simulations

After the mesh was generated, it was imported into ANSYS Fluent, where the cell zone properties and boundary conditions were set and prepared for running simulations. Section 3.5.1 lists the equations that were being solved and Section 3.5.2 lists the cell and boundary conditions that were used in this work. Section 3.5.3 then describes the simulation process within ANSYS Fluent.

3.5.1 Modeling Criteria and Equations

All simulations in this work were modeled as being steady state. ANSYS Fluent solved a series of equations for each simulation including the continuity equation, momentum conservation equations, and energy conservation equation(s). These equations balanced mass, momentum, and energy, but different forms were used for laminar and turbulent flow. The remainder of this section lists and elaborates upon these equations.

The first equation that must be solved for each simulation is the continuity equation, or mass conservation equation. The continuity equation is defined by:

$$\frac{\partial}{\partial x}(\rho u_x)+\frac{\partial}{\partial y}(\rho u_y)+\frac{\partial}{\partial z}(\rho u_z)+\frac{\partial \rho}{\partial t}=0 \tag{12}$$

where u_x is the x-component of the velocity, u_y is the y-component of the velocity, and u_z is the z-component of the velocity. The second set of equations being solved ensures momentum is conserved in the x, y, and z directions for each simulation. These equations can be expressed as followed:

$$\rho\left[\frac{\partial u_x}{\partial t}+u_x\frac{\partial u_x}{\partial x}+u_y\frac{\partial u_x}{\partial y}+u_z\frac{\partial u_x}{\partial z}\right]=-\frac{\partial p}{\partial x}+\mu\left(\frac{\partial^2 u_x}{\partial x^2}+\frac{\partial^2 u_x}{\partial y^2}+\frac{\partial^2 u_x}{\partial z^2}\right) \tag{13}$$

$$\rho\left[\frac{\partial u_y}{\partial t}+u_x\frac{\partial u_y}{\partial x}+u_y\frac{\partial u_y}{\partial y}+u_z\frac{\partial u_y}{\partial z}\right]=-\frac{\partial p}{\partial y}+\mu\left(\frac{\partial^2 u_y}{\partial x^2}+\frac{\partial^2 u_y}{\partial y^2}+\frac{\partial^2 u_y}{\partial z^2}\right) \tag{14}$$

$$\rho\left[\frac{\partial u_z}{\partial t}+u_x\frac{\partial u_z}{\partial x}+u_y\frac{\partial u_z}{\partial y}+u_z\frac{\partial u_z}{\partial z}\right]=-\frac{\partial p}{\partial z}+\mu\left(\frac{\partial^2 u_z}{\partial x^2}+\frac{\partial^2 u_z}{\partial y^2}+\frac{\partial^2 u_z}{\partial z^2}\right) \tag{15}$$

In addition to mass and momentum, energy must also be conserved. Therefore, the final equation that must be solved for all simulations is the energy balance, which can generally be expressed by the following:

$$\begin{aligned}\rho c\left[\frac{\partial T}{\partial t}+u_x\frac{\partial T}{\partial x}+u_y\frac{\partial T}{\partial y}+u_z\frac{\partial T}{\partial z}\right]&=k\left[\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right]+\ldots\\ \mu\left[\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)^2+2\left(\left(\frac{\partial u_x}{\partial x}\right)^2+\left(\frac{\partial u_y}{\partial x}\right)^2+\left(\frac{\partial u_z}{\partial x}\right)^2\right)\right]&\end{aligned} \tag{16}$$

3.5.2 Setting Cell Zone and Boundary Conditions

After the modeling criteria were defined, the fluid cell and boundary face zones were set. For all simulations in this work, the fluid cell zone was set to air and Fluent's default properties for air were used: a density ρ = 1.225 kg/m^3, a heat capacity c_p = 1006.433 J/kg-K, a conductivity k = 0.0242 W/m-K, and a viscosity μ = 1.7894 x 10^{-5} kg/m-s[2].

The model's eight boundary conditions, each corresponding to a face zone, were then set. The sides, the top of the computational domain, and the top outside the computational domain

[2] These constant values were also used in all calculations.

were given symmetry boundary conditions. The bottom of the computational domain, the bottom outside the computational domain, and the pins were modeled as walls. To simulate case 1 (heated pins), the pins were modeled as walls with uniform surface temperature T_s = 400 K and the bottom of the computational domain was modeled as an adiabatic wall. To simulate case 2 (heated bases), the pins were modeled as adiabatic walls and the bottom of the computational domain was modeled as a wall with uniform surface temperature T_s = 400 K. The boundary condition assignments are labeled in Figure 17.

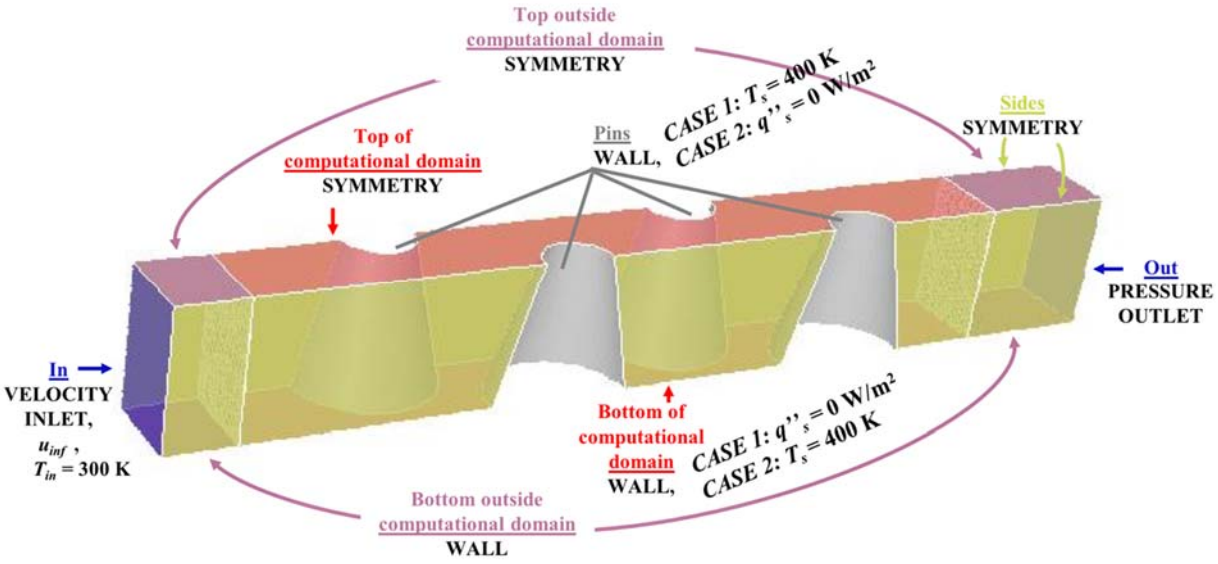


Figure 17: Boundary zone assignments for a bank of tapered pins.

For both cases, air entered at plane In with a uniform velocity u_{in} and temperature T_{in} = 300 K then traversed through the inlet air block, Start interface, the Computational Domain, End interface, and finally exited at plane Out, which was modeled as a pressure-outlet with a gauge pressure p_{out} = 0 Pa.

3.5.3 Simulation Process

Each ANSYS Fluent simulation was controlled by a journal file, which included commands for setting cell zones, boundary conditions and solver settings. The journal file also set convergence criteria for continuity and energy equation residuals and included commands that report values of interest by writing them to an output file. A typical journal file, **jouneric.jou**, had the following code:

```
    ;~~~ general setup ~~~
    /file/start-transcript outputfile.trn
    file rc "mesheric.msh"
    solve set equations flow y
    define models energy y n y n y
    solve set equations temp y
    define models solver pressure-based y
    define models viscous laminar y

    ;~~~ boundary conditions ~~~
    ;~~~ both cases
    define bc zone-type top_cd symmetry
    define bc zone-type top_outside_cd symmetry
    define bc zone-type sides symmetry

    define bc zone-type out pressure-outlet

    define bc zone-type in velocity-inlet
    define bc velocity-inlet in n n y y n u_inf n 0 n 300


    ;~~~ CASE 1: heated pins & adiabatic bases
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    ;~~~ boundary conditions ~~~
    define bc wall pins 0 n 0 n y temp n 400 n n n n ,

    ;~~~ initialization settings
    solve initialize set-defaults x-velocity u_inf
    solve initialize set-defaults temp 300
    solve initialize initialize-flow

    ;~~~ solver settings ~~~
    solve monitors residual conv-crit , , , , .00001
    solve set equations flow y
    solve set equations temp y
    solve monitors surface set-monitor tm_end "Mass-Weighted Average" temp
end-src , y 2 n n 1
    solve monitors surface set-monitor p_start "Area-Weighted Average"
pressure start-src , y 3 n n 1
```

```
    solve monitors surface set-monitor p_end "Area-Weighted Average" pressure
end-src , y 3 n n 1

    ;~~~ solve ~~~
    solve iterate 1000

    ;~~~ extract data ~~~
    ;~~~ both cases
    report surface-integrals area-weighted-avg in , velocity y "generic_uinf"
    report surface-integrals mass-flow-rate in , y "generic_mdot"
    report surface-integrals area-weighted-avg start-src end-src , pressure y
"generic_p"
    ;~~~ case 1
    report surface-integrals mass-weighted-avg start-src end-src , temp y
"case1_generic_tm"
    ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    ;~~~ CASE 2: heated bases & adiabatic pins
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    ;~~~ boundary conditions ~~~
    define bc wall pins 0 n 0 n y heat-flux n 0 n n n n ,
    define bc wall bot_cd 0 n 0 n y temp n 400 n n n n ,

    solve initialize initialize-flow ,

    ;~~~ solve ~~~
    solve iterate 1000

    ;~~~ extract data ~~~
    ;~~~ case 2
    report surface-integrals mass-weighted-avg start-src end-src , temp y
"case2_generic_tm"
    ;~~~ both cases
    report surface-integrals area pins bot_cd , y "mesheric_area"
    ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    /file/write-case-data generic

    /file/stop-transcript

    exit
    y
```

Before **jouneric.jou** could be used, the highlighted variables (i.e., **mesheric**, **u_inf**, and **generic**) needed to be substituted with simulation-specific values: **mesheric** was replaced with the name of the geometry, **u_inf** was replaced with the inlet velocity, and **generic** was replaced with the name of the specific simulation.

The journal file was also fundamental to running Fluent simulations on the CHTC server. Section 5 of this document explains how the journal file was used to run simulations on the CHTC server.

3.6. Data Extraction

The data from each simulation were extracted from six ANSYS Fluent output files, which reported the pin and base surface areas, the velocity magnitude at the inlet, the mass flow rate at the inlet, the pressure of the fluid at the Start and End mesh interfaces, and the temperature of the fluid at the Start and End mesh interfaces for both cases. All output files were formatted according to ANSYS Fluent's built-in template for Surface Integral Reports. The six output files had the following formats:

1. Area output file

```
                        "Surface Integral Report"

                           Area                 (m2)
------------------------------ --------------------
                           pins        1.9419131e-05
                         bot_cd        9.0397493e-06
                --------------- --------------------
                            Net         2.845888e-05
```

2. Velocity magnitude output file

```
                        "Surface Integral Report"

          Area-Weighted Average
             Velocity Magnitude                (m/s)
------------------------------ --------------------
                             in            1.5371165
```

3. Mass flow rate output file

"Surface Integral Report"

Mass Flow Rate	(kg/s)
in	6.5903871e-06

4. Pressure output file

"Surface Integral Report"

Area-Weighted Average Static Pressure	(pascal)
start-src	18.622483
end-src	-1.3301565
Net	8.6461633

5. Case 1 (heated pins) temperature output file

"Surface Integral Report"

Mass-Weighted Average Static Temperature	(k)
start-src	300.00103
end-src	341.37191
Net	321.70733

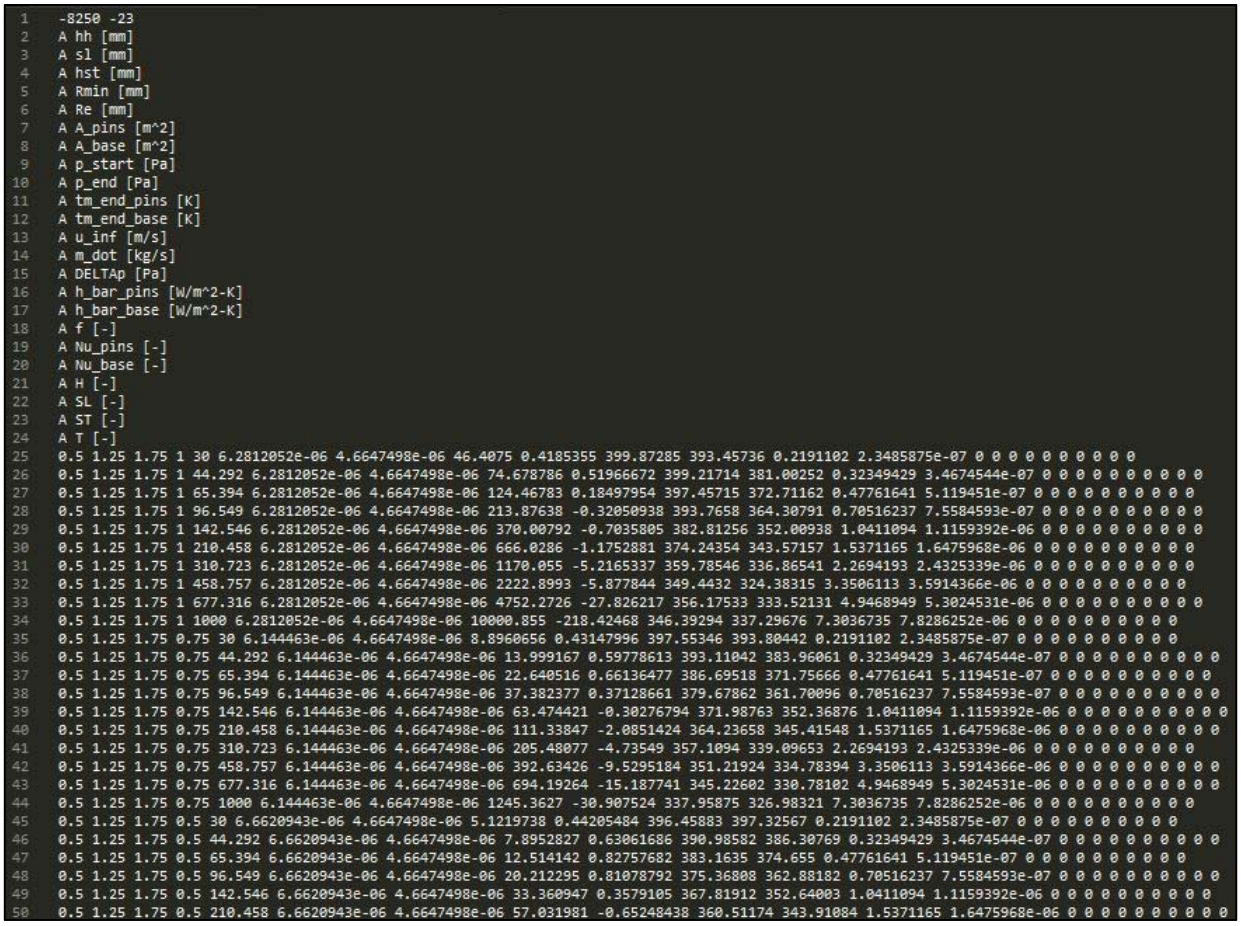
6. Case 2 (heated bases) temperature output file

"Surface Integral Report"

Mass-Weighted Average Static Temperature	(k)
start-src	300.22964
end-src	315.96174
Net	308.49749

Using a Python code, the data were read from all output files and parsed into a single text file, which could later be opened as a lookup table in EES (Engineering Equation Solver). The Python code, **`parse_output_data.py`**, is provided in Section 5.4.

Each output data file contained 8,275 lines, which represented the format and contents of an EES Lookup file. The first 50 lines of an output data file and shown in
Figure 18. Line 1 specified the number of rows (i.e., 8,250) and columns (i.e., 23) in the Lookup table. Lines 2-24 specified the format, title, and units of all 23 columns. Finally, lines 25-8,275 specified the contents of the Lookup table. Each line corresponded to a specific simulation and row in the Lookup table. Table values of 999 indicated missing data and values of 0 indicated values that were to be calculated in EES.



```
-8250 -23
A hh [mm]
A sl [mm]
A hst [mm]
A Rmin [mm]
A Re [mm]
A A_pins [m^2]
A A_base [m^2]
A p_start [Pa]
A p_end [Pa]
A tm_end_pins [K]
A tm_end_base [K]
A u_inf [m/s]
A m_dot [kg/s]
A DELTAp [Pa]
A h_bar_pins [W/m^2-K]
A h_bar_base [W/m^2-K]
A f [-]
A Nu_pins [-]
A Nu_base [-]
A H [-]
A SL [-]
A ST [-]
A T [-]
0.5 1.25 1.75 1 30 6.2812052e-06 4.6647498e-06 46.4075 0.4185355 399.87285 393.45736 0.2191102 2.3485875e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 1 44.292 6.2812052e-06 4.6647498e-06 74.678786 0.51966672 399.21714 381.00252 0.32349429 3.4674544e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 1 65.394 6.2812052e-06 4.6647498e-06 124.46783 0.18497954 397.45715 372.71162 0.47761641 5.119451e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 1 96.549 6.2812052e-06 4.6647498e-06 213.87638 -0.32050938 393.7658 364.30791 0.70516237 7.5584593e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 1 142.546 6.2812052e-06 4.6647498e-06 370.00792 -0.7035805 382.81256 352.00938 1.0411094 1.1159392e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 1 210.458 6.2812052e-06 4.6647498e-06 666.0286 -1.1752881 374.24354 343.57157 1.5371165 1.6475968e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 1 310.723 6.2812052e-06 4.6647498e-06 1170.055 -5.2165337 359.78546 336.86541 2.2694193 2.4325339e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 1 458.757 6.2812052e-06 4.6647498e-06 2222.8993 -5.877844 349.4432 324.38315 3.3506113 3.5914366e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 1 677.316 6.2812052e-06 4.6647498e-06 4752.2726 -27.826217 356.17533 333.52131 4.9468949 5.3024531e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 1 1000 6.2812052e-06 4.6647498e-06 10000.855 -218.42468 346.39294 337.29676 7.3036735 7.8286252e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 30 6.144463e-06 4.6647498e-06 8.8960656 0.43147996 397.55346 393.80442 0.2191102 2.3485875e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 44.292 6.144463e-06 4.6647498e-06 13.999167 0.59778613 393.11042 383.96061 0.32349429 3.4674544e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 65.394 6.144463e-06 4.6647498e-06 22.640516 0.66136477 386.69518 371.75666 0.47761641 5.119451e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 96.549 6.144463e-06 4.6647498e-06 37.382377 0.37128661 379.67862 361.70096 0.70516237 7.5584593e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 142.546 6.144463e-06 4.6647498e-06 63.474421 -0.30276794 371.98763 352.36876 1.0411094 1.1159392e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 210.458 6.144463e-06 4.6647498e-06 111.33847 -2.0851424 364.23658 345.41548 1.5371165 1.6475968e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 310.723 6.144463e-06 4.6647498e-06 205.48077 -4.73549 357.1094 339.09653 2.2694193 2.4325339e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 458.757 6.144463e-06 4.6647498e-06 392.63426 -9.5295184 351.21924 334.78394 3.3506113 3.5914366e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 677.316 6.144463e-06 4.6647498e-06 694.19264 -15.187741 345.22602 330.78102 4.9468949 5.3024531e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.75 1000 6.144463e-06 4.6647498e-06 1245.3627 -30.907524 337.95875 326.98321 7.3036735 7.8286252e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.5 30 6.6620943e-06 4.6647498e-06 5.1219738 0.44205484 396.45883 397.32567 0.2191102 2.3485875e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.5 44.292 6.6620943e-06 4.6647498e-06 7.8952827 0.63061686 390.98582 386.30769 0.32349429 3.4674544e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.5 65.394 6.6620943e-06 4.6647498e-06 12.514142 0.82757682 383.1635 374.655 0.47761641 5.119451e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.5 96.549 6.6620943e-06 4.6647498e-06 20.212295 0.81078792 375.36808 362.88182 0.70516237 7.5584593e-07 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.5 142.546 6.6620943e-06 4.6647498e-06 33.360947 0.3579105 367.81912 352.64003 1.0411094 1.1159392e-06 0 0 0 0 0 0 0 0 0 0
0.5 1.25 1.75 0.5 210.458 6.6620943e-06 4.6647498e-06 57.031981 -0.65248438 360.51174 343.91084 1.5371165 1.6475968e-06 0 0 0 0 0 0 0 0 0 0
```

Figure 18: Lines 1-50 of an output text file.

4. PERFORMANCE OF AXIALLY-TAPERED PIN-FIN ARRAYS

4.1. Calculation of Performance Parameters from Simulations

After running simulations, the data were extracted from three output text files, which reported the pin surface area, the pressure of the fluid at the Start and End mesh interfaces, and the mean temperature of the fluid at the Start and End mesh interfaces. These data were used to calculate dimensional performance metrics including the pressure drop Δp between the computational domain's inlet and outlet and the heat transfer coefficient $\bar{h}$. Then these dimensional performance metrics were used to find the dimensionless quantities: friction factor f and average Nusselt number $\overline{Nu}$.

For each case being investigated for a bank of pins geometry, two dimensionless parameters were calculated: the friction factor and average Nusselt number. The friction factor is related to the pressure drop across a bank and the average Nusselt number is related to the average heat transfer coefficient. Both dimensionless parameters can be correlated against Reynolds numbers Re (note that the Prandtl number used for the simulations is consistent with air and not varied during this work as the only fluid of interest is air).

The pressure drop across the bank Δp was found each time the inlet velocity was updated (i.e., for the full range of Reynolds number) by subtracting the average outlet pressure p_{out} from the average inlet pressure p_{in}:

$$\Delta p = p_{in} - p_{out} \tag{17}$$

Values for p_{in} and p_{out} were extracted from the simulation by taking the area-weighted average pressure at the Start and End mesh interfaces, respectively.

The pressure drop, Δp , can be used to calculate the friction factor *f* as follows:

$$f = 2\frac{\Delta p}{\rho u_{\inf}{}^2 N_L} \tag{18}$$

where N_L is the number of rows ($N_L = 4$ in this work) and u_{ref} is the reference velocity of the fluid, which is defined in one of two ways depending on the pitch ratio S_T/S_L . For $S_T/S_L < 1.25$, $u_{ref} = u_{in}$. Whereas, for $S_T/S_L \geq 1.25$, $u_{ref} = V_{\max}$.

The average heat transfer coefficient $\bar{h}$ was also found for the full range of Re using the ε - NTU method. The effectiveness ε of a given heat exchanger is defined by:

$$\varepsilon = \frac{T_{out} - T_{in}}{T_s - T_{in}} \tag{19}$$

where T_s is the temperature of the heated surface and T_{in} and T_{out} are the temperatures of the fluid at the inlet and outlet, respectively. The values for T_s and T_{in} are held constant in calculations (T_s = 400 K and T_{in} = 300 K) just as they are kept constant across simulations. The value for T_{out} is extracted by taking the mass-weighted average temperature at the computational outlet. The effectiveness ranges from 0, least effective, to 1, most. After obtaining the effectiveness, it was used to find the number of transfer units, or NTU . For heat exchangers with one fluid and a constant wall temperature, NTU is defined as follows [7]:

$$NTU = -\ln(1-\varepsilon) \tag{20}$$

The result for NTU is then used to find the average heat transfer coefficient $\bar{h}$:

$$\bar{h} = \frac{NTU\ \dot{m}\ c_p}{A_s}. \tag{21}$$

where A_s is the area of the heated surface and $\dot{m}$ is the mass flow rate of the fluid.

The average Nusselt number $\overline{Nu}$ can be found using:

$$\overline{Nu} = \frac{\bar{h}\, D}{k} \tag{22}$$

where D is the base diameter. After obtaining a set of dimensionless parameters for a given bank, they can be correlated against the corresponding set of *Re*. These results are shown in Section 4.4.

4.2. Number of Rows Investigation

The correlations in this work are meant to be applied to banks with any number of rows. Thermal performance metrics are defined per heated area and hydrodynamic performance metrics are provided per row of pins. However, the number of rows is an important consideration when developing correlations for banks of pins because it determines computational (or material) expenses and can impact results. On one hand, simulating many rows is computationally expensive since the file size and run time both increase. On the other hand, simulating fewer rows conserves computational resources but the results might be impacted by entry and exit effects.

By varying the number of rows in a specific bank and comparing simulation results, the impact of the number of rows could be investigated. Specifically, a bank with H=1.25, $\overrightarrow{S_L}$ =0.75, $\overrightarrow{S_T}$ =1.5, and T =0.5 was modeled with 4, 6, 8 and 10 rows. For each bank, Reynolds numbers of 30, 210.5, 677.3, and 1000 were simulated. A total of 16 simulations were included in this investigation (4 Reynolds numbers for each of the 4 banks).

For each simulation, the pressure drop per pin, the heat transfer coefficient of the pins, and the heat transfer coefficient of the base were calculated (shown in Table 3, Table 4, and Table 5, respectively). The net and percent differences in results from the 4- and 10-row banks were also calculated for each simulated Reynolds number.

Table 3: Pressure drop per pin from banks with 4, 6, 8, and 10 rows of pins.

		pressure drop per pin [Pa]					
		4 rows	6 rows	8 rows	10 rows	**diff b/w 4 and 10 rows [Pa]**	**% diff b/w 4 and 10 rows**
Reynolds number [-]	30	0.3661	0.3613	0.3598	0.3613	0.0048	1.33 %
	210.5	5.684	5.612	5.543	5.512	0.172	3.12 %
	677.3	41.22	38.43	37.67	36.39	4.83	13.3 %
	1000	77.86	74.69	71.76	70.03	7.83	11.2 %

Table 4: Heat transfer coefficient of pins from banks with 4, 6, 8, and 10 rows of pins.

		heat transfer coefficient of pins [W/m^2-K]				**diff b/w 4 and 10 rows [W/m^2-K]**	**% diff b/w 4 and 10 rows**
		4 rows	6 rows	8 rows	10 rows		
Reynolds number [-]	30	93.96	94.96	95.71	96.21	2.25	2.34 %
	210.5	197.2	198.6	200.9	202.1	4.9	2.4 %
	677.3	350.7	347.4	351.8	357.6	6.9	1.9 %
	1000	429.2	408.8	429.2	438	8.8	2.0 %

Table 5: Heat transfer coefficient of bases from banks with 4, 6, 8, and 10 rows of pins.

		heat transfer coefficient of base [W/m^2-K]				**diff b/w 4 and 10 rows [W/m^2-K]**	**% diff b/w 4 and 10 rows**
		4 rows	6 rows	8 rows	10 rows		
Reynolds number [-]	30	59.63	59.14	61.46	65.27	5.64	8.64 %
	210.5	122.8	122.2	124.3	124.9	2.1	1.7 %
	677.3	257.1	256.3	264.7	259.3	2.2	0.85 %
	1000	347.7	309.3	358.9	338.3	9.4	2.8 %

The percent differences between results from the 4- and 10-row banks were used to evaluate the impact of entry and exit effects on results from the bank with 4 rows. A percent difference of 0% indicates no impact and larger percent differences indicate greater impacts. As shown in Table 3, Table 4, and Table 5, the percent difference ranged between 1.33% and 13.3% for the pressure drop per pin, between 1.9% to 2.4% for the heat transfer coefficient of the pins, and between 0.85% and 8.64% for the heat transfer coefficient of the base. The results from the 4-row bank were within 13.3% of the results from the 10-row bank, which is not within the ideal 10% range.

4.3. Correlation Development

4.3.1 Linear Regression Process

After calculating the friction factor *f* and the average Nusselt number for pins $\overline{Nu}_{pins}$ and bases $\overline{Nu}_{base}$ for each simulation, these quantities were correlated with flow condition (quantified by the Reynolds number, *Re*) and geometric parameters (which include the dimensionless longitudinal pitch, $\overline{S_L}$, the dimensionless transverse pitch, $\overline{S_T}$, the dimensionless height, H, and the degree of taper, T).

All three correlations were developed using EES' Linear Regression command, which performs regressions using data stored in tables. Specifically, the Lookup table containing simulation data and calculations was used. Before performing any linear regression, the natural logarithm was calculated for all eight dimensionless parameters and added to the Lookup table.

Each linear regression was performed using EES' Linear Regression command, which can be accessed in the "Tables" menu of the software's top menu bar. Once selected, a Linear Regression dialog window pops up, as shown in Figure 19. First, the Lookup table containing simulation data and calculations are selected. Then the dependent variable and independent variables are selected and the equation form is specified. The correlations in this work had a polynomial order of 2 and cross-terms were included. After selecting dependent and independent variables and the equation form, "Fit" was clicked to perform the linear regression.

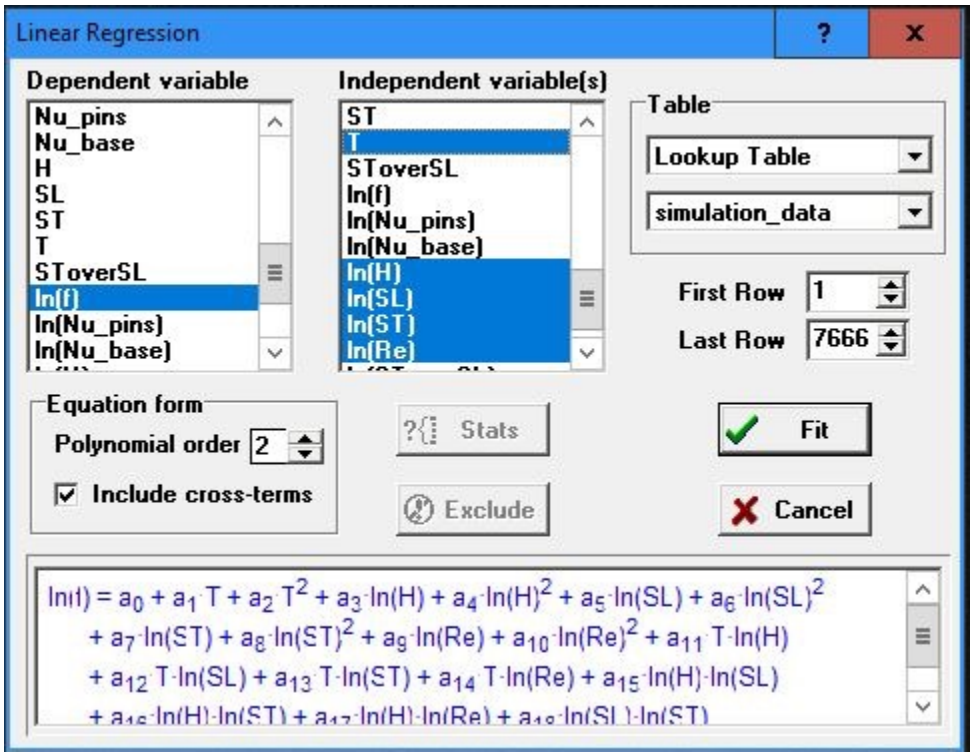


Figure 19: Linear Regression dialog window in EES

After a successful fit, the "Stats" button was clicked to open a Linear Regression Coefficients dialog window (shown in Figure 20), which provided a table of linear regression

coefficients and their corresponding standard errors, the bias of the fit, and the coefficient of determination, R^2. The standard error is the square root of a coefficient's estimated variance. The bias, or mean bias error, is the average discrepancy between the data and the predicted value from the correlation. The bias indicates a correlation's tendency to either overestimate (if the bias is positive) or underestimate (if the bias is negative). The R^2 value is the percentage of variance in the dependent variable that is explained by the independent variables. The value can range from 0% (none of the variation is explained by the correlation) to a perfect fit of 100% (all variation is explained by the correlation). Higher R^2 values indicate a better fit between the correlation and the data.

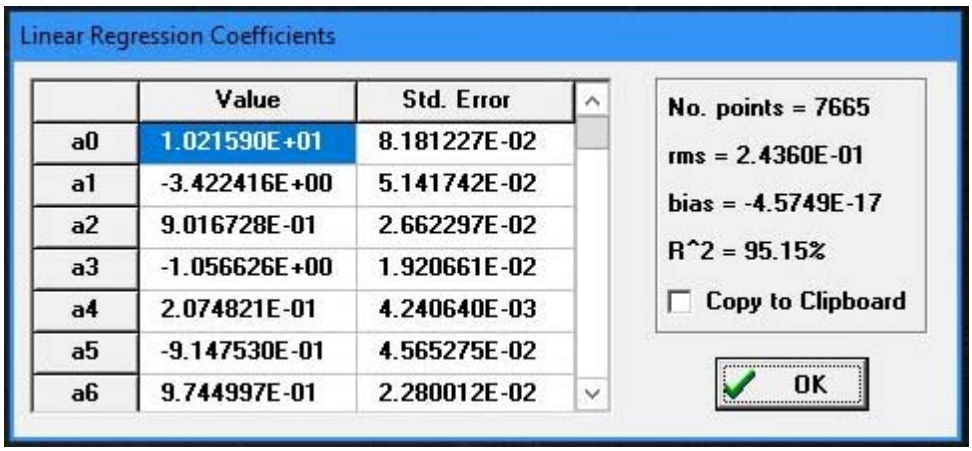


Figure 20: Linear Regression Coefficients dialog window in EES

4.4. Resulting Correlations

Each dimensionless performance parameter (i.e., f, $\overline{Nu}_{pins}$ and $\overline{Nu}_{base}$) was correlated with Re, $\overline{S_L}$, $\overline{S_T}$, H, and T.

$$f = f\left(\mathrm{Re}, S_L, S_T, H, T\right) \tag{23}$$

$$\overline{Nu}_{pins} = f\left(\mathrm{Re}, S_L, S_T, H, T\right) \tag{24}$$

$$\overline{Nu}_{base} = f\left(\mathrm{Re}, S_L, S_T, H, T\right) \tag{25}$$

The regression equation for all three performance parameters was a 2nd order polynomial with cross terms. These functional relations are defined as follows:

$$\begin{aligned}\ln(f) = {} & \ln(a_0) + a_1\ln(H) + a_2\ln\left(H\right)^2 + a_3\ln\left(\overline{S_L}\right) + a_4\ln\left(\overline{S_L}\right)^2 + a_5\ln\left(\overline{S_T}\right) + \\ & a_6\ln\left(\overline{S_T}\right)^2 + a_7\ln(\mathrm{Re}) + a_8\ln(\mathrm{Re})^2 + a_9 T + a_{10}T^2 + a_{11}\ln\left(L_c\right) + a_{12}\ln\left(L_c\right)^2 + \\ & a_{13}\ln\left(H\right)\ln\left(\overline{S_L}\right) + a_{14}\ln\left(H\right)\ln\left(\overline{S_T}\right) + a_{15}\ln\left(H\right)\ln\left(\mathrm{Re}\right) + a_{16}\ln\left(H\right)T + \\ & a_{17}\ln\left(H\right)\ln\left(L_c\right) + a_{18}\ln\left(\overline{S_L}\right)\ln\left(\overline{S_T}\right) + a_{19}\ln\left(\overline{S_L}\right)\ln\left(\mathrm{Re}\right) + a_{20}\ln\left(\overline{S_L}\right)T + \\ & a_{21}\ln\left(\overline{S_L}\right)\ln\left(L_c\right) + a_{22}\ln\left(\overline{S_T}\right)\ln\left(\mathrm{Re}\right) + a_{23}\ln\left(\overline{S_T}\right)T + a_{24}\ln\left(\overline{S_T}\right)\ln\left(L_c\right) + \\ & a_{25}\ln\left(\mathrm{Re}\right)T + a_{26}\ln\left(\mathrm{Re}\right)\ln\left(L_c\right) + a_{27}T\ln\left(L_c\right)\end{aligned} \tag{26}$$

$$\begin{aligned}\ln(\overline{Nu}_{pins}) = {} & \ln(b_0) + b_1\ln(H) + b_2\ln\left(H\right)^2 + b_3\ln\left(\overline{S_L}\right) + b_4\ln\left(\overline{S_L}\right)^2 + b_5\ln\left(\overline{S_T}\right) + \\ & b_6\ln\left(\overline{S_T}\right)^2 + b_7\ln(\mathrm{Re}) + b_8\ln(\mathrm{Re})^2 + b_9 T + b_{10}T^2 + b_{11}\ln\left(L_c\right) + b_{12}\ln\left(L_c\right)^2 + \\ & b_{13}\ln\left(H\right)\ln\left(\overline{S_L}\right) + b_{14}\ln\left(H\right)\ln\left(\overline{S_T}\right) + b_{15}\ln\left(H\right)\ln\left(\mathrm{Re}\right) + b_{16}\ln\left(H\right)T + \\ & b_{17}\ln\left(H\right)\ln\left(L_c\right) + b_{18}\ln\left(\overline{S_L}\right)\ln\left(\overline{S_T}\right) + b_{19}\ln\left(\overline{S_L}\right)\ln\left(\mathrm{Re}\right) + b_{20}\ln\left(\overline{S_L}\right)T + \\ & b_{21}\ln\left(\overline{S_L}\right)\ln\left(L_c\right) + b_{22}\ln\left(\overline{S_T}\right)\ln\left(\mathrm{Re}\right) + b_{23}\ln\left(\overline{S_T}\right)T + b_{24}\ln\left(\overline{S_T}\right)\ln\left(L_c\right) + \\ & b_{25}\ln\left(\mathrm{Re}\right)T + b_{26}\ln\left(\mathrm{Re}\right)\ln\left(L_c\right) + b_{27}T\ln\left(L_c\right)\end{aligned} \tag{27}$$

$$\begin{aligned}\ln(\overline{Nu}_{base}) = {} & \ln(c_0) + c_1 \ln(H) + c_2 \ln(H)^2 + c_3 \ln\left(\overline{S_L}\right) + c_4 \ln\left(\overline{S_L}\right)^2 + c_5 \ln\left(\overline{S_T}\right) + \\ & c_6 \ln\left(\overline{S_T}\right)^2 + c_7 \ln(\mathrm{Re}) + c_8 \ln(\mathrm{Re})^2 + c_9 T + c_{10} T^2 + c_{11} \ln(L_c) + c_{12} \ln(L_c)^2 + \\ & c_{13} \ln(H) \ln\left(\overline{S_L}\right) + c_{14} \ln(H) \ln\left(\overline{S_T}\right) + c_{15} \ln(H) \ln(\mathrm{Re}) + c_{16} \ln(H) T + \\ & c_{17} \ln(H) \ln(L_c) + c_{18} \ln\left(\overline{S_L}\right) \ln\left(\overline{S_T}\right) + c_{19} \ln\left(\overline{S_L}\right) \ln(\mathrm{Re}) + c_{20} \ln\left(\overline{S_L}\right) T + \\ & c_{21} \ln\left(\overline{S_L}\right) \ln(L_c) + c_{22} \ln\left(\overline{S_T}\right) \ln(\mathrm{Re}) + c_{23} \ln\left(\overline{S_T}\right) T + c_{24} \ln\left(\overline{S_T}\right) \ln(L_c) + \\ & c_{25} \ln(\mathrm{Re}) T + c_{26} \ln(\mathrm{Re}) \ln(L_c) + c_{27} T \ln(L_c)\end{aligned} \tag{28}$$

where $a_0, \rightleftharpoons, a_{27}$, $b_0, \rightleftharpoons, b_{27}$, and $c_0, \rightleftharpoons, c_{27}$ are regression constants, and L_c is the dimensionless clearance between diagonal pins, which is defined as follows:

$$L_c = \sqrt{\frac{1}{4}\overline{S_T}^2 + \overline{S_L}^2} - 1 \tag{29}$$

The dimensionless parameter L_c quantified how closely or loosely packed a bank was, as can be seen in Figure 21.

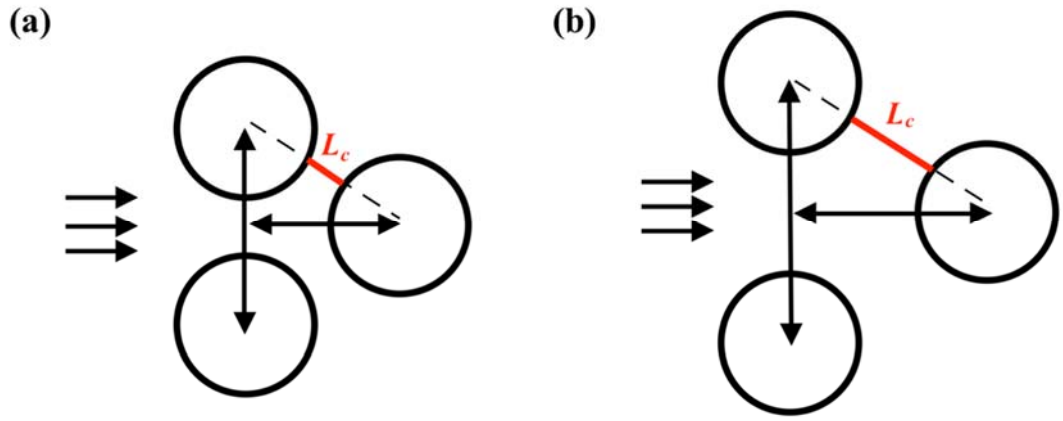


Figure 21: Closely-packed (a) and loosely-packed (b) pin arrangements with the same pitch ratio $\overline{S_T}/\overline{S_L}$.

Regression constants are listed for cylinders ($T=0$) and tapered pin fins ($0.25 \leq T \leq 1$) in Table 6. For small tapers ($0 < T < 0.25$), f, $\overline{Nu}_{pins}$, and $\overline{Nu}_{base}$ can be found via interpolation.

Table 6: Regression constants, a_i, b_i, and c_i, for f, $\overline{Nu}_{pins}$, and $\overline{Nu}_{base}$, respectively.

	a_i		b_i		c_i	
i	***T=0***	**$0.25 \leq T \leq 1$**	***T=0***	**$0.25 \leq T \leq 1$**	***T=0***	**$0.25 \leq T \leq 1$**
0	17.39064	9.3258	1.684038	2.164717	2.421935	1.300663
1	-0.632251	-2.093251	0.134146	-0.87197	-1.152584	0.232061
2	0.159105	0.480194	0.00451	0.162306	0.134273	-0.003877
3	-7.822854	-1.111391	1.302641	-0.09216	-1.31056	-1.363688
4	-1.919748	0.224504	0.889068	-0.01297	0.943943	0.153207
5	-14.05634	-0.03282	-2.280561	-0.31832	-1.548246	-0.362006
6	3.653954	0.572987	1.608669	-0.02787	1.019385	0.060085
7	-1.555485	-4.123046	-0.204223	-0.70577	-0.318876	0.798916
8	0.082262	0.765521	0.05236	-0.07925	0.058536	-0.161237
9	-	-1.430124	-	0.014031	-	-0.057312
10	-	0.080744	-	0.037096	-	0.05596
11	5.619648	0	-0.902667	0	0.727845	0
12	1.138258	0	-0.038057	0	0.08058	0
13	-0.458414	-0.272111	-0.050096	-0.02749	-0.118773	-0.067914
14	-0.529958	0.940883	-0.095203	0.211044	-0.047185	0.057924
15	0.102565	1.370578	-0.033263	0.543092	0.137017	0.103148
16	-	-0.123574	-	0.031514	-	-0.075866
17	0.048	0	0.015039	0	-0.065966	0
18	6.370566	-0.230633	-0.314037	0.048669	0.990635	-0.141062
19	-0.114072	-0.314208	0.017649	0.041861	0.134865	-0.016753
20	-	0.145824	-	-0.00423	-	0.175095
21	0.628818	0	-0.447749	0	-0.662223	0
22	0.299524	0.32471	0.105994	0.143373	-0.0045	0.534613
23	-	-0.174522	-	-0.00988	-	-0.069292
24	-2.403834	0	-0.765263	0	-0.927512	0
25	-	0.148388	-	0.004447	-	-0.198935
26	0.00855	0	0.036281	0	-0.110431	0
27	-	0	-	0	-	0

The closeness of fit of each correlation was assessed by comparing correlation predictions with corresponding simulation data. Correlation predictions were obtained by

evaluating the correlations with every combination of input parameters that was simulated. Figures Figure 22, Figure 23, and Figure 24 shows correlation predictions plotted against corresponding simulation data for $\overline{Nu}_{pins}$, $\overline{Nu}_{base}$, and f, respectively. Each plot includes data points for all 8,250 simulations, a line of perfect correlation (solid blue line) and lines indicating correlation variances of +10%, -10%, +20%, and -20% from simulation data (red dashed lines).

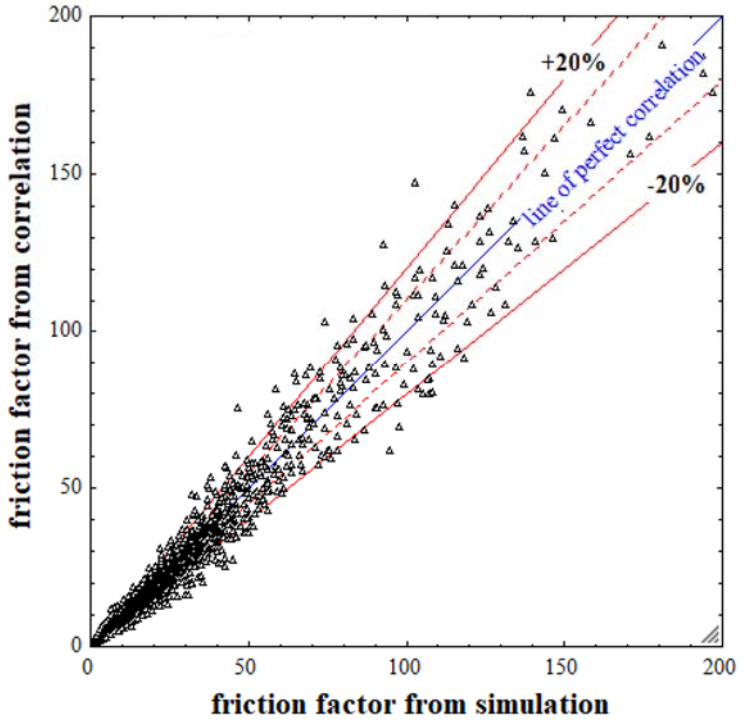


Figure 22: Correlation vs simulation friction factor.

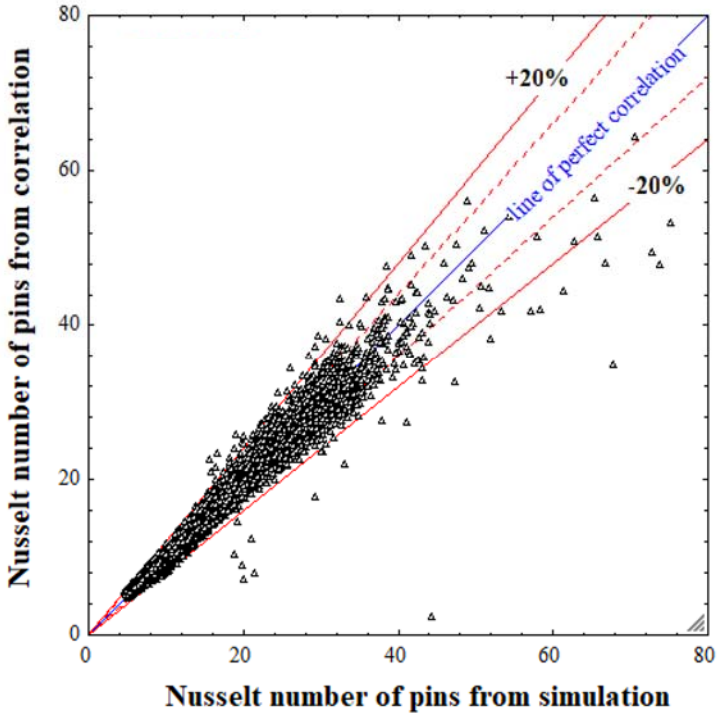


Figure 23: Correlation vs simulation Nusselt number of pins.

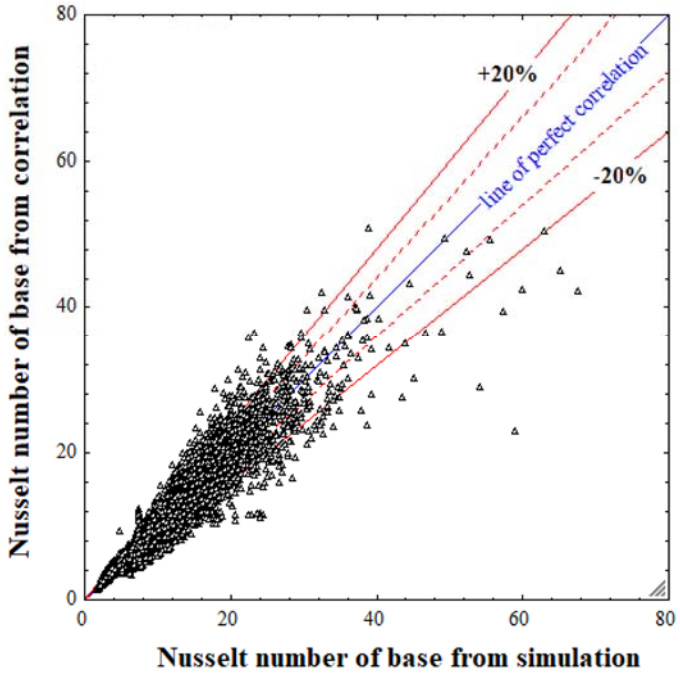


Figure 24: Correlation vs simulation Nusselt number of base.

The set of correlations predicted simulation data as follows: 92.4% of *f* correlation values, 99.1% of $\overline{Nu}_{pins}$ correlation values, and 87.5% of $\overline{Nu}_{base}$ correlation values were within 20% of simulation data.

The set of correlations for are provided in the following EES code, which requires $\mathrm{Re}, \overline{S_L}, \overline{S_T}, H$ and T as inputs and provides $\overline{Nu}_{pins}$, $\overline{Nu}_{base}$, and *f* as outputs:

```
Procedure tapered_pin_fin_corr(H, SL, ST, Re, T: f_corr, Nu_pins_corr, Nu_base_corr)
    L_crit = sqrt(((ST/2)^2)+(SL^2))-1

// friction factor
    f_tpr = exp(9.32579978E+00-2.09325115E+00*T+4.80194322E-01*T^2-
    1.11139091E+00*ln(H)+2.24503554E-01*ln(H)^2-3.28201554E-02*ln(SL)+5.72986919E-
    01*ln(SL)^2-4.12304618E+00*ln(ST)+7.65520714E-01*ln(ST)^2-
    1.43012397E+00*ln(Re)+8.07442247E-02*ln(Re)^2-2.72111265E-01*T*ln(H)+9.40883043E-
    01*T*ln(SL)+1.37057831E+00*T*ln(ST)-1.23574347E-01*T*ln(Re)-2.30633121E-
    01*ln(H)*ln(SL)-3.14208273E-01*ln(H)*ln(ST)+1.45824356E-01*ln(H)*ln(Re)+3.24710231E-
    01*ln(SL)*ln(ST)-1.74522254E-01*ln(SL)*ln(Re)+1.48387963E-01*ln(ST)*ln(Re))

    f_cyl = exp(1.73906375E+01-6.32251129E-01*ln(H)+1.59104629E-01*ln(H)^2-
    7.82285379E+00*ln(SL)-1.91974774E+00*ln(SL)^2-
    1.40563448E+01*ln(ST)+3.65395396E+00*ln(ST)^2-1.55548467E+00*ln(Re)+8.22615071E-
    02*ln(Re)^2+5.61964785E+00*ln(L_crit)+1.13825823E+00*ln(L_crit)^2-4.58414096E-
    01*ln(H)*ln(SL)-5.29957681E-01*ln(H)*ln(ST)+1.02564539E-01*ln(H)*ln(Re)+4.79995040E-
    02*ln(H)*ln(L_crit)+6.37056637E+00*ln(SL)*ln(ST)-1.14071640E-
    01*ln(SL)*ln(Re)+6.28817763E-01*ln(SL)*ln(L_crit)+2.99523741E-01*ln(ST)*ln(Re)-
    2.40383403E+00*ln(ST)*ln(L_crit)+8.54959320E-03*ln(Re)*ln(L_crit))

    If ( T = 0 ) Then f_corr = f_cyl
    If ( T > 0.2499 ) Then f_corr = f_tpr
    If ( T > 0 ) and ( T < 0.25 ) Then f_corr = 0.5 * ( ( ( T / 0.25 ) * f_cyl ) + ( ( 0.25 - T/ 0.25 ) * f_tpr
    ) )

// Nusselt number of pins
    Nu_pins_cyl = exp(1.68403808E+00+1.34145862E-01*ln(H)+4.50993678E-
    03*ln(H)^2+1.30264142E+00*ln(SL)+8.89067531E-01*ln(SL)^2-
    2.28056077E+00*ln(ST)+1.60866925E+00*ln(ST)^2-2.04223443E-01*ln(Re)+5.23596060E-
    02*ln(Re)^2-9.02667424E-01*ln(L_crit)-3.80570495E-02*ln(L_crit)^2-5.00959455E-
    02*ln(H)*ln(SL)-9.52034662E-02*ln(H)*ln(ST)-3.32628113E-02*ln(H)*ln(Re)+1.50386978E-
    02*ln(H)*ln(L_crit)-3.14036653E-01*ln(SL)*ln(ST)+1.76494049E-02*ln(SL)*ln(Re)-
    4.47748709E-01*ln(SL)*ln(L_crit)+1.05993790E-01*ln(ST)*ln(Re)-7.65263114E-
    01*ln(ST)*ln(L_crit)+3.62807086E-02*ln(Re)*ln(L_crit))

    Nu_pins_tpr=exp(2.16471688E+00-8.71970437E-01*T+1.62305691E-01*T^2-9.21590372E-
    02*ln(H)-1.29714319E-02*ln(H)^2-3.18316315E-01*ln(SL)-2.78676713E-02*ln(SL)^2-
    7.05773586E-01*ln(ST)-7.92511729E-02*ln(ST)^2+1.40308155E-02*ln(Re)+3.70961628E-
    02*ln(Re)^2-2.74896287E-02*T*ln(H)+2.11044413E-01*T*ln(SL)+5.43092149E-
    01*T*ln(ST)+3.15142709E-02*T*ln(Re)+4.86693130E-02*ln(H)*ln(SL)+4.18605728E-
```

```
    02*ln(H)*ln(ST)-4.23054548E-03*ln(H)*ln(Re)+1.43373010E-01*ln(SL)*ln(ST)-9.88051000E-
    03*ln(SL)*ln(Re)+4.44731095E-03*ln(ST)*ln(Re))

    If ( T = 0 ) Then nu_pins_corr = Nu_pins_cyl
    If ( T > 0.2499 ) Then nu_pins_corr = Nu_pins_tpr
    If ( T > 0 ) and ( T < 0.25 ) Then nu_pins_corr = 0.5 * ( ( ( T / 0.25 ) * Nu_pins_cyl ) + ( ( 0.25 -
    T/ 0.25 ) * Nu_pins_tpr ) )

// Nusselt number of base
    Nu_base_tpr = exp(1.30066268E+00+2.32060955E-01*T-3.87685114E-03*T^2-
    1.36368794E+00*ln(H)+1.53207060E-01*ln(H)^2-3.62005955E-01*ln(SL)+6.00853819E-
    02*ln(SL)^2+7.98915897E-01*ln(ST)-1.61237154E-01*ln(ST)^2-5.73118482E-
    02*ln(Re)+5.59602196E-02*ln(Re)^2-6.79142438E-02*T*ln(H)+5.79242307E-
    02*T*ln(SL)+1.03148201E-01*T*ln(ST)-7.58661113E-02*T*ln(Re)-1.41061763E-
    01*ln(H)*ln(SL)-1.67531891E-02*ln(H)*ln(ST)+1.75095046E-01*ln(H)*ln(Re)+5.34612726E-
    01*ln(SL)*ln(ST)-6.92920624E-02*ln(SL)*ln(Re)-1.98935123E-01*ln(ST)*ln(Re))

    Nu_base_cyl = exp(2.42193500E+00-1.15258382E+00*ln(H)+1.34272851E-01*ln(H)^2-
    1.31056010E+00*ln(SL)+9.43942878E-01*ln(SL)^2-
    1.54824594E+00*ln(ST)+1.01938472E+00*ln(ST)^2-3.18876170E-01*ln(Re)+5.85358230E-
    02*ln(Re)^2+7.27845432E-01*ln(L_crit)+8.05798967E-02*ln(L_crit)^2-1.18772867E-
    01*ln(H)*ln(SL)-4.71845360E-02*ln(H)*ln(ST)+1.37016683E-01*ln(H)*ln(Re)-6.59656321E-
    02*ln(H)*ln(L_crit)+9.90635191E-01*ln(SL)*ln(ST)+1.34865061E-01*ln(SL)*ln(Re)-
    6.62222663E-01*ln(SL)*ln(L_crit)-4.50010649E-03*ln(ST)*ln(Re)-9.27511628E-
    01*ln(ST)*ln(L_crit)-1.10431252E-01*ln(Re)*ln(L_crit))

    If ( T = 0 ) Then nu_base_corr = Nu_base_cyl
    If ( T > 0.2499 ) Then nu_base_corr = Nu_base_tpr
    If ( T > 0 ) and ( T < 0.25 ) Then nu_base_corr = 0.5 * ( ( ( T / 0.25 ) * Nu_base_cyl ) + ( ( 0.25 -
    T/ 0.25 ) * Nu_base_tpr ) )
End
```

4.5. Comparing Correlation and Experimental Results

To verify the physical accuracy of the correlations, the correlations were implemented into a heat exchanger optimization model and the results were compared to test data. The heat transfer rate and the pressure drop from the model and test are plotted against air velocity in Figure 25 and Figure 26, respectively.

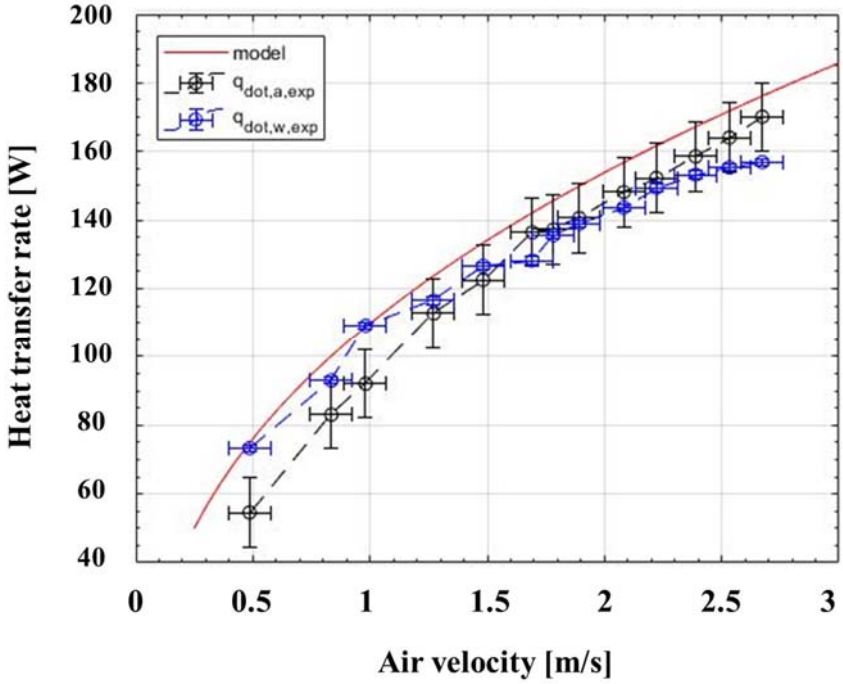


Figure 25: Heat transfer rate vs air velocity from model and experimental data [8].

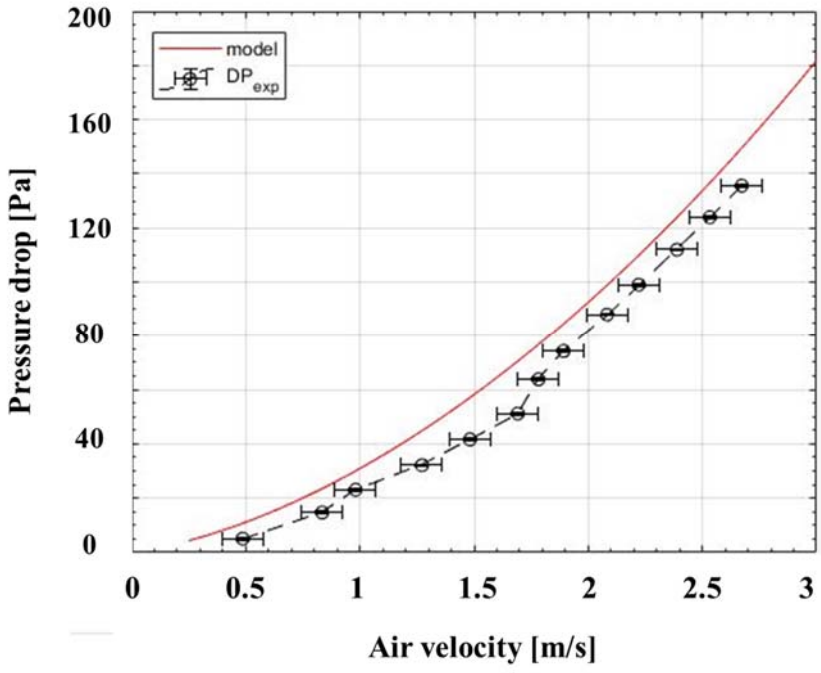


Figure 26: Pressure drop vs air velocity from model and experimental data [8].

Figure 25 and Figure 26 show good agreement between the model and experimental results. In fact, the model was within 10% of experimental results for both heat transfer rate and pressure drop for all tested heat exchangers.

4.6. Performance of Tapered Pin-Fins in Heat Exchanger

The axially-tapered pin-fin correlations developed in this work were implemented into an optimization model for a 3D-printed geometry and showed improved performance compared to airfoils and airfoils with subfins, as shown in Figure 27 [8].

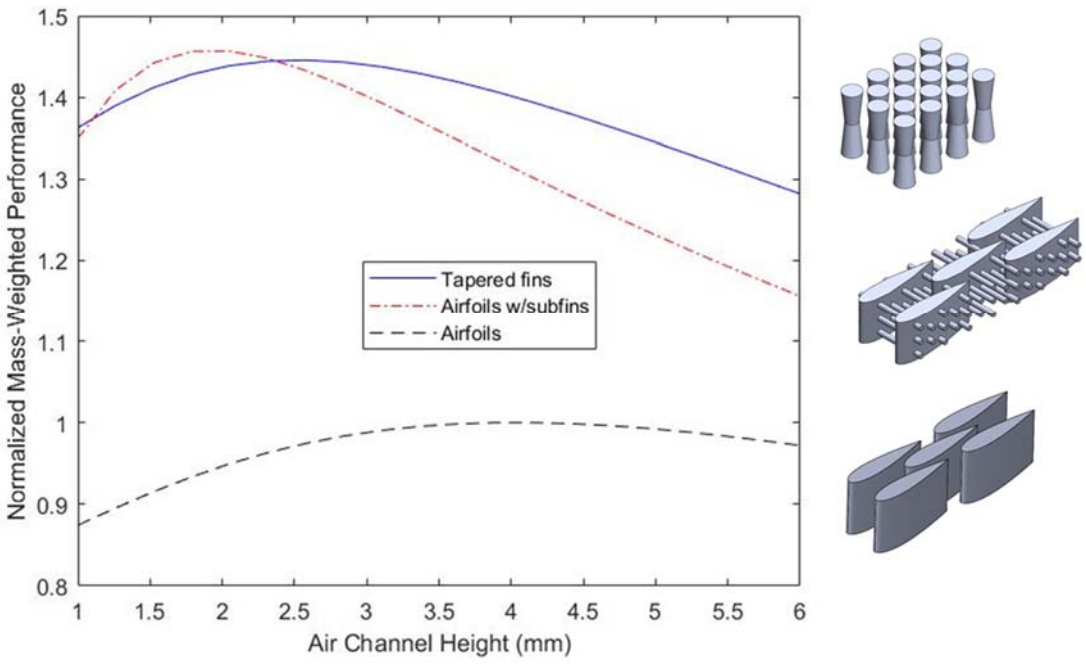


Figure 27: Mass-weighted heat exchanger performance of axially-tapered pins-fins, airfoils, and airfoils with sub-fins [8].

5. IMPLEMENTATION OF HIGH-THROUGHPUT COMPUTING (HTC)

To provide a reasonable turnaround on the large parametric study in this work, High Throughput Computing (HTC) was utilized. HTC was enabled by the University of Wisconsin-Madison's Center for High Throughput Computing (CHTC), which uses HTCondor (a task managing software developed at UW-Madison) to distribute jobs from users to its thousands of multi-core computers. It would've taken nearly 2 years to serially run all 8,250 simulations, since the average completion time for simulations in this work was approximately 2 hours. However, by using HTC to run hundreds of simulations simultaneously, it only took 2 weeks to run the full set of simulations.

The remainder of this section describes the process used by this work to implement HTC for large parametric analyses. Section 5.1 provides an overview then Sections 5.2-5.5 provide full scripts and more detailed explanations.

5.1. Overview of HTC Workflow

The process for conducting a parametric analysis using HTC involved preparation (steps 1-5), testing/debugging (steps 6-8), submission (steps 9-10), and data compilation (steps 11-12) and it was executed as follows:

1. Transfer meshes to CHTC submit server

```
scp -r meshes/ cleeds@submit-5.chtc.wisc.edu:simulations/
```

2. Log on to CHTC submit server

```
ssh cleeds@submit-5.chtc.wisc.edu
```

3. Make (or transfer) list of simulated values for each input parameter (**`hh_list.txt`**, **`SL_list.txt`**, **`hST_list.txt`**, **`Rmin_list.txt`**, and **`Re_list.txt`**)
4. Make simulation directories to store/organize output files and create list of arguments **`arglist.txt`** for each job to be submitted to the server

   ```
   bash make_directories_and_arglist.sh
   ```

5. Make (or transfer) journal file **`jouneric.jou`**, submit file **`subneric.sub`**, and executable file **`shneric.sh`**
6. Run test job(s)

   ```
   condor_submit subneric.sub
   ```

7. Debug any issues
8. Read memory usage from log file **`*.log`** and adjust `request_memory` in submit file **`subneric.sub`**

   ```
   nano *.log
   nano subneric.sub
   ```

9. Submit all remaining jobs

   ```
   condor_submit subneric.sub
   ```

10. Check on progress of submitted jobs

    ```
    condor_q    // reports the number of idle, running, held, and completed jobs for each batch
    bash run_data_diagnostic.sh
    ```

11. After jobs are finished running, parse s data from output files into single text file

    ```
    python parse_output_data.py
    ```

12. Copy simulation data text file from CHTC submit node to CAE I: drive

    ```
    scp data_date-time.txt cleeds@best-tu5.cae.wisc.edu:
    ```

The process was executed using shell commands within the command-line interface. The process was also automated using several scripts, including the HTCondor submit file (**`subneric.sub`**), ANSYS Fluent journal file (**`jouneric.jou`**), Python code (**`parse_output_data.py`**), and three shell scripts (**`make_directories_and_arglist.sh`**, **`shneric.sh`**, and **`run_data_diagnostic.sh`**). These scripts are shown in sections 5.2-5.5. The HTC process is depicted schematically in Figure 28.

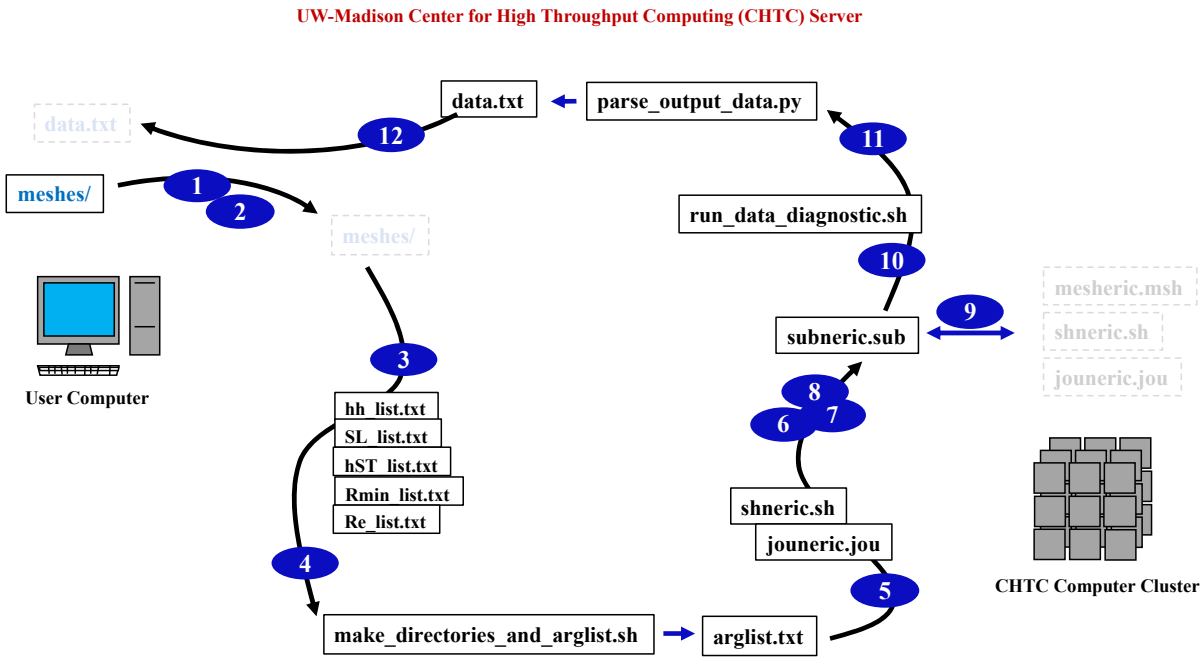


Figure 28: Schematic of HTC Workflow.

5.2. Preparation of Directories and Argument Lists

Before submitting a batch of jobs, input files and directories for storing output files were prepared. These input files included three HTC files (i.e., the submit file, **`subneric.sub`**, the

executable file, **shneric.sh**, and list of arguments for each job, **arglist.txt**) and ANSYS Fluent input files (i.e., meshes for all jobs and the journal file, **jouneric.jou**). This section focuses on the creation of **arglist.txt** and directories. The directory path for each job can generally be expressed by:

hh/ hh_SL_hST/ hh_SL_hST_Rmin/ hh_SL_hST_Rmin_Re

The five parameters have the following defintions: **hh** is the height of the computational domain, **SL** is the longitudinal pitch, **hST** is half the transverse pitch, **Rmin** is the minimum radius, and **Re** is the Reynolds numbers. The **arglist.txt** file contains a line for each job that can similarly be generally expressed:

hh SL hST Rmin Re

The simulated values for each parameter were listed in broken lines in **hh_list.txt**, **SL_list.txt**, **hST_list.txt**, **Rmin_list.txt**, and **Re_list.txt**, as shown:

hh_list.txt	SL_list.txt	hST_list.txt	Rmin_list.txt	Re_list.txt
0.5	1.25	1.25	1	30
1.25	1.5	1.5	0.75	44.292
2	1.75	1.75	0.5	65.394
4	2.25	2	0.25	96.549
6	3	2.25	0.0005	142.546
	4	2.5		210.458
				310.723
				458.757
				677.316
				1000

By looping through each parameter in nested for loops, the process for creating directories and **arglist.txt** could be automated. The **make_directories_and_arglist.sh** shell script employs this strategy, as shown:

```
#!/bin/bash
# Usage: bash make_directories_and_arglist.sh
```

```
      # exit if arglist.txt already exists
      if [ -f arglist.txt ]; then
            echo "ARGLIST ALREADY EXISTS"
            exit 0
      fi

      # loop through simulation parameters
      # make directories if they don't already exist

      # loop through heights in hh_list.txt
      while read h; do

            # make hh/ directory if it doesn't exist
            if [ ! -d $h/ ]; then
                  mkdir $h/
            fi

            # loop through longitudinal pitches in SL_list.txt
            while read l; do
            # loop through 1/2 transverse pitches in hST_list.txt
            while read s; do

                  # check pitch combination
                  # if ( SL >= 1.75 ) or
                  #    ( SL >= 1.5 AND hST >= 1.5) or
                  #    ( SL >= 1.25 AND hST >= 1.75)
                  if (( (( $(echo ""$l" > 1.74" | bc -l) )) || \
                        (( $(echo ""$l" > 1.49" | bc -l) &&
                              $(echo ""$s" > 1.49" | bc -l) )) || \
                        (( $(echo ""$l" > 1.24" | bc -l) &&
                              $(echo ""$s" > 1.74" | bc -l) )) )
                  then
                        # make hh/hh_SL_hST/ directory if it doesn't
                        exist
                        if [ ! -d $h/"$h"_"$l"_"$s"/ ]; then
                              mkdir $h/"$h"_"$l"_"$s"/
                        fi

                        # loop through minimum radii in Rmin_list.txt
                        while read p; do

                              # make hh/hh_SL_hST/hh_SL_hST_Rmin/
                              directory if it doesn't exist
                                    if [ ! -d
                  $h/"$h"_"$l"_"$s"/"$h"_"$l"_"$s"_"$p"/ ]; then
                                    mkdir
$h/"$h"_"$l"_"$s"/"$h"_"$l"_"$s"_"$p"/
                                    fi

                                    # loop through Reynolds numbers in Re_list.txt
                                    while read r; do

                                          # make hh/hh_SL_hST/hh_SL_hST_Rmin/
hh_SL_hST_Rmin_Re/ directory if it doesn't exist
```

```
                                    if [ ! -d
$h/"$h"_"$l"_"$s"/"$h"_"$l"_"$s"_"$p"/"$h"_"$l"_"$s"_"$p"_"$r"/ ]; then
                                        mkdir
$h/"$h"_"$l"_"$s"/"$h"_"$l"_"$s"_"$p"/"$h"_"$l"_"$s"_"$p"_"$r"/
                                    fi

                                    # write current simulation's parameters
in arglist.txt
                                    echo -n "$h $l $s $p $r" >> arglist.txt
                                    echo "" >> arglist.txt
                                done <Re_list.txt
                            done <Rmin_list.txt
                        fi
                done <hST_list.txt
                done <SL_list.txt
done <hh_list.txt
```

5.3. Job Submission and Status

After preparing input files and directories for storing output files, a test job and, eventually, a batch of jobs can be submitted. The basic workflow for submitting jobs on the CHTC server is depicted in Figure 29. The submit file, **`subneric.sub`**, communicates with the CHTC submit node to assign jobs and allocate memory on specific computers, and also transfer job-specific input files and parameters. Finally, **`subneric.sub`** determines where output files for a specific job will be sent.

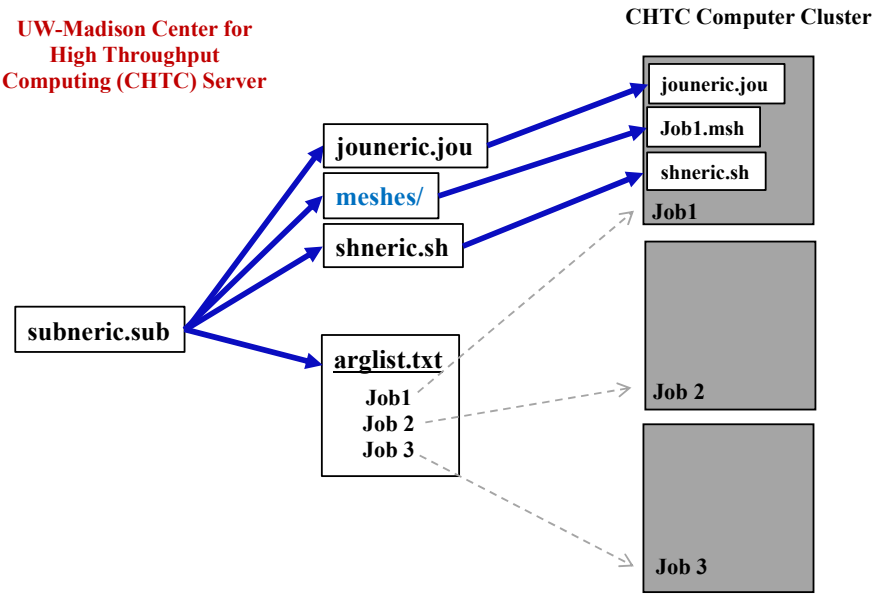


Figure 29: CHTC server job submission workflow.

The submit file, **`subneric.sub`**, utilizes CHTCondor scripting to communicate with the CHTC submit node, as shown:

```
executable = shneric.sh
#
# set directory for job
InitialDir =
$(hh)/$(hh)_$(SL)_$(hST)/$(hh)_$(SL)_$(hST)_$(Rmin)/$(hh)_$(SL)_$(hST)_
$(Rmin)_$(Re)/
# specify input files to transfer for job
transfer_input_files =
../../../../jouneric.jou,../../../../meshes/$(hh)_$(SL)_$(hST)_$(Rmin).
msh
# define job arguments
arguments = $(hh) $(SL) $(hST) $(Rmin) $(Re)
#
# job info files
log = $(hh)_$(SL)_$(hST)_$(Rmin)_$(Re)_$(Cluster)_$(Process).log
output = $(hh)_$(SL)_$(hST)_$(Rmin)_$(Re)_$(Cluster)_$(Process).out
error = $(hh)_$(SL)_$(hST)_$(Rmin)_$(Re)_$(Cluster)_$(Process).err
#
# transfer output files from job to specified directory
should_transfer_files = YES
#
# set request memory, disk, and cpus
# !!! adjust request memory for size of jobs being submitted
```

```
request_memory = 1500MB
request_disk = 50MB
request_cpus = 1
#
# require servers with the ANSYS module in CHTC Gluster share:
Requirements = (Target.HasGluster == true) && (OpSysMajorVer == 6)
getenv = true
# read job arguments line by line from arglist.txt
queue hh,SL,hST,Rmin,Re from arglist.txt
# END
```

After the job-specific input files are transferred to a specific computer, the executable file, , controls the job. Shell scripting is used by **`shneric.sh`** to substitute job-specific input parameters into the journal file, **`jouneric.jou`** (provided in Section 3.5.3), then run an ANSYS Fluent simulation, as shown:

```
#!/bin/bash
# shneric.sh input arguments:
# $1    hh (height)
# $2    SL (longitudinal pitch)
# $3    hST (1/2 transverse pitch)
# $4    Rmin (minimum radius)
# $5    Re (Reynolds number)

# activate modules inside an HTC Job
. /etc/profile.d/modules.sh

# calculate inlet velocity based on Reynolds number
# u = ( mu * Re ) / ( rho * Db )
u=$(echo "scale=10; (0.000017894*$5)/(1.225*0.002)" | bc)

# substitute current simulation's parameters in journal file
sed "s/mesheric/$1_$2_$3_$4/g;\
  s/generic/$1_$2_$3_$4_$5/g;\
  s/u_inf/$u/g;" jouneric.jou >> $1_$2_$3_$4_$5.jou

# load ANSYS modules and run fluent:
module load ansys-17.2
module load mpi/gcc/openmpi-1.6.4
fluent 3ddp -g -t1 -i $1_$2_$3_$4_$5.jou

# END
```

The output files for a successful job include simulation data from ANSYS Fluent and three CHTCondor output files (i.e., the log, output, and error files), which contain information about the job. The log file, **`*.log`**, summarizes the memory usage, the output file, **`*.out`**, is a

transcript of all shell and ANSYS Fluent commands and responses, and the error file, **`*.err`**, provides information about any job errors. Before submitting a batch with all jobs, a test job should be submitted to debug potential issues and allocate the correct amount of memory.

5.3.1 Checking Job Status

To provide additional information about which jobs had finished successfully and which jobs were either unsuccessful or hadn't completed, the **`run_data_diagnostic.sh`** shell script was developed:

```
#!/bin/bash
# usage: bash run_data_diagnostic.sh

# delete jobsToReSubmit.txt and successfulJobs.txt if they already
exist
if [ -f jobsToReSubmit.txt ]; then
      rm jobsToReSubmit.txt
fi
if [ -f successfulJobs.txt ]; then
    rm successfulJobs.txt
fi

# loop through simulation parameters
# loop through heights in hh_list.txt
while read h; do
      # loop through SL and hST combinations in pitchList.txt
      while read p; do
            # loop through minimum radii in Rmin_list.txt
            while read t; do
                  a=0
                  # loop through Reynolds numbers in Re_list.txt
                  while read r; do
                        # check if simulation output file exists and
has a size greater than 0
                        if [ -s
""$h"/"$h"_"$p"/"$h"_"$p"_"$t"/"$h"_"$p"_"$t"_"$r"/"$h"_"$p"_"$t"_"$r".
cas" ]; then
                              echo "$h $p $t $r" >> successfulJobs.txt
                        else
                              echo "$h $p $t $r" >> jobsToReSubmit.txt
                              # clear output files from unsuccessful
job
                              if [ -f
""$h"/"$h"_"$p"/"$h"_"$p"_"$t"/"$h"_"$p"_"$t"_"$r"/*" ]; then
```

```
                                    rm
""$h"/"$h"_"$p"/"$h"_"$p"_"$t"/"$h"_"$p"_"$t"_"$r"/*"
                                fi
                                a=$(($a+1))
                        fi
                    done <Re_list.txt

                    # if no jobs were successful for a given geometry,
add it to meshesToCheck.txt
                    if [ "$a" -eq "10" ]; then
                        echo "$h $p $t" >> meshesToCheck.txt
                        a=0
                    fi
                done <Rmin_list.txt
            done <pitchList.txt
done <hh_list.txt

# count the number of unsuccessful and successful jobs
numR=$(wc -l < "jobsToReSubmit.txt")
numS=$(wc -l < "successfulJobs.txt")

# count the total number of jobs
numTot=$(echo "$numR + $numS" | bc)

# print the number of successful, unsuccessful, and total jobs
printf "OUT OF %s JOBS:
      %s    JOBS WERE SUCCESSFUL
      %s    JOBS NOT COMPLETED/SUCCESSFUL
" $numTot $numS $numR

# END
```

The **run_data_diagnostic.sh** script prints the number of successful, unsuccessful, and total jobs to the command-line interface and creates up to three text files: **successfulJobs.txt**, **jobsToReSubmit.txt**, and **meshesToCheck.txt**. A mesh is added to the **meshesToCheck.txt** if all jobs that were ran using it were unsuccessful.

5.4. Data Compilation/Extraction

Finally, data was extracted from the directories storing job output files using the **parse_output_data.py** Python script. By looping through **parse_output_data.py**, can find these data and parse them into a single text file, as shown:

```
import os
import datetime

# record current date
now = datetime.datetime.now()
stamp = now.strftime("%d%h%y-%Hh%Mm")

# define surfaces at which output data is collected
surf = ['pins', 'bot_cd', 'in', 'start-src', 'end-src']

# create lookup file to store output data and set up column headers
with open ("data_" + stamp + ".txt", "w+") as df:
   df.write("-23 -23\n")
   df.write("A hh [mm]\n")
   df.write("A sl [mm]\n")
   df.write("A hst [mm]\n")
   df.write("A Rmin [mm]\n")
   df.write("A Re [mm]\n")
   df.write("A A_pins [m^2]\n")
   df.write("A A_base [m^2]\n")
   df.write("A p_start [Pa]\n")
   df.write("A p_end [Pa]\n")
   df.write("A tm_end_pins [K]\n")
   df.write("A tm_end_base [K]\n")
   df.write("A u_inf [m/s]\n")
   df.write("A m_dot [kg/s]\n")
   df.write("A DELTAp [Pa]\n")
   df.write("A h_bar_pins [W/m^2-K]\n")
   df.write("A h_bar_base [W/m^2-K]\n")
   df.write("A f [-]\n")
   df.write("A Nu_pins [-]\n")
   df.write("A Nu_base [-]\n")
   df.write("A H [-]\n")
   df.write("A SL [-]\n")
   df.write("A ST [-]\n")
   df.write("A T [-]\n")
df.close()

# parse input arguments line by line into lists
with open ("hh_list.txt", "r") as f:
   hh = f.read().splitlines()
with open ("pitchList.txt", "r") as f:
   pitch = f.read().splitlines()
with open ("Rmin_list.txt", "r") as f:
   Rmin = f.read().splitlines()
with open ("Re_list.txt", "r") as f:
   Re = f.read().splitlines()

# loop through each combination of input arguments
ih = 0
while ih < len(hh):
   ip = 0
   while ip < len(pitch):
          it = 0
          while it < len(Rmin):
```

```
                ir = 0
                while ir < len(Re):
                        # define directory according to input arguments
                        args = hh[ih] + '_' + pitch[ip] + '_' + Rmin[it] + '_' +
Re[ir]
                        geom = hh[ih] + '_' + pitch[ip] + '_' + Rmin[it]
                        dir = hh[ih] + '/' + hh[ih] + '_' + pitch[ip] + '/' + geom + '/'
+ args

                        SL, hST = pitch[ip].split("_")

                        # parse output data files into space-delimited lists
                        if os.path.isfile(dir + '/' + geom + '_area'):
                                with open (dir + '/' + geom + '_area', "r") as f:
                                        area_list = f.read().split()
                                f.close()
                        else:
                                area_list = [surf[0], "999", surf[1], "999"]

                        if os.path.isfile(dir + '/' + args + '_p'):
                                with open (dir + '/' + args + '_p', "r") as f:
                                        p_list = f.read().split()
                             f.close()
                        else:
                                p_list = [surf[3], "999", surf[4], "999"]

                        if os.path.isfile(dir + '/' + 'case1_' + args + '_tm'):
                                with open (dir + '/' + 'case1_' + args + '_tm', "r")
as f:
                                        case1_tm_list = f.read().split()
                             f.close()
                        else:
                                case1_tm_list = [surf[4], "999"]

                        if os.path.isfile(dir + '/' + 'case2_' + args + '_tm'):
                                with open (dir + '/' + 'case2_' + args + '_tm', "r")
as f:
                                case2_tm_list = f.read().split()
                        f.close()
                        else:
                                case2_tm_list = [surf[4], "999"]

                        if os.path.isfile(dir + '/' + args + '_uinf'):
                                with open (dir + '/' + args + '_uinf', "r") as f:
                                        uinf_list = f.read().split()
                                f.close()
                        else:
                                uinf_list = [surf[2], "999"]

                        if os.path.isfile(dir + '/' + args + '_mdot'):
                                with open (dir + '/' + args + '_mdot', "r") as f:
                                        mdot_list = f.read().split()
                                f.close()
                        else:
                                mdot_list = [surf[2], "999"]
```

```
                        with open ("data_" + stamp + ".txt", "a") as df:
                              df.write(hh[ih] + " ")
                              df.write(SL + " ")
                              df.write(hST + " ")
                              df.write(Rmin[it] + " ")
                              df.write(Re[ir] + " ")

                              if surf[0] in area_list:
                                    for idx, a in enumerate(area_list):
                                          if a == surf[0]:
                                                df.write(area_list[idx+1] + " ")
                              else:
                                    df.write("999 ")

                              if surf[1] in area_list:
                                    for idx, a in enumerate(area_list):
                                          if a == surf[1]:
                                                df.write(area_list[idx+1] + " ")
                              else:
                                    df.write("999 ")

                              if surf[3] in p_list:
                                    for idx, p in enumerate(p_list):
                                         if p == surf[3]:
                                                df.write(p_list[idx+1] + " ")
                              else:
                                    df.write("999 ")

                              if surf[4] in p_list:
                                    for idx, p in enumerate(p_list):
                                    if p == surf[4]:
                                          df.write(p_list[idx+1] + " ")
                              else:
                                    df.write("999 ")

                              if surf[4] in case1_tm_list:
                                    for idx, t in enumerate(case1_tm_list):
                                    if t == surf[4]:
                                          df.write(case1_tm_list[idx+1] + " ")

                              else:
                                    df.write("999 ")

                              if surf[4] in case2_tm_list:
                                    for idx, t in enumerate(case2_tm_list):
                                          if t == surf[4]:
                                                df.write(case2_tm_list[idx+1] +
    " ")
                              else:
                                    df.write("999 ")

                              if surf[2] in uinf_list:
                                    for idx, u in enumerate(uinf_list):
                                    if u == surf[2]:
```

```
                                    df.write(uinf_list[idx+1] + " ")
                        else:
                              df.write("999 ")

                        if surf[2] in mdot_list:
                              for idx, u in enumerate(mdot_list):
                              if u == surf[2]:
                                    df.write(mdot_list[idx+1] + " ")
                        else:
                              df.write("999 ")

                        # clear lists
                        area_list[:] = []
                        p_list[:] = []
                        case1_tm_list[:] = []
                        case2_tm_list[:] = []
                        uinf_list[:] = []
                        mdot_list[:] = []
                  df.close()

                  with open ("data_" + stamp + ".txt", "a") as df:
                        df.write("0 0 0 0 0 0 0 0 0 0\n")
                  df.close()

                  ir += 1
            it += 1
      ip += 1
  ih += 1
df.close()
```

6. CONCLUSIONS AND FUTURE WORK

The parametric analysis and correlations for axially-tapered pin-fin arrays in this work show promising results. Axially-tapered pin-fins can be implemented into 3D-printed heat exchangers for increased hydrodynamic and thermal performance [8]. The turn-around time for the large parametric analysis in this work was reduced by a factor of 1000 using High-Throughput Computing (HTC). However, to improve the correlations and HTC process in this work, future works might consider:

- **Reducing the impact of entry effects on correlations**. The correlations in this work were developed from simulations with only 4 rows of pins, which were shown to be impacted by entry and exit effects. This could be mitigated by simulating more than 4 rows of pins or by implementing periodic boundary conditions on the inlet and outlet (note: periodic boundary conditions cannot be implemented for 3D models in ANSYS Fluent, so another CFD software would be needed).
- **Investigating the thermal performance of the bases further.** Ideally, the correlation for Nusselt number of the base should better fit simulation data.
- **Automating mesh generation using HTC.** The geometry and mesh generation portion of the parametric analysis took longer than the full HTC process. If geometry and mesh generation were incorporated into the HTC process, this would greatly reduce the required user input and overall turn-around time of the parametric analysis.

REFERENCES

[1] S. B. Beale, "Tube Banks, Crossflow over," Thermopedia, 2011. [Online]. Available: http://www.thermopedia.com/content/1211/. [Accessed 2017].

[2] A. Žukauskas, "Heat transfer from tubes in crossflow," Advances in Heat Transfer, vol. 8, 1972.

[3] S. A. Klein. (2018). Engineering Equation Solver (10.381) [software]. F-Chart.

[4] F. P. Incopera et al., “External Flow,” in Fundamentals of Heat and Mass Transfer, 6th ed. New York: J. Wiley, 2002, ch. 7, sec. 4, pp. 423–443.

[5] S. W. Churchill et al., “A Correlating Equation for Forced Convection From Gases and Liquids to a Circular Cylinder in Crossflow,” Journal of Heat Transfer vol. 99, pp. 300-307, May 1977.

[6] SHARCNET, "FLUENT 6.3 User's Guide," 20 09 2006. [Online]. Available: https://www.sharcnet.ca/Software/Fluent6/html/ug/main_pre.htm. [Accessed 2018].

[7] G. F. Nellis and S. A. Klein, “Heat Exchangers,” in Heat Transfer. New York: Cambridge University Press, 2009, ch. 8, sec. 3, pp. 851-858.

[8] J. Boxleitner, “Additive Design and Manufacturing of a Composite Polymer Heat Exchanger,” to be presented at 17th International Refrigeration and Air Conditioning Conference at Purdue, West Lafayette, IN. 2018