Experimental and Numerical Study of Impingement Jet Heat Transfer

by

Andrew Urban Schroder

Bachelor of Science, University of Cincinnati, 2007

A thesis submitted to the

Faculty of the Graduate School of the

University of Cincinnati in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

College of Engineering

Committee Chair: Dr. Urmila Ghia

Thursday, May 5^{th}, 2011

Abstract

An experimental test facility has been designed, constructed, and commissioned for studying the convective heat transfer of an array of 55 impingement jets. Spatial variation in time averaged Nusselt number as well as spanwise time averaged Nusselt number are presented for jet Reynolds numbers of 4,000, 8,000, 12,000, and 15,000 for jet to target standoff distances of z/D=3, 4, and 5. For each of these configurations the exit flow configuration has also been varied to include both a single exit and double exit configuration. For each jet standoff distance and exit configuration, time and overall area averaged Nusselt number is presented as a function of jet Reynolds number. Animations of measured unsteady Nusselt number are presented for selected cases.

Numerical simulations have been conducted using the Fluent Computational Fluid Dynamics software package. The three dimensional, compressible, Navier Stokes equations are solved. Results for Nusselt number are presented for a grid dependency study of a steady, single impingement jet impacting a target surface at a standoff distance of z/D=3, at jet Reynolds numbers of 4,000 and 15,000. In the single jet grid dependency study flow is exhausted in all directions after impacting the target surface. Grids ranging from 1.2 million to 13.2 million grid points are evaluated.

Unsteady simulations were conducted of a single impingement jet at a jet Reynolds number of 4000, a jet to target standoff distance of z/D=3, flow exiting in two directions, and a pair of repeating boundaries in the other two directions. Two cases were studied. The first with a spacing between the two repeating boundaries of y/D=6 and the second with a spacing of y/D=3. For both cases, contour plots of time averaged, as well as animations of unsteady in plane velocity magnitude, normal component of vorticity, and Nusselt number are presented.

Unsteady simulations were also conducted of eleven impingement jets at a jet Reynolds number of 4,000, a jet to target standoff distance of z/D=3, and a double exit configuration. Two cases

were studied. The first with a spacing between the two repeating boundaries of y/D=4.03 and the second with a spacing of y/D=3. For both cases, contour plots of time averaged in plane velocity magnitude, normal component of vorticity, pressure, temperature, and Nusselt number are presented. Animations of contours of in plane velocity magnitude, normal component of vorticity, and Nusselt number are also presented.

Spanwise time average Nusselt number for both eleven jet numerical cases is compared to that of the central row of an experimental case with a double exit configuration. Comparisons are also made between spanwise average Nusselt number for the central jet of an experimental case, a single jet numerical, the single jet numerical cases with the repeating boundaries, as well as the central jet of the eleven jet numerical cases.

Acknowledgements

I would like to extend sincere gratitude towards Dr. Shichuan Ou of the Air Force Research Laboratory. His mentorship throughout the course of my graduate studies has been greatly appreciated. It has been a great pleasure to work with Dr. Ou over the last two years developing a completely new experimental test facility. The freedom, respect, and encouragement Dr. Ou has provided during the course of the design, assembly, and commissioning of the test rig has been truly instrumental in the successful development of the test rig.

I would also like to thank the Air Force Research Laboratory for providing lab space, funding for manufacturing and purchasing components for the test rig, computing resources for the numerical component of this study, as well as financial support for a portion of this project. Additionally, the friendly support staff at the Air Force Research Laboratory Department of Defense Supercomputing Resource Center was always available whenever I would run into any issues with their computing facility.

I would also like to thank the Dayton Area Graduate Studies Institute Student Fellowship Program for providing financial support for a portion of this project and for providing a tuition scholarship while completing my master's degree, as well as helping me to make my initial connection with Dr. Ou and the Air Force Research Laboratory.

Finally, I would like to thank all of my thesis committee members. I would like to thank Dr. Urmila Ghia of the University of Cincinnati for serving as my committee chair and for providing detailed technical and grammatical feedback on this document as well as my thesis defense. I would like to thank Dr. Milind Jog of the University of Cincinnati and Dr. Shichuan Ou for also serving as committee members and for providing critical feedback.

Contents

List of Figures

List of Tables

Nomenclature

Greek Symbols

α	thermal diffusivity, m^2/s
$\alpha_{film,natural}$	thermal diffusivity based upon film temperature, natural convection side, m^2/s
β	expansion coefficient, $1/K$
δ_L	turbulent boundary layer thickness at location L for flow over a flat plate, m
ϵ	imaged surface emissivity
γ	ratio of specific heat at constant pressure to the specific heat at constant volume
μ	absolute viscosity, $kg/(m * s)$
μ_0	fluid reference viscosity, $kg/(m * s)$
ν	kinematic viscosity, m^2/s
$\nu_{film,natural}$	kinematic viscosity based upon film temperature, natural convection side, m^2/s
π	ratio of the circumference of a circle to its diameter
ρ	volumetric resistivity, $\Omega * m$
ρ_{fluid}	fluid density, kg/m^3
ρ_{throat}	fluid density at the venturi nozzle throat, kg/m^3
σ	Stefan-Boltzmann constant, $W/(m^2 * K^4)$
$\tau_{w,L}$	local wall shear stress at location L, N/m^2

Roman Symbols

$\dot{m}$	total mass flow rate through all impingement jet nozzles, kg/s
a	speed of sound in a gas, m/s
A_s	surface area of the foil, m^2
A_{throat}	cross sectional area of the venturi nozzle throat, m^2

b	heated foil thickness, m
c_p	specific heat at constant pressure, $J/(kg * K)$
c_v	specific heat at constant volume, $J/(kg * K)$
$C_{f,L}$	skin friction coefficient of turbulent flow over a flat plate at location L
D	impingement jet nozzle diameter, m
D_{throat}	diameter of the venturi nozzle throat, m
g	combined gravitational and centrifugal acceleration on the earth's surface, m/s^2
H	height of the imageable region of the foil heater, H
$h(x, y, t)$	local impingement jet convective heat transfer coefficient, $W/(m^2 * K)$
$h_{natural}(x, y, t)$	estimated natural convection heat transfer coefficient, $W/(m^2 * K)$
I	current, *Amps*
k	thermal conductivity, $W/(m * K)$
$k(x, y, t)$	local thermal conductivity based on film temperature, $W/(m * K)$
$k_{natural}(x, y, t)$	local thermal conductivity based on film temperature, natural convection side, $W/(m * K)$
l	heated foil length, m
M	Mach number, the ratio of the speed of a fluid to the speed of sound of that fluid
N	total number of impingement jet nozzles
$Nu(x, y, t)$	local Nusselt number
Nu_{mean}	time and area averaged Nusselt number
$Nu_{natural}$	Nusselt number for natural convection
P	foil heater heating power, W
p_{throat}	static pressure at the venturi nozzle throat, N/m^2
p_{throat}	static pressure downstream of the venturi nozzle throat, N/m^2
p_T	total pressure of the fluid upstream of the venturi nozzle throat, N/m^2
q''_{foil}	foil heater heat flux, W/m^2
q''_{jets}	impingement jet heat flux, W/m^2
q''_{losses}	heat losses on the imaged side of the foil, W/m^2
R	heated foil resistance, Ω
R_{fluid}	specific ideal gas constant of a fluid, $J/(kg * K)$

Ra_y	Rayleigh number
Re_L	Reynolds number based upon the distance L from the leading edge of the plate
Re_{jet}	Reynolds number based on the jet nozzle diameter
S	Sutherland constant, K
T	static temperature, K
t	time, s
T^*	static temperature of a fluid that is accelerated or decelerated to the speed of sound, K
T_0	fluid reference temperature, K
$T_s(x, y, t)$	local foil surface temperature, K
T_T	total temperature, K
$T_{ambient}$	static temperature of the ambient air in the test cell, K
$T_{film,natural}(x, t)$	spanwise average film temperature, natural convection side, K
$T_{s,spanwise}(x, t)$	spanwise average surface temperature, K
T_{throat}	static temperature at the venturi nozzle throat, K
$T_{upstream}$	average fluid temperature measured in the pressure chamber, K
U_∞	free stream fluid velocity, m/s
$u_{\tau,L}$	friction velocity at location L, m/s
V	voltage drop across the foil heater, $Volts$
V_{throat}	fluid velocity at the venturi nozzle throat, m/s
w	heated foil width, m
x	horizontal dimension, m
y	vertical dimension, m
y_L^+	dimensionless distance at location L

Chapter 1

Introduction

1.1 Motivation

Gas turbine engine designs are increasingly pushing the structural and thermal limitations of today's materials. In a gas turbine engine, as in any heat engine, higher temperature and higher pressure cycles lead to higher efficiency and higher power density. Gas turbine engines typically use high thermal conductivity metals due to their cost, durability, strength and toughness. There is an increasing push for the use of Ceramic Matrix Composite (CMC) materials in gas turbine hot sections; however, CMCs also have temperature limitations. Regardless of the maximum service temperature of the material, heat transfer must be carefully controlled in order to keep parts from overheating.

Internal cooling of gas turbine hot section components, such as turbine blades, is commonly performed through the combination of impingement cooling and serpentine channels. This research is focused on impingement cooling. Impingement cooling has the advantage of a thin boundary layer due to the stagnation point flow when the jet core impacts the target surface. Additionally, mixing of the cool supply air and the hotter, spent air is reduced due to the separation of these two fluids by an impingement nozzle plate. Unfortunately, when large arrays of impingement jets are implemented inside a gas turbine blade, cross flow effects from spent jet air reduce the cooling effectiveness of downstream jets. The jet is bent, the strength of the jet is reduced, and the spent air engages in additional mixing prior to impacting the surface to be cooled.

The earliest techniques utilized arrays of thermocouples to study the spatially dependent heat

transfer coefficients on an impingement target surface [1, 2]. Later developments primarily utilized a transient liquid crystal thermography technique [3, 4, 5, 6]. The transient liquid crystal technique does not allow for time averaging of the convective heat transfer coefficient, but rather, assumes that the convective heat transfer coefficient is independent of time. Additionally, the transient liquid crystal technique requires complex liquid crystal color vs temperature calibration procedures, sensitive test procedures and custom data reduction software. These transient techniques typically utilize recirculation loops in order to heat the supply air to an appropriate temperature. Large transient liquid crystal facilities are oftentimes utilized in order to study convective heat transfer at high Reynolds numbers [7].

In order to overcome these complexities and limitations, a new type of test rig has been developed with Dr. Shichuan Ou of the Air Force Research Laboratory (AFRL) for studying the convective heat transfer coefficient of impingement jets in the presence of cross flow. This new, compact test rig utilizes a steady state Infrared Thermography technique which allows for the determination of unsteady, as well as time averaged convective heat transfer coefficient with respect to position on the impingement surface. Additionally, a computational study of similar geometry was initiated in an effort to gain more detailed, three dimensional insight into the flow physics involved with impingement cooling jets.

1.2 Overview of Impingement Jets and Impingement Jet Arrays

Impingement jets can be used as a heat transfer enhancement technique to either heat or cool the target surface. A schematic of a sectional view of a single impingement jet is presented in Figure 1.1. Although not to scale, Figure 1.1 illustrates several key features of impingement jet flow and impingement jet nozzles. Impingement jets are typically either slot jets or cylindrical jets, which in the sectional view presented in Figure 1.1 appear the same. Cylindrical impingement jet nozzles are typically mandated in gas turbine blade applications due to structural and manufacturing requirements of the gas turbine blade. The present work focuses exclusively on cylindrical impingement jet nozzles. Fluid in a higher pressure reservoir, typically referred to as the pressure chamber, is accelerated through the impingement nozzle, and then impacts the target surface at a high velocity. A free shear layer exists between the high velocity fluid, known as the jet core,

and the remaining air in the impingement region. Due to the free shear layer, a recirculation is induced in the surrounding fluid. After impacting the target surface, the impingement jet fluid then travels to an exit. Depending on the application, the fluid can exit in all directions or just one specific direction. As was just mentioned, impingement jets can be used to heat or cool the target surface. Due to the exclusive use of impingement jets for cooling applications in gas turbine blades, the present work focuses primarily on the use of impingement jets for cooling applications. Additionally, impingement jets can be multiphase, however, due to the use of only air in gas turbine blade cooling, only single phase impingement jet heat transfer is studied in the present work.

Figure 1.1: Sectional of a Singe Impingement Jet

Figure 1.2 illustrates a sectional view of the use of multiple impingement jets as an impingement jet array. Due the the sharp drop in convective heat transfer as the boundary layer thickens as the fluid travels away from the jet core, multiple impingement jets are typically needed in order to heat or cool a large surface area. In Figure 1.2, the exhaust fluid is given the opportunity to exit in two directions. The total mass flow rate through the cross section increases as the exhaust

fluid travels towards the exits because the exhausted fluid (cross flow) must travel around the core of each downstream impingement jet. As the exhaust fluid travels around these downstream jets, the performance of the downstream jets is reduced due to this cross flow interaction. Additional reduction in convective heat transfer is due to jet to jet interactions. Jet to jet interactions can be described as the interaction between two neighboring jets. As can be seen in Figure 1.1, the high velocity fluid turns after impacting the target surface and becomes a "wall jet" parallel to the target surface. With two nearby jets, the resulting two "wall jets", which are parallel to the target surface, interact with each other, and cause a second stagnation region. In this second stagnation region, the boundary layer thickens, and the heat transfer coefficient is reduced.

Figure 1.2: Sectional View of an Array of Impingement Jets

The purpose of the present study is to better characterize the effects of the cross flow and jet to jet interactions on jet performance in impingement jet arrays. Although the application of impingement cooling to gas turbine blades is of primary interest, the geometry studied was much simpler than typically found in a gas turbine blade. The impingement jet nozzles were arranged in a rectangular patter. The target surface was flat. The impingement jet nozzle plate was primarily rectangular in cross section. In an application found in an actual turbine blade, curved nozzle plates and target surfaces would be required, and unequal placement of impingement jet nozzles may be necessary due to the cooling needs of the turbine blade. In addition, no rotational affects were studied, which would be present in an actual, rotating turbine blade. Chapter 2 discusses the design of an experimental facility and the convective heat transfer characteristics measured for an

impingement jet array. Chapter 3 discusses numerical simulations of heat transfer characteristics of single impingement jets as well as heat transfer characteristics of arrays of multiple impingement jets.

Chapter 2

Experimental Study of Impingement Jet Array Heat Transfer

2.1 Test Rig Design

The design of the experimental test facility was initiated without any prior experience with impingement jet heat transfer. The test rig was developed along with Dr. Shichuan Ou of the Air Force Research Laboratory (AFRL), and was installed in Research Cell 21 in Building 18C at Wright Patterson Air Force Base. Key driving factors in the test rig design were the desire to utilize an infrared thermography technique, utilize a standard shop air system for the impingement jet air supply, and minimize size and cost. With these constraints in mind, it was difficult to develop a facility where the infrared camera could obtain a direct line of sight to the impingement jet target surface due to the narrow channel between the impingement jet nozzle plate and the impingement jet target surface. As a result, a facility was developed which utilized an electrically heated foil which was cooled by impinging jets. Due to the thin nature of the foil, an infrared camera was able to image the side of the foil opposite of impingement. Figure 2.1 shows an overall view of the test rig with key components annotated. Additionally, a schematic of the layout of the test rig is also shown in Figure 2.2. The remainder of Section 2.1 discusses the design and selection of the components shown in Figures 2.1 and 2.2 in further detail.

Figure 2.1: Completely Assembled Test Rig

Figure 2.2: Schematic of the Test Rig

2.1.1 Flow Configuration

Figure 2.3 depicts a cross sectional schematic of the inlet diffusers and the test section. The test rig is supplied with room temperature air via a shop air supply system. The air supply enters the test section at the transition duct. The transition duct contains a series of diffusers through which the air passes, and then enters the pressure chamber. The transition duct is manufactured primarily of aluminum. All threaded connections were sealed with pipe dope and Teflon tape. Other mating surfaces were sealed with a gasket material. In the pressure chamber, the air is assumed to be at a nearly uniform velocity and static pressure throughout the pressure chamber.

Figure 2.3: Rendering of a Cross Sectional View of the Test Section

The walls of the remainder of the test section were manufactured from an optically transparent polycarbonate. This material was selected due to ease of machining, the ability to continually inspect the flow path in the test rig, and low thermal conductivity. The test section is mounted onto two aluminum frames. Material thickness and bolt pitch were selected to match existing hardware in the test cell. All mating surfaces within the pressure chamber were sealed with clear RTV in order to ensure that no air would leak, which would disrupt the mass flow rate measurement. The impingement jet nozzle plate was designed with 5 rows of 11 nozzles, laid out in a rectangular pattern. Figures 2.4 and 2.5 show dimensional top and side views at sectional planes placed at the

center of the test section. Each impingement jet nozzle consisted of a simple cylindrical hole drilled into the nozzle plate. The hole size diameter (D) is 4.7625mm. The non-dimensional spacing of the holes in the nozzle plate is such that centers of the holes are placed at a regular spacing of x/D = 3 and y/D = 3. Hole spacing was selected by Dr. Shichuan Ou, and was motivated by the lack of results available in the public domain for such a configuration. Although the distance between the hole centers in the array was the same, the distance between the centerline of the outermost rows and the sidewalls was not. In order to maintain periodicity with the sidewalls, the distance between the sidewalls and the centerline of the outer rows should be 1/2 the distance between the centerlines of all other holes in the interior of the array. Due to mounting constraints of the heated foil, the distance between the sidewalls and the centerline of the outermost rows had to be increased 172% to y/D=4.08, rather than the ideal periodic spacing of y/D=1.5. The non-dimensional standoff distance of the foil was based upon the thickness of the sidewalls. Results presented in Section 2.4 were generated using sidewall spacers with a non-dimensional thickness (z/D) of 3, 4, and 5. The nozzle plate and the inlet supply are setup in a "co-flow" arrangement. After passing through the nozzle plate and impacting the impingement surface, the spent jet air can exit in two directions. The exits are maintained at atmospheric pressure. One of the two exists can be blocked with a spacer, forcing all air to exhaust in one direction, increasing the cross flow.

Figure 2.4: Top, Sectional View of the Test Section Flow Path

Figure 2.5: Side, Sectional View of the Test Section Flow Path

Figure 2.6: Side View of the Test Section

2.1.2 Heated Foil Impingement Surface

The heated foil consisted of a 38.1μm thin stainless steel foil. Due to Joule heating, the foil was heated volumetrically by passing a DC electrical current was passed through it. Stainless steel was chosen from a variety of materials because it is the cheapest and most commonly available conductor with a high volumetric resistivity. Nichrome is also a conductor with a very high electrical resistance, and is commonly utilized in the construction of electrical heaters. Nichrome foil, however, was of limited availability, and the high cost prohibited its use in the test rig.

Throughout the test rig design process, the heated foil was assumed to be of uniform thickness, rectangular cross section and rectangular surface area. For a conductor with a uniform, perfectly rectangular cross section, and a perfectly rectangular surface area, the resistance is defined as

$$R = \frac{\rho * l}{w * b} \tag{2.1}$$

where ρ is the volumetric resistivity, l is the length of the conductor, w is the width of the conductor, and b is the thickness of the conductor. In a conductor with resistance R, the electrical power converted to heat due to Joule heating is represented by

$$P = I^2 * R \tag{2.2}$$

Substituting R from Equation (2.1) into Equation (2.2) yields

$$P = \frac{I^2 * \rho * l}{w * b} \tag{2.3}$$

Although Joule heating is a volumetric phenomenon, the heating is assumed to be a surface phenomenon in the foil heater due to the thin cross section of the foil. Lateral conduction in the foil is also assumed to be negligible because of the thin cross section. If the cross section of the foil and the volumetric resistivity are also assumed to be uniform, the heat flux on the foil can be assumed to be uniform and represented by

$$q''_{foil} = \frac{P}{A_s} \tag{2.4}$$

where A_s is the surface area of the foil and is represented by

$$A_s = l * w \tag{2.5}$$

Substituting Equations (2.3) and (2.5) into Equation (2.4), and simplifying yields

$$q''_{foil} = \frac{I^2 * \rho}{b * w^2} \tag{2.6}$$

which results in a uniform, constant surface heat flux boundary condition. As can be seen in Equation (2.6), the heat flux in the foil heater is proportional to the square of the current flowing through the foil, inversely proportional to the square of the width of the heater, inversely proportional to the thickness of the foil, and linearly proportional to the volumetric resistivity of the foil. The heat flux is independent of the length of the foil, provided the power supply circuit can supply a voltage sufficient to overcome the additional resistance, while still maintaining the same electrical current. With stainless steel selected as the material, foil thickness and width were key parameters which

could be modified during the design process in order to vary the maximum heat flux obtainable in the test rig, with the maximum electrical current being limited by the available DC power supply.

Great difficulty was experienced in the development of an assembly procedure for a strong, minimally intrusive, attachment scheme between the thin heated foil and the copper heater circuit busbar in the test rig. This challenge caused a delay of over 6 months in the commissioning of the test rig. Key challenges were due to the high current levels, high temperature gradients within the foil, high temperature gradients between the foil and the rest of the test rig, large difference in thermal expansion coefficient between the foil and the rest of the test rig, and stagnation pressure due to the impingement jets impacting the foil surface, all of which caused a severe wrinkling of the thin stainless steel foil heater. The use of Invar, a material with an extremely low coefficient of thermal expansion was evaluated as an alternative to stainless steel. Although the material's extremely low thermal expansion coefficient would have prevented wrinkling or warping due to temperature gradients induced by impingement jet heat transfer, the material proved to be unusable because its thermal expansion characteristics are too different than any other common materials. As a result, any attachment scheme between Invar foil and the copper busbars resulted in the busbars causing the Invar foil to wrinkle due to the significantly higher expansion or contraction of the copper busbars with changes in temperature. Etched copper Kapton flex circuits were explored due to the ease of solderability of copper, and the common thermal expansion characteristics of a copper Kapton flex circuit and the copper busbars. Etched copper Kapton flex circuits were not used due to the complex manufacturing process, low thermal conductivity of the Kapton substrate which would have caused an increased temperature gradient across the thickness of the foil, as well as the rough surface characteristics of the etched patterns which would be required.

For the stainless steel foil, various attachment strategies were explored which included the use of clamping mechanisms, electrically conductive tape, electrically conductive epoxies, and hot soldering. Clamping mechanisms were deemed inappropriate due to their intrusion into the flow field. Electrically conductive tapes exhibited too high of a bond electrical resistivity. Electrically conductive epoxies advertised reasonably low bond electrical resistivity, however, there was no success obtaining the advertised bond resistivity during the assembly process. As a result, there was excessive heating at the joint between the heated foil and the busbar, which impacted the measurement procedure, and also caused the foil to warp and debond. An intricate soldering

process was developed to attach the thin stainless steel foil to the much thicker set of copper busbars (which connected to the DC electrical power supply), while avoiding wrinkling of the foil and also maintaining a strong, highly conductive bond. With this process, the width of the foil could be reduced to 92.0mm, and the minimum thickness of the foil was 38.1μm stainless steel. A tedious tensioning process was used to ensure that the foil would not distort due to the stagnation pressure of the impingement jets. Due to the difference in thermal expansion between the thin stainless steel foil and the polycarbonate frame in which it was mounted, the foil was re-tensioned periodically as the ambient temperature in the test cell and shop air supply varied with the seasons.

2.1.3 Power Supply and Heat Flux Measurement

Electrical current was supplied to the foil heater by a digitally controlled American Reliance PQ40-165 DC power supply. The device is shown in Figure 2.7. The unit was operated in constant current mode. Due to the relatively low resistance of the foil heater, the unit always ran at a very low voltage, and the safety shutdown limit was set at 5V. A set of 7.348mmø (42.4mm^2/1AWG) copper wires were used to connect the DC power supply to the busbar connections on the test rig. The wire and busbar connection are shown in Figure 2.6. The busbar and cables were sized to carry a maximum of 150ADC. With this maximum current limitation, the foil heater was designed to be capable of heat fluxes up to 56kW/m^2.

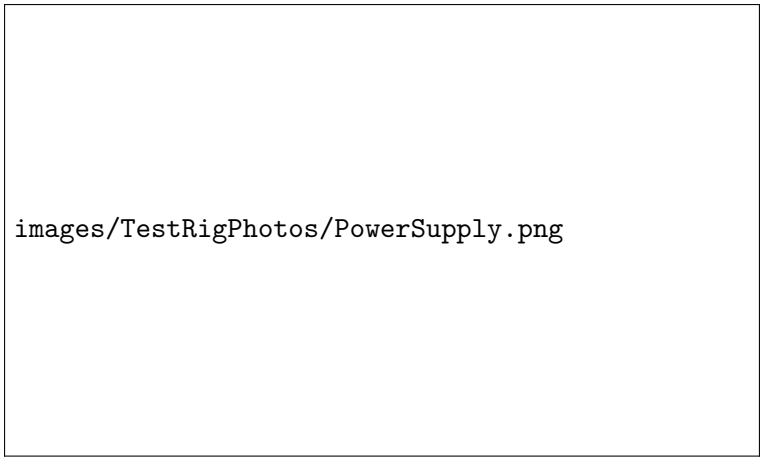


Figure 2.7: DC Power Supply

DC current measurement is commonly performed by measuring the voltage drop across a precision shunt resistor. During the commissioning of the test rig, it was identified that this traditional

technique was not acceptable. The test rig was to be operated throughout a wide range of currents, which would have required several shunt resistors in order to accurately measure the current. In order to keep shunt resistors from operating at high temperatures, they must be selected such that the voltage drop is low. With the shunt resistor connected to the electrical ground, additional electrical noise was introduced, and with the low output voltage, the measurement quality was deemed unacceptable. As an alternative, DC electrical current was instead measured via a CTL-201/200 Hall effect current transducer manufactured by Ohio Semitronics, shown in Figure 2.8. Hall effect sensors have the capability to measure static and dynamic magnetic fields, making them appropriate for DC electrical current measurement. The CTL-201/200 Hall effect current transducer was purchased with a matching amplifier which provided a full scale output of 10V. The transducer/amplifier combination were calibrated as a single unit from Ohio Semitronics. Ohio Semitronics indicated that the unit was calibrated within $\pm 0.5\%$ of the full scale output, and that this calibration was determined with a confidence of 99.875%. Because the Hall effect current transducer measures magnetic field strength, it is not as susceptible to electrical noise because the transducer is not electrically grounded on the transducer end of the measurement circuit. The Hall effect current sensor was configurable for a full scale output of 100A or 200A by changing the number of passes the power cable makes through the transducers measurement window. The third power cable visible in Figure 2.8 allows for ease of selection between the number of passes through the transducers measurement window. The small window size, the large cable diameter and cable connectors, as well as the solid core transducer design complicated this selection procedure, and the third cable was added to overcome these obstacles. Although the power cables, busbars, and current transducer were sized for current levels up to 150ADC, the facility has not yet been tested at currents over 100ADC, as the range of Reynolds numbers surveyed in this study did not produce high enough convective heat transfer rates in order require higher current levels.

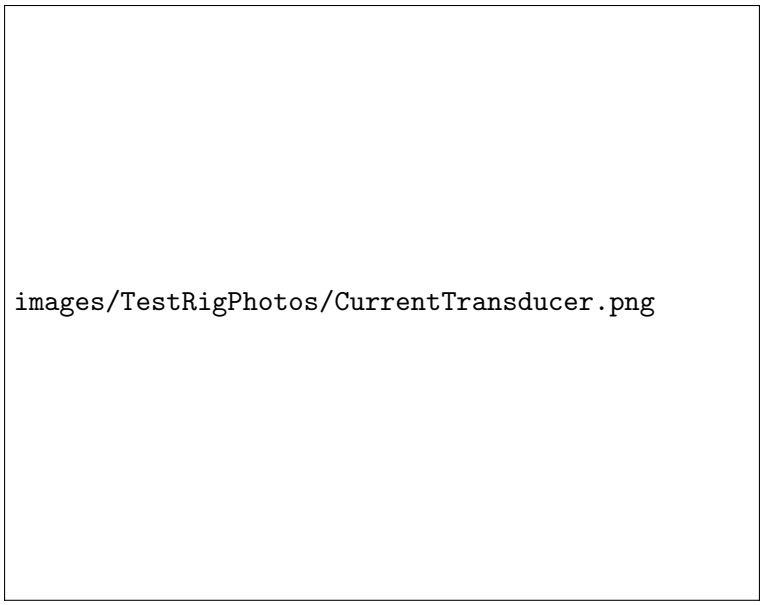


Figure 2.8: Hall Effect Current Transducer

Equation (2.6) used during the design process represents the heat flux in terms of the electrical current, volumetric resistivity, foil thickness, and foil width. Foil width was measured with a ruler during the assembly process. No volumetric resistivity of the stainless steel foil was specified by the manufacturer. The volumetric resistivity of stainless steel foil is dependent upon the alloy of stainless steel, as well as the material temperature. In addition, precise verification of uniform foil thickness was difficult, as the foil was very thin and flexible. Due to all of these uncertainties, measurement of the heated foil resistance directly was desired, rather than calculating it based on derived quantities. Using Ohm's law, the resistance of the foil can be represented by

$$R = \frac{V}{I} \tag{2.7}$$

where V represents the voltage drop across the foil heater. Using Equations (2.2), (2.4), (2.5), and (2.7), the heated foil surface heat flux can also be represented by

$$q''_{foil} = \frac{I * V}{l * w} \tag{2.8}$$

In order to provide closure to Equation (2.8), the voltage drop (V) across the foil heater was also measured by soldering two 0.255mmø (0.051mm^2/30AWG) wires at each end of the measurement

domain, and the distance (l) between the two attachment points was measured with a ruler. This technique proved to be more tedious to implement than measuring voltage drop directly at the busbar connection. However, it eliminated all voltage drops in between the busbar connection and the measurement domain, which can be quite difficult to predict due to the soldering techniques utilized to bond the stainless steel foil to the copper busbar. Because the foil heater circuit was grounded on the negative side of the circuit, the electrical circuit for the voltage measurement was left ungrounded on the end which was connected to the analog to digital conversion hardware, a practice which is contradictory to traditional data acquisition configurations. It is important to note that the resistance measured was a volume weighted average resistance of the foil.

2.1.4 Mass Flow Measurement

Impingement jet heat transfer characteristics are known to be highly dependent upon the impingement jet nozzle Reynolds number (Re_{jet}), hence, it is very important to characterize Re_{jet} accurately. In order to establish the impingement jet nozzle Reynolds number, the nozzle mass flow rate was measured rather than measuring both the mean nozzle flow velocity and the mean nozzle fluid density. Due to the complexity and intrusive nature of measuring the individual Reynolds numbers for flow through each impingement jet nozzle, the total mass flow rate through all impingement jet nozzles was measured, and Re_{jet} was computed based upon the area weighted average impingement jet nozzle mass flow rate. In terms of mass flow rate, the impingement jet nozzle Reynolds number can be represented as

$$Re_{jet} = \frac{4\dot{m}}{N\mu\pi D} \tag{2.9}$$

where $\dot{m}$ is the total mass flow rate through all impingement jet nozzles, N is the total number of jets, μ is the fluid viscosity, and D is the impingement jet nozzle diameter. With air as the working fluid, the fluid viscosity was evaluated using the Sutherland Law for viscosity which states that [8]

$$\frac{\mu}{\mu_0} \approx \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + S}{T + S} \tag{2.10}$$

where μ_0 is the fluid reference viscosity, T_0 is the fluid reference temperature, and S is the Sutherland constant. For air, these parameters are $1.716 * 10^{-5} N * s/m^2$, $273K$, and $111K$, respectively [9]. The temperature of the fluid inside the impingement jet nozzle was assumed to be at the same temperature as the fluid upstream in the pressure chamber, $T_{upstream}$, which was measured with thermocouples. In reality, some drop in temperature is expected as the fluid accelerates into the nozzle, however, this change in temperature was assumed to be negligible due to the low fluid Mach numbers inside of the impingement jet nozzles.

Mass flow rate was measured via a sonic converging diverging nozzle (choked venturi nozzle) due to high accuracy, durable construction, and simple instrumentation technique. As with any flow measurement device, there are limits within which the device can operate. In the case of a choked venturi nozzle, the upper limit on the mass flow rate is bound by the structural limitations of the venturi nozzle, which are dependent upon temperature and pressure, available supply air pressure, and the range of the temperature and pressure measurement devices used. In Research Cell 21, flow was limited on the upper end by the maximum shop air supply pressure, which operated at approximately 8bar absolute. The lower limit on mass flow rate is determined by the minimum mass flow rate in which sonic flow will occur in the throat of the nozzle. Although fluid is still able to flow through a venturi nozzle at lower rates, measuring the mass flow rate becomes a greater challenge when flow at the throat becomes subsonic. Additional temperature and pressure measurements are required in order to establish the fluid velocity and density at the throat of the nozzle, which involves additional instrumentation, signal conditioning, and data acquisition systems, as well as increased venturi nozzle manufacturing costs.

In order for choked flow to occur at the throat of the nozzle, the ratio of the static pressure at the throat of the nozzle (p_{throat}) to the total pressure (p_T) should be less than 0.528 ($p_{throat}/p_T < 0.528$) [10]. As was mentioned earlier, manufacturing costs typically discourage pressure measurements at the throat of the nozzle. As a result, static pressure is typically measured downstream of the nozzle, and the static pressure downstream of the nozzle ($p_{downstream}$) is compared to the total pressure upstream of the nozzle ($p_{downstream}/p_T$). If the nozzle is operating at $p_{downstream}/p_T = 0.528$ and there is negligible total pressure loss between the throat of the nozzle and the location of the downstream static pressure measurement, the ratio of the static pressure at the throat to the upstream total pressure, p_{throat}/p_T, will actually be less than 0.528. This technique allows for a

cheaper, more conservative process for verifying that the appropriate pressure ratio is maintained, while limiting the minimum measurable mass flow rate to a higher value. With the pressure ratio below 0.528 and the flow choked, some portion of the flow transitions to supersonic flow in the divergent portion of the venturi nozzle. Flow then transitions to subsonic flow due to a normal shock wave in the divergent portion of the venturi nozzle. Significant total pressure loss occurs in the nozzle across this normal shock at low pressure ratios. With the required maximum pressure ratio of 0.528, and the maximum supply air pressure in Research Cell 21, a venturi nozzle with a throat diameter of 6.35mm (0.250in) was installed in the test rig in order to achieve the jet Reynolds numbers of 4,000 through 15,000 for the results presented in Section 2.4. The test rig was designed to also accept venturi nozzles of throat diameter 3.18mm (0.125in) and 8.99mm (0.354in) with the 38.1mmø (1.5inchø) piping utilized in the test rig, expanding the testable range of jet Reynolds numbers to 1,000 to 30,000.

Figure 2.9: Uninstalled Converging Diverging Venturi Nozzle

In order to establish the mass flow rate through a choked venturi nozzle, the flow is idealized as 1D, inviscid flow. The mass flow rate at the throat of the nozzle can be represented as

$$\dot{m} = \rho_{throat} * A_{throat} * V_{throat} \tag{2.11}$$

where ρ_{throat}, A_{throat}, and V_{throat} are the density, cross sectional area, and velocity at the throat, respectively. Because the cross section at the throat is a circle, the cross sectional area at the throat can be easily calculated based upon its diameter (D_{throat}) as

$$A_{throat} = \frac{\pi}{4} D_{throat}^2 \tag{2.12}$$

If the venturi nozzle is choked, the velocity of the fluid at the throat is sonic. The speed of sound of a fluid is represented by [10]

$$a = \sqrt{\gamma \left(R_{fluid}\right) T} \tag{2.13}$$

where γ is the ratio of the specific heat at constant pressure (c_p) to the specific heat at constant volume (c_v), R_{fluid} is the specific ideal gas constant of the fluid, and T is the static temperature of the fluid.

The static temperature of a fluid that is accelerated or decelerated to the speed of sound can be represented by [10]

$$T^* = T_T \left(\frac{2}{\gamma + 1}\right) = T_{throat} \tag{2.14}$$

where T_T is the total, or stagnation temperature of the fluid upstream of the venturi nozzle throat. Substituting T_{throat} into Equation (2.13) yields the speed of sound at the venturi nozzle throat as

$$a_{throat} = \sqrt{\gamma \left(R_{fluid}\right) T_T \left(\frac{2}{\gamma + 1}\right)} = V_{throat} \tag{2.15}$$

Using the ideal gas law, the density of the fluid at the throat of the venturi nozzle is

$$\rho_{throat} = \frac{p_{throat}}{R_{fluid} T_{throat}} \tag{2.16}$$

Fluid static pressure can be determined based upon its total pressure (p_T), Mach number (M), and the ratio of specific heats (γ) as [10]

$$p = p_T \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\gamma/(1-\gamma)} \tag{2.17}$$

At the throat of a choked venturi nozzle, the Mach number is 1. Equation (2.17) can then be

further simplified to

$$p_{throat} = p_T \left(\frac{1+\gamma}{2} \right)^{\gamma/(1-\gamma)} \tag{2.18}$$

Upstream of the venturi nozzle, the Mach number is sufficiently low, such that the total pressure (p_T) and temperature (T_T) are approximately equal to the static pressure and temperature. Because of this, the venturi nozzle upstream metering tube was designed such that the static pressure and temperature would be measured, reducing the cost of the unit. In this impingement cooling test rig, p_T was measured with a pressure transducer, and T_T was measured with a thermocouple. Figure 2.10 shows the inlet metering tube, the venturi nozzle, and the outlet metering tube. A pressure transducer with a full scale output of 1.03MPa was used for the upstream pressure measurement. The upstream temperature measurement was made with a 0.127mmø (.013mm^2/36AWG) type J thermocouple. The downstream pressure measurement was made with a pressure transducer with a 34.5kPa full scale output.

Figure 2.10: Nozzle Installed into the Supply Air Line

With the static pressure at the venturi nozzle throat known, the density at the venturi nozzle throat can then be expressed in terms of known quantities using Equations (2.14), (2.16), and (2.18)

as

$$\rho_{throat} = \frac{p_T \left(\frac{1+\gamma}{2}\right)^{\gamma/(1-\gamma)}}{R_{fluid} T_T \left(\frac{2}{\gamma+1}\right)} \tag{2.19}$$

Simplifying Equation (2.19) leads to

$$\rho_{throat} = \frac{p_T}{R_{fluid} T_T} \left(\frac{1+\gamma}{2}\right)^{1/(1-\gamma)} \tag{2.19}$$

Substituting equations (2.12), (2.15), and (2.19) into Equation (2.11) yields

$$\dot{m} = \frac{p_T}{R_{fluid} T_T} \left(\frac{1+\gamma}{2}\right)^{1/(1-\gamma)} * \frac{\pi}{4} D_{throat}^2 * \sqrt{\gamma \left(R_{fluid}\right) T_T \left(\frac{2}{\gamma+1}\right)} \tag{2.20}$$

Simplifying gives

$$\dot{m} = \frac{p_T}{\sqrt{R_{fluid} T_T}} \left(\gamma \left(\frac{1+\gamma}{2}\right)^{(1+\gamma)/(1-\gamma)}\right)^{(1/2)} * \frac{\pi}{4} D_{throat}^2 \tag{2.20}$$

Equation (2.20) represents the mass flow rate through a choked venturi nozzle, assuming ideal gas, 1D, isentropic flow. Equation (2.20) can be rewritten as

$$\dot{m} = C_{ideal}^* * \frac{p_T}{\sqrt{R_{fluid} T_T}} * \frac{\pi}{4} D_{throat}^2 \tag{2.21}$$

with

$$C_{ideal}^* = \left(\gamma \left(\frac{1+\gamma}{2}\right)^{(1+\gamma)/(1-\gamma)}\right)^{(1/2)} \tag{2.22}$$

and is called the one dimensional, isentropic, ideal gas critical flow function. In real flows, there will be some anisentropy and real gas effects. In order to compensate for these deviations, the venturi nozzle was calibrated. The testing laboratory, Colorado Engineering Experiment Station, Inc. (CEESI), provided a modification to Equation (2.21) that can be represented as

$$\dot{m} = C_d * C^* * \frac{p_T}{\sqrt{R_{fluid} T_T}} * \frac{\pi}{4} D_{throat}^2 \tag{2.23}$$

where C_d is the discharge coefficient which accounts for anisentropy and 3D effects in the flow, and C^* is a real gas critical flow function. Both C_d and C^* are determined experimentally by the

venturi nozzle calibration procedure and have been tabulated by CEESI in the calibration report as a function of throat Reynolds number [11]. Because throat Reynolds number is a function of mass flow rate, an iterative process was required to determine mass flow rate based upon the measured values, p_T and T_T. As is evident in Equation (2.23), mass flow rate is linearly proportional to total pressure upstream of the venturi nozzle throat. In order to vary the mass flow rate in the test rig, the upstream pressure was regulated with a pressure regulator which was installed in line with the flow.

2.1.5 Surface Temperature Measurement

Primary impingement surface temperature measurement was performed via a FLIR ThermoCAM SC3000 infrared (IR) camera, which featured an image resolution of 320 by 240 pixels. The camera utilized a GaAs Quantum Well Infrared Photon Detector (QWIP) which featured a spectral range of 8-9μm. The detector was cooled by a Stirling chiller to 70 Kelvin. The IR thermography technique relies on the principle of blackbody radiation to detect the surface temperature of an object. In order for the highest signal to be read by the IR camera, the imaged surface was painted with a black paint which had an emissivity of approximately 0.96. Because of the thin nature of the heated foil impingement surface, both sides of the foil were assumed to be at an equal temperature. With this assumption, the side of the foil opposite of impingement was imaged with the IR camera, obtaining a spatial temperature distribution ($T_s(x, y, t)$) on the foil. The IR camera was attached to a Windows XP workstation via a PCI frame grabber card, and the FLIR ThermaCAM Researcher 2000 software package was utilized to record each image. For each test case, 1500 frames were recorded at a frame rate of 43.78 frames/second and the temperature distribution for each frame was converted to the MATLAB data file format for post processing.

Figure 2.11: Heated Surface as Viewed by the IR Camera

Figure 2.12: Infrared Camera Mounted in the Test Rig

A mapping and alignment procedure was developed to map the physical space to that of each IR camera pixel, and focus the camera. Although the IR camera records images with a size of 320 pixels x 240 pixels, the measurement domain was reduced because the aspect ratio of the measurement

domain did not match the aspect ratio of the IR camera. The measurement domain was reduced additionally due to excessive thermal conduction near the connection of the thick copper busbar to the thin stainless steel foil. Because of this effect, the measurement domain had to be truncated such that an appropriate attenuation of this heat sink effect was achieved. Unfortunately, this reduced the field of view such that only 45 impingement jets could be imaged, rather that the total 55 present in the test rig. In the high cross flow test case (single exit), the total number of jets imaged was further reduced to 40 due to bending of the jets outside of the measurement domain due to high cross flow. The resultant image resolution was approximately 1.8 pixels/mm.

In addition to surface temperature measurement with the IR camera, two 0.127mmø ($0.013mm^2$/36AWG) type J thermocouples were soldered to the foil to allow for verification of the IR camera temperature reading. Due to the voltage gradients in the foil, difficulty in placing the thermocouples at precisely the same x location on the foil, and minor variations in resistance within the foil, the thermocouples were configured to use separate reference junctions and were left ungrounded at the connection to the analog to digital conversion hardware, a configuration contrary to traditional data acquisition configurations.

2.1.6 Pressure Chamber Instrumentation

The fluid temperature was measured in the pressure chamber of the test rig using 0.127mmø ($0.013mm^2$/36AWG) type J thermocouples. Three thermocouples were utilized in the pressure chamber, and the measured values were averaged. Because this temperature is measured upstream of the impingement jet nozzles, this average temperature is designated $T_{upstream}$. For all cases there was less than .5 K variation between the maximum and minimum temperatures measured in the pressure chamber.

Although used for references purposes only in the experimental component of this study, a pressure transducer was also installed to measure static gauge pressure in the pressure chamber. The full scale output of this pressure transducer was 17.24kPa. When operating at a jet Reynolds number of 4,000, the transducer was reading approximately 340Pa, and at a jet Reynolds number of 15,000, the transducer was reading approximately 2.05kPa. During the design process, larger static pressures were anticipated in the pressure chamber during testing. Because of the mismatch between the full scale pressure transducer measurement and the actual pressures being measured,

static pressure measured in the pressure chamber was not used for boundary conditions for the numerical simulations in Chapter 3. In future studies, a more appropriate pressure transducer is recommended if accurate pressure data in the pressure chamber is desired.

2.1.7 Signal Conditioning and Data Acquisition

Except for the IR camera, all instrumentation was attached to a National Instruments SCXI series signal conditioning and PXI series analog to digital (A/D) conversion hardware, shown in Figure 2.13. Data was set to be recorded at 200 samples/second for a period of 1 second. The data was then time averaged in order to eliminate any noise introduced by the instrumentation, wiring, etc. The National Instruments LabView software running on a Microsoft Windows XP workstation was utilized to record data, perform averaging, calculate Reynolds number and heat flux, and to monitor all parameters.

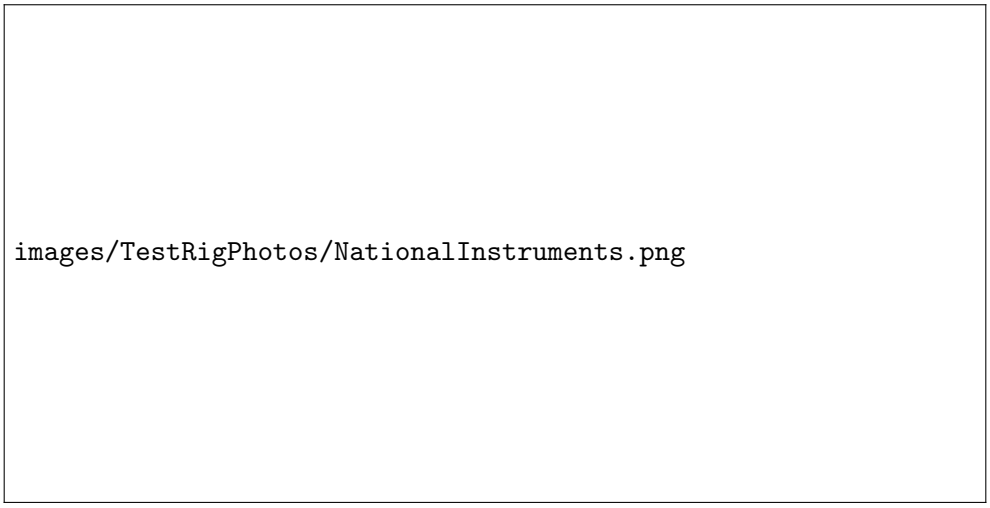


Figure 2.13: National Instruments Data Acquisition System

2.2 Test Sequence

Each day of testing began by first ensuring that the foil heater was tight. If there was a significant change in test cell temperature since the last day of testing, the foil could be warped due to thermal expansion. If the foil was warped, it first was re-tensioned. Next, the test rig was configured for the desired impingement jet target spacing (z/D spacing), and was then completely assembled. The LabView and IR camera workstations were powered on. The IR camera was powered on. Because

the IR camera incorporates a Stirling chiller in order to keep the detector at the appropriate operating temperature of 70K, the camera had to be powered on approximately 6 minutes in advance in order to give the device time to reach this temperature. The ambient atmospheric temperature was entered into the FLIR Researcher software so that corrections could be made to account for atmospheric transmission effects. The atmospheric pressure was entered into the LabView interface. Bridge circuits on all pressure transducers were also rebalanced in order to accommodate for the shift in atmospheric pressure since the last day of testing.

The air supply system was unlocked and powered on, but the pressure regulator was set at 0 Pa. Electrical power was supplied to the DC power supply, and the unit was powered on, but set at a current of 0 Amps. The signal conditioning and A/D conversion hardware was powered on. The supply air pressure was slowly regulated until the desired Reynolds number is achieved. Once the desired Reynolds number was achieved, electrical power was supplied to the DC power supply and the unit was powered on. In order to avoid thermal shock of the foil, the current was increased such that the increase in heat flux was less than $300 W/(m^2 * minute)$. When the maximum temperature of the foil reached approximately 75°C and stabilized, the heat flux was held constant. The thermocouples soldered to the foil were monitored in order to verify that the temperature had indeed stabilized. This process took approximately 1 hour.

Once the temperature had stabilized, data was then recorded by both the LabView system and the IR camera workstation. Records were also made in an electronic spreadsheet in order to log the current flow configuration (jet Reynolds number, target spacing, exit configuration, etc.) After data had been recorded, one of the exits was blocked. The current flowing through the foil was then adjusted in order to maintain a peak surface temperature of approximately 75°C. Once the surface temperature had re-stabilized (approximately 1 hour later), data was again recorded by both the LabView system and the IR camera workstation. The Reynolds number was then increased. After the Reynolds number was set at the next test point, the electrical current was adjusted, and the temperature was allowed to stabilize. Data was recorded again. The spacer was then removed from the exit, the electrical current was adjusted, and the temperature was allowed to stabilize. Data was taken, the Reynolds number was increased, and the process then repeated until data for all desired Reynolds numbers and exit flow configurations was obtained for the current target spacing. After all testing was complete, the supply pressure/Reynolds number and current/heat flux were

slowly decreased, avoiding thermal shock of the foil. Once supply pressure and current had been reduced to zero, the supply pressure and power supply were shut down. After the shutdown was complete, each frame recorded by the IR camera was exported as a MATLAB .MAT file for post processing.

2.3 Post Processing Methodology

Incorporating all of the assumptions indicated in the previous sections in Chapter 2, the heat flux of the impingement jets can be represented, for each flow configuration (Reynolds number, target spacing, and exit configuration) by

$$q''_{jets} = h(x,y,t) * (T_s(x,y,t) - T_{upstream}) = q''_{foil} - q''_{losses} \tag{2.24}$$

Equation (2.24) can then be solved for $h(x,y,t)$, the heat transfer coefficient, to obtain

$$h(x,y,t) = \left(q''_{foil} - q''_{losses}\right) \frac{1}{(T_s(x,y,t) - T_{upstream})} \tag{2.25}$$

The local thermal conductivity of air, $k(x,y,t)$, was evaluated using a linear fit of the thermal conductivity of air vs. temperature, along with the local film temperature, $T_{film}(x,y,t)$, on the impingement side of the foil. The film temperature was evaluated by taking the average of the upstream air temperature ($T_{upstream}$) and the surface temperature ($T_s(x,y,t)$). The non-dimensional heat transfer coefficient (Nusselt Number) could then be represented as

$$Nu(x,y,t) = \frac{h(x,y,t) * D}{k(x,y,t)} \tag{2.26}$$

Equation (2.24) includes the term q''_{losses} which accounts for some heat losses from the foil heater that was not due to impingement jet heat transfer. All unheated boundaries in the test rig are manufactured of an optically transparent polycarbonate which has a low thermal conductivity (0.190 W/(m*K) [12]). Due to this low thermal conductivity, the assumption was made that these surfaces are adiabatic. Natural convection losses on the imaged side of the foil heater were accounted for using a correlation for laminar natural convection on a vertical isothermal surface. The foil

heater surface is not isothermal, but was presumed to locally behave as an isothermal surface because the dominant factor driving the temperature on the foil heater was forced convection due to the impingement jets, rather than natural convection. The foil heater surface also did exhibit some edge effects which may have caused natural convection to deviate from the ideal vertical surface. Nusselt number for laminar natural convection of air on an isothermal vertical surface is represented by [13]

$$Nu_{natural} = 0.387 Ra_y^{1/4} \tag{2.27}$$

where Ra_y is the Rayleigh number and is represented by

$$Ra_y = \frac{g * \beta * (T_s - T_{ambient}) * y^3}{\alpha * \nu} \tag{2.28}$$

In Equation (2.28) the vertical dimension, y is relative to the bottom of the imageable edge of the foil heater. $T_{ambient}$ is the static temperature of the ambient air in the test cell. g is the acceleration due to the combined effects of gravity and centrifugal acceleration at the earth's surface and is assumed to be constant. β is the expansion coefficient, which for an ideal gas is the inverse of the film temperature. α is the thermal diffusivity of the fluid which is defined as $\alpha = k/(\rho_{fluid} c_p)$. ν is the kinematic viscosity, which is the ratio of the fluid dynamic viscosity (μ) and the fluid density (ρ_{fluid}).

In an effort to account for the variation in surface temperature in some way, the Rayleigh number was computed locally based upon not only the vertical direction, but also on the spatial variation in surface temperature. A spanwise average surface temperature was evaluated for each horizontal position (x) and each time (t), and can be represented analytically by

$$T_{s,spanwise}(x,t) = \frac{1}{H} \int_0^H T_s(x,y,t) dy \tag{2.29}$$

where H represents the height of the imageable region of the foil heater. The spanwise average film temperature on the natural convection side of the foil ($T_{film,natural}(x,t)$) was then evaluated by taking the average of the spanwise average surface temperature ($T_{s,spanwise}(x,t)$) and the ambient temperature in the test cell ($T_{ambient}$). This spanwise average film temperature on the natural

convection side was then used to evaluate the thermal diffusivity ($\alpha_{film,natural}$) and kinematic viscosity ($\nu_{film,natural}$). When evaluating the thermal diffusivity and kinematic viscosity at this spanwise average film temperature, specific heat of air was assumed to be constant, density was evaluated using the ideal gas law, the Sutherland Law for viscosity was used to account for the temperature dependence of viscosity, and the thermal conductivity was evaluated using a linear fit, as was done for equation (2.26). The resultant local Rayleigh number

$$Ra_y(x,t) = \frac{g * (T_{s,spanwise}(x,t) - T_{ambient}) * y^3}{T_{s,spanwise}(x,t) * \alpha_{film,natural} * \nu_{film,natural}} \tag{2.30}$$

For all cases, this local Rayleigh number was monitored and verified to be less than 10^9 everywhere, indicating that the flow should be laminar, and within the range of the correlation in Equation (2.27).

Using the definition of the Nusselt number and the correlation shown in Equation (2.27), the heat transfer coefficient due to natural convection on the imaged side can be estimated as

$$h_{natural}(x,y,t) = \frac{\left(0.387 Ra_y(x,t)^{1/4}\right) * k_{natural}(x,t)}{y} \tag{2.31}$$

where $k_{natural}(x,y,t)$ is the local thermal conductivity evaluated at the spanwise average film temperature on the natural convection side ($T_{film,natural}(x,t)$), using a linear fit for the thermal conductivity of air. With this estimate for the natural convection heat transfer coefficient on the imaged side of the foil heater, an estimate for the heat flux due to natural convection heat losses can be made.

$$q''_{loss,natural}(x,y,t) = h_{natural}(x,y,t) * (T_s(x,y,t) - T_{ambient}) \tag{2.32}$$

It is evident that this technique is not rigorous, however, the approach was utilized due to the lack of a better alternative.

In addition to losses due to natural convection, there were also losses on the imaged side of the heated foil due to radiative heat transfer. When estimating radiative heat losses, the heated foil was assumed to have a view factor to its surroundings in the test cell of unity. The test cell was assumed to be at uniform temperature. With these assumptions, the heat flux on the surface due

to radiative exchange can be estimated using the Stefan-Boltzmann law as

$$q''_{loss,radiation}(x,y,t) = \epsilon * \sigma * (T_s(x,y,t)^4 - T^4_{ambient}) \tag{2.33}$$

where σ is the Stefan-Boltzmann constant, and ϵ is the surface emissivity, which was approximately 0.96 for the black paint. Radiative heat transfer on the side of the foil heater which was cooled by the impingement jets was considered to be negligible due to the low emissivity of unpainted stainless steel.

The total estimated heat losses on the imaged side of the heated foil surface are the combination of the natural convection and radiative heat losses:

$$q''_{losses}(x,y,t) = q''_{loss,natural}(x,y,t) + q''_{loss,radiation}(x,y,t) \tag{2.34}$$

All operations laid out in Equations (2.24) thru (2.34) were performed numerically using the software application MATLAB. Scripts were developed to facilitate this process and can be reviewed in Appendix A.2. The reader is encouraged to review these scripts, as the references for any physical constants used are listed within the source code, rather than the References section. Results were time averaged as well as span averaged. In an effort to simplify the post processing scripts, all variables were stored in memory until the post processing script terminated. With this simplicity came added memory demands which required the use of the 64 bit version of MATLAB. Surface contours as well as line plots of spanwise averages are plotted and are presented in Section 2.4. For selected cases, contours of Nusselt number were plotted for each time step. An animations was then generated of the contours of Nusselt number using the command line tool FFmpeg.

2.4 Results and Discussion

Results obtained using the experimental test rig are presented and discussed in Sections 2.4.1 to 2.4.5. A summary of all cases is shown in Table 2.1.

Jet Reynolds Number	Hole Spacing (x/D)	Hole Spacing (y/D)	Target Spacing (z/D)	Hole Pattern (x × y)	Exit Configuration	Sample Time (s)	Test Date
4,000	3	3	3	11×5	Double	34.26	2011-02-16
8,000	3	3	3	11×5	Double	34.26	2011-02-16
12,000	3	3	3	11×5	Double	34.26	2011-02-16
15,000	3	3	3	11×5	Double	34.26	2011-02-16
4,000	3	3	3	11×5	Single	34.26	2011-02-16
8,000	3	3	3	11×5	Single	34.26	2011-02-16
12,000	3	3	3	11×5	Single	34.26	2011-02-16
15,000	3	3	3	11×5	Single	34.26	2011-02-16
4,000	3	3	4	11×5	Double	34.26	2011-02-17
8,000	3	3	4	11×5	Double	34.26	2011-02-17
12,000	3	3	4	11×5	Double	34.26	2011-02-17
15,000	3	3	4	11×5	Double	34.26	2011-02-17
4,000	3	3	4	11×5	Single	34.26	2011-02-17
8,000	3	3	4	11×5	Single	34.26	2011-02-17
12,000	3	3	4	11×5	Single	34.26	2011-02-17
15,000	3	3	4	11×5	Single	34.26	2011-02-17
4,000	3	3	5	11×5	Double	34.26	2011-02-18
8,000	3	3	5	11×5	Double	34.26	2011-02-18
12,000	3	3	5	11×5	Double	34.26	2011-02-18
15,000	3	3	5	11×5	Double	34.26	2011-02-18
4,000	3	3	5	11×5	Single	34.26	2011-02-18
8,000	3	3	5	11×5	Single	34.26	2011-02-18
12,000	3	3	5	11×5	Single	34.26	2011-02-18
15,000	3	3	5	11×5	Single	34.26	2011-02-18

Table 2.1: List of Experimental Test Cases

2.4.1 Non-Dimensional Jet Standoff of z/D=3

Double Exit - Time Averaged Nusselt Number

Results obtained from the experimental test rig with two exits and at a non-dimensional target spacing of z/D=3 are presented in Figures 2.14 to 2.18. Spatial dependence on Nusselt number is presented for both the x and y directions. The spatial dimensions have been non-dimensionalized based upon the impingement jet nozzle diameter. The intersection of each grid line represents the center of each impingement jet nozzle. Although these results were obtained via an IR camera with a fixed resolution of approximately 1.8 pixels/mm, the smoothing of the results was due to an interpolation scheme which was built into MATLAB's contour plotting functionality. As can

be seen, Nusselt number (non-dimensional heat transfer coefficient) is highly spatially dependent. At the core of the impingement jets, the convective heat transfer coefficient is the highest. At the center of the impingement jet array (x/D=0) cross flow is at a minimum, if at all, and the variation in Nusselt number is primarily a radial phenomenon. The heat transfer coefficient of the center (x/D=0) is affected by the jet to jet interactions.

As was mentioned in Section 2.1.5, the measurement domain had to be reduced to observe only 45 jets for the double exit configuration due to challenges with the measurement technique near the connection of the stainless steel foil to the copper busbar. This reduction consisted of 1 row being truncated on the left and the right sides of the measurement domain. In the single exit configuration, the bending of the jet core was different from that of the double exit configuration, which resulted in an additional 1/2 of a jet being truncated on each side of the measurement domain in order to include an integer number of rows of impingement jets, so that only 40 jets were included in the measurement domain. In the single exit configuration, this reduction consisted of 1 row being truncated on the left side of the measurement domain and 2 rows were truncated on the right side of the measurement domain.

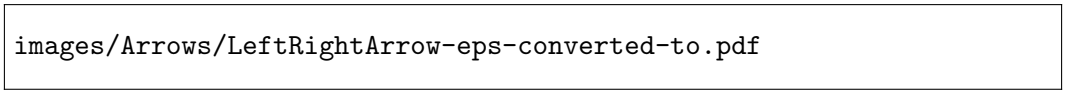


Figure 2.14: Re=4,000 - Double Exit - z/D=3 - Time Averaged Nusselt Number - Experimental

As the flow travels towards the exits, cross flow levels increase and the Nusselt number is no longer primarily a radial phenomenon. The cores of the jets are bent, and the region of highest

heat transfer coefficient is no longer in line with the centers of the impingement jet nozzles. An interesting characteristic is that the region of highest heat transfer coefficient is shifted in both the x and y directions as the flow travels towards the exits. The shape of the region of highest heat transfer coefficient also changes with both spatial directions. These variations in the y direction are believed to be caused by the additional space between the outer rows and the sidewalls when compared to the spacing between jets (as was described in Section 2.1.1), rather than an interaction with the walls themselves.

Figure 2.15: Re=8,000 - Double Exit - z/D=3 - Time Averaged Nusselt Number - Experimental

It is evident that the heat transfer coefficient at the center of the jets is lower for the downstream jets (closer to the exhausts) compared to the upstream jets (near x=0). Additionally, the local heat transfer coefficient appears to decrease less rapidly with the distance from the jet centers for the downstream jets compared to the upstream jets. Because the cross flow levels are much higher closer towards the exits, the cross flow is believed to facilitate enhanced mixing in the areas outside of the jet core, leading to the higher heat transfer coefficients outside of the jet core.

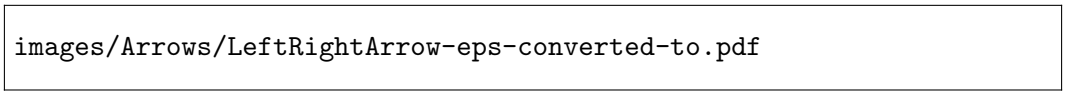
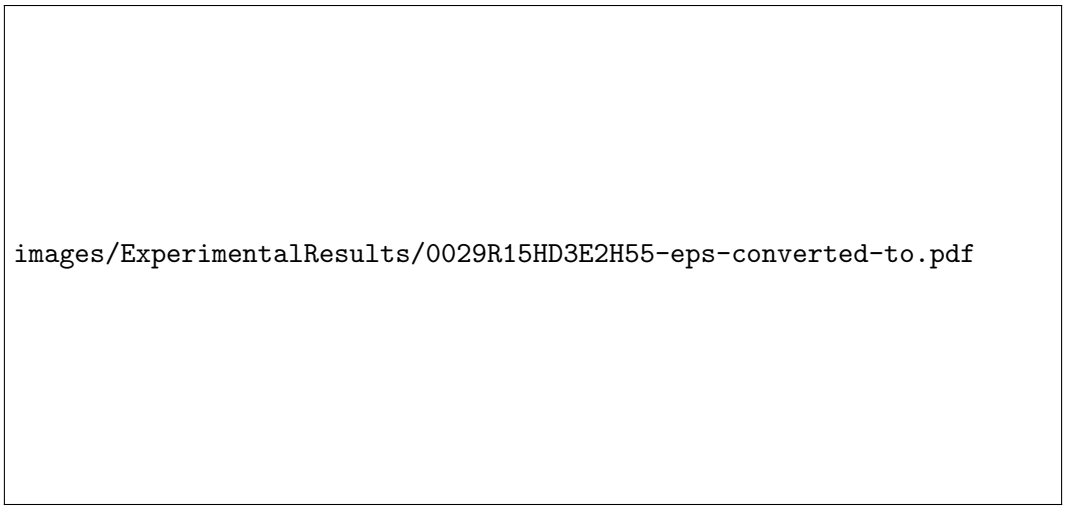


Figure 2.16: Re=12,000 - Double Exit - z/D=3 - Time Averaged Nusselt Number - Experimental

As can be seen when reviewing Figures 2.14 through 2.18, the Nusselt number is highly dependent upon jet Reynolds number. For the higher Reynolds number cases, some minor leakage of air was observed during the testing process, primarily around the bottom edge of the heated foil. This caused some additional convection heat transfer on the imaged side of the heated foil, slightly affecting the measured heat transfer coefficient. In all cases, near x/D=0 and y/D=$\pm$7.5 there were also small regions where the measured heat transfer coefficient was adversely affected by the thermocouples that were soldered to the foil surface. These thermocouples can be viewed in Figure 2.11. Therefor, caution is advised when interpreting data in these spatial regions.

In an effort to provide consistency in comparing results from multiple cases, a fixed scale of 0 to 100 has been selected for all contours of Nusselt number. The higher Reynolds number cases have some saturation in the contours presented. As a result, it is advised that the results for spanwise average Nusselt number be compared in order to gain more quantitative insight of Nusselt Number.



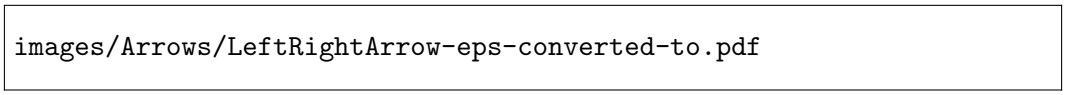


Figure 2.17: Re=15,000 - Double Exit - z/D=3 - Time Averaged Nusselt Number - Experimental

Figure 2.18 presents a time and span averaged Nusselt number for all Reynolds numbers with two exits and a non-dimensional standoff distance of z/D=3. As can be seen, the time and span averaged Nusselt number has a decreasing trend with distance from the center of the impingement jet array. The maximum value of spanwise average Nusselt number is not located at x/D=0. It is currently uncertain whether the slight misalignment of the center jet is due to a misalignment of the IR camera or a physical characteristic of the flow.

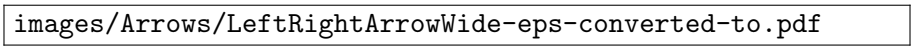


Figure 2.18: Double Exit - z/D=3 - Time and Span Averaged Nusselt Number - Experimental

Double Exit - Unsteady Nusselt Number

As can be seen in Figures 2.19 and 2.20, there is unsteadiness present in the Nusselt number throughout the flow field. A detailed study was not attempted to spatially analyze the experimental results in the time and frequency domains for each flow configuration. Preliminary investigations showed that the frequency of the dominant unsteadiness in Nusselt number was on the order of 1 Hertz for the double exit case at a jet Reynolds number of 4,000 and a target spacing of z/D=3. Future work could be conducted to further quantify the spatial and temporal variation in Nusselt number for each flow configuration. There is some question, however, to the degree that the heat capacity of the foil heater may dampen the measured time dependent characteristics. Further investigations should be conducted to evaluate this effect prior to analyzing the present results in more detail in the time and frequency domains.

Figure 2.19: Re=4,000 - Double Exit - z/D=3 - Nusselt Number - Experimental

Figure 2.20: Re=15,000 - Double Exit - z/D=3 - Nusselt Number - Experimental

Single Exit - Time Averaged Nusselt Number

Figures 2.21 to 2.25 present results for time averaged Nusselt number for the single exit configuration with a non-dimensional target spacing of z/D=3 for each jet Reynolds number. As can be seen, heat transfer coefficient is highest at the left which corresponds to the closed end where cross flow is at a minimum. Heat transfer coefficient is lowest at the right, which corresponds to the location

closest to the exit, because cross flow is at maximum near the exit. Similarly to Figures 2.14 to 2.18, as cross flow is increased, heat transfer coefficient appears to reduce less rapidly with the distance from the center of the jet core. In Figures 2.21 to 2.25, this effect is more pronounced closer towards the exit due to the increased level of cross flow.

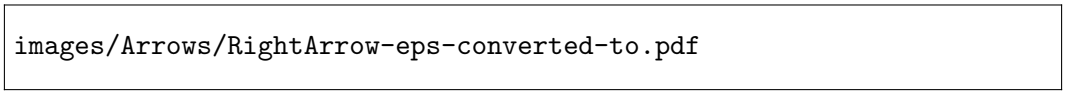


Figure 2.21: Re=4,000 - Single Exit - z/D=3 - Time Averaged Nusselt Number - Experimental

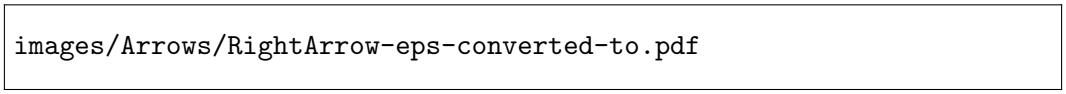


Figure 2.22: Re=8,000 - Single Exit - z/D=3 - Time Averaged Nusselt Number - Experimental

When comparing Figures 2.21 to 2.25 for the singe exit configuration to Figures 2.14 to 2.18 for the double exit configuration it is interesting to note that, in addition to the reduction heat transfer coefficient as the flow progresses towards the single exit, there is also a staggering of the region of highest heat transfer coefficient near the left region (closed end) of the measurement domain. No explanation for this effect has been identified. Although not presented in this document, animations of unsteady Nusselt number were generated using the 34.26 seconds of data and reviewed, and did not indicate that the staggering effect was an unsteady phenomenon.

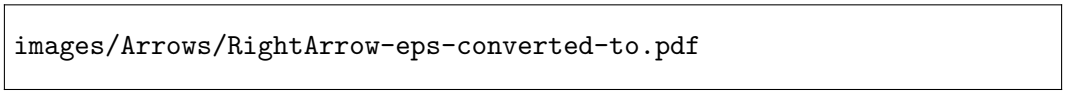


Figure 2.23: Re=12,000 - Single Exit - z/D=3 - Time Averaged Nusselt Number - Experimental

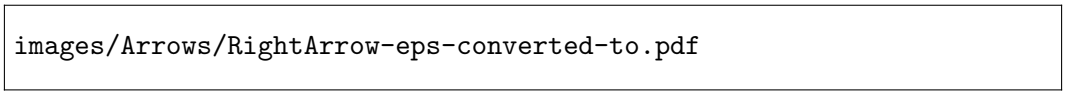


Figure 2.24: Re=15,000 - Single Exit - z/D=3 - Time Averaged Nusselt Number - Experimental

Figure 2.25 presents the time and spanwise average Nusselt number for all Reynolds numbers

at a non-dimensional target spacing of z/D=3 and a single exit configuration. As with the double exit configuration, with an increasing amount of cross flow, the maximum time and span averaged Nusselt number is decreased and the minimum time and span averaged Nusselt number is increased. With the increased cross flow levels present in the single exit configuration, one can observe that there seems to be somewhat of a critical cross flow level at a non-dimensional position of x/D=3 where the time and span averaged Nusselt number tends to decrease more rapidly for the higher jet Reynolds number cases, particularly at a jet Reynolds number of 15,000.

Figure 2.25: Single Exit - z/D=3 - Time and Span Averaged Nusselt Number - Experimental

Single Exit vs Double Exit - Time and Span Averaged Nusselt Number

Figures 2.26 to 2.29 compare time and spanwise average Nusselt for the single and double exit configurations, at the same Reynolds number. It is evident that the time and spanwise averaged Nusselt number is significantly higher for the majority of the surface for the double exit configuration.

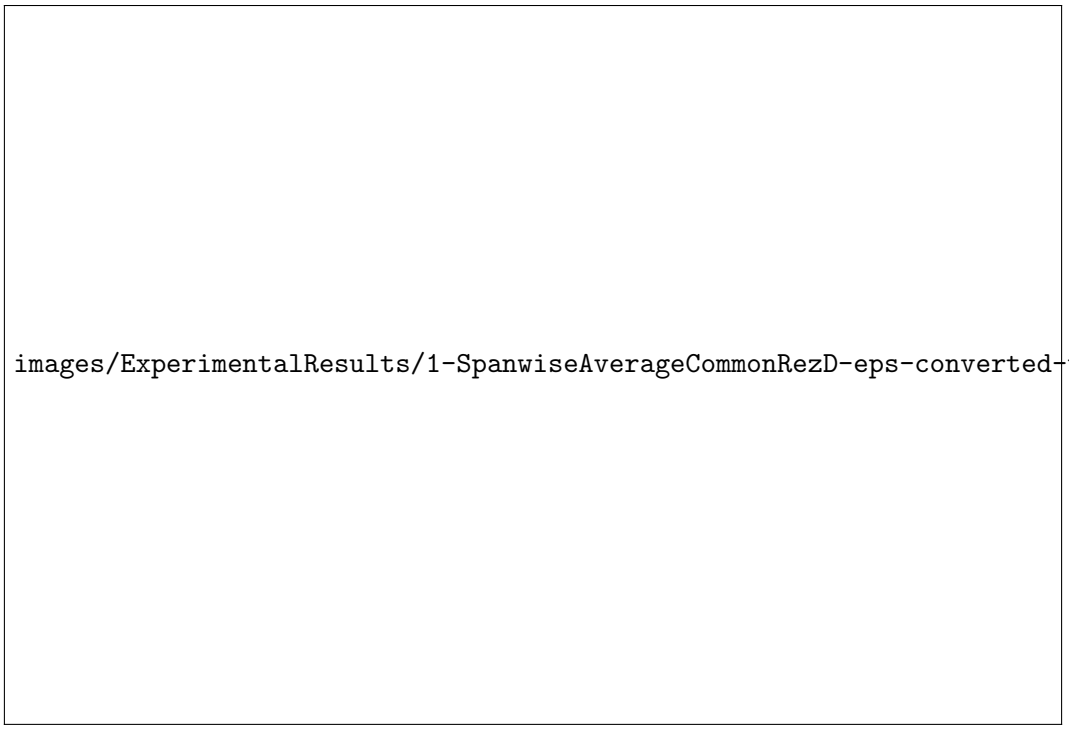


Figure 2.26: Re=4,000 - z/D=3 - Time and Span Averaged Nusselt Number - Experimental

Figure 2.27: Re=8,000 - z/D=3 - Time and Span Averaged Nusselt Number - Experimental

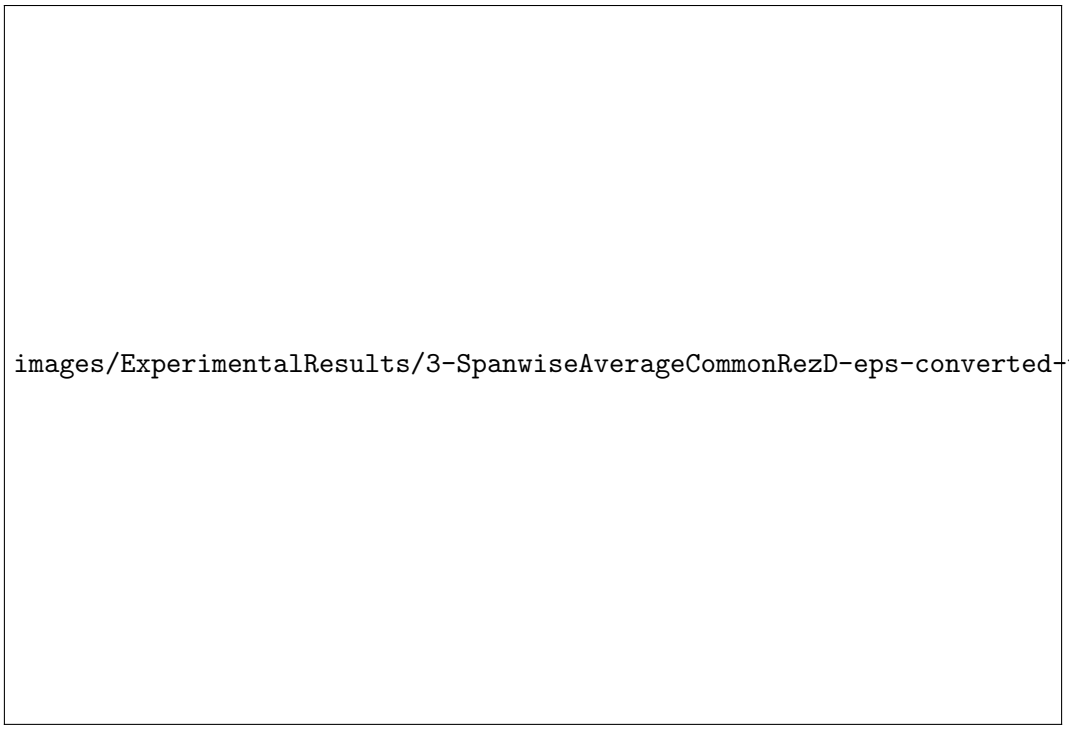


Figure 2.28: Re=12,000 - z/D=3 - Time and Span Averaged Nusselt Number - Experimental

Figure 2.29: Re=15,000 - z/D=3 - Time and Span Averaged Nusselt Number - Experimental

2.4.2 Non-Dimensional Jet Standoff of z/D=4

Figures 2.30 to 2.43 show the time averaged Nusselt number and time and span averaged Nusselt number for a non-dimensional target spacing of z/D=4 for each jet Reynolds number tested and both exit flow configurations. Although quantitatively lower than the z/D=3 cases, the results appear qualitatively similar. At first thought, one might anticipate higher convective heat transfer coefficients for the z/D=4 cases compared to the z/D=3 cases due to less intense cross flow interactions resulting from the increased cross sectional area for spent jet air to travel towards the exhaust(s). It appears, however, that the further deceleration of the flow prior to impacting the target surface must cause a larger reduction in convective heat transfer than an increase in convective heat transfer due to reduced cross flow.

Double Exit - Time Averaged Nusselt Number

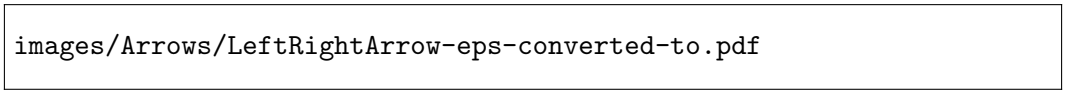


Figure 2.30: Re=4,000 - Double Exit - z/D=4 - Time Averaged Nusselt Number - Experimental

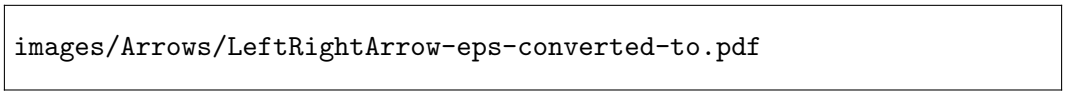


Figure 2.31: Re=8,000 - Double Exit - z/D=4 - Time Averaged Nusselt Number - Experimental

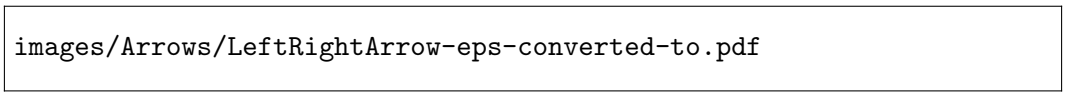


Figure 2.32: Re=12,000 - Double Exit - z/D=4 - Time Averaged Nusselt Number - Experimental

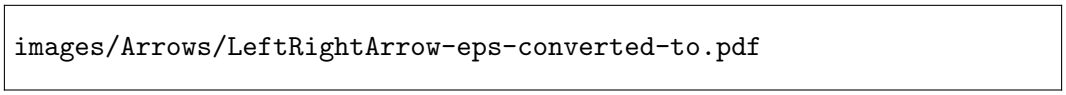


Figure 2.33: Re=15,000 - Double Exit - z/D=4 - Time Averaged Nusselt Number - Experimental

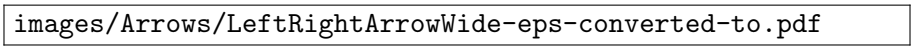


Figure 2.34: Double Exit - z/D=4 - Time and Span Averaged Nusselt Number - Experimental

Single Exit - Time Averaged Nusselt Number

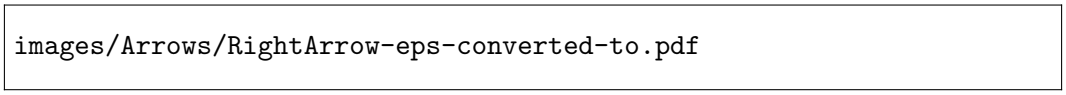


Figure 2.35: Re=4,000 - Single Exit - z/D=4 - Time Averaged Nusselt Number - Experimental

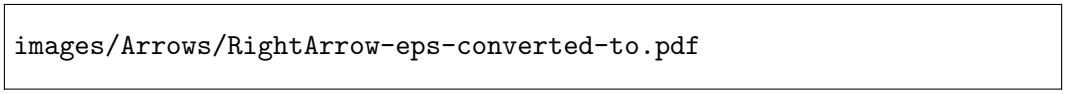


Figure 2.36: Re=8,000 - Single Exit - z/D=4 - Time Averaged Nusselt Number - Experimental

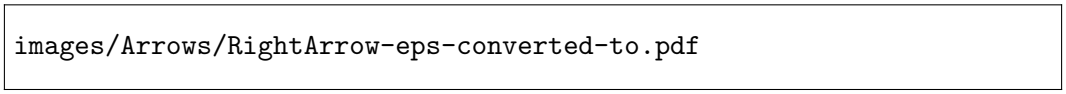


Figure 2.37: Re=12,000 - Single Exit - z/D=4 - Time Averaged Nusselt Number - Experimental

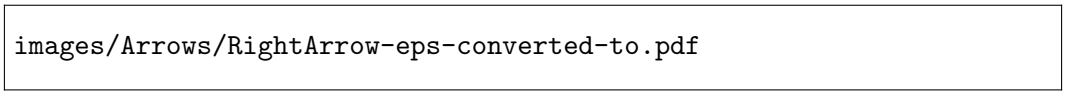


Figure 2.38: Re=15,000 - Single Exit - z/D=4 - Time Averaged Nusselt Number - Experimental

images/Arrows/RightArrowWide-eps-converted-to.pdf

Figure 2.39: Single Exit - z/D=4 - Time and Span Averaged Nusselt Number - Experimental

Single Exit vs Double Exit - Time and Span Averaged Nusselt Number

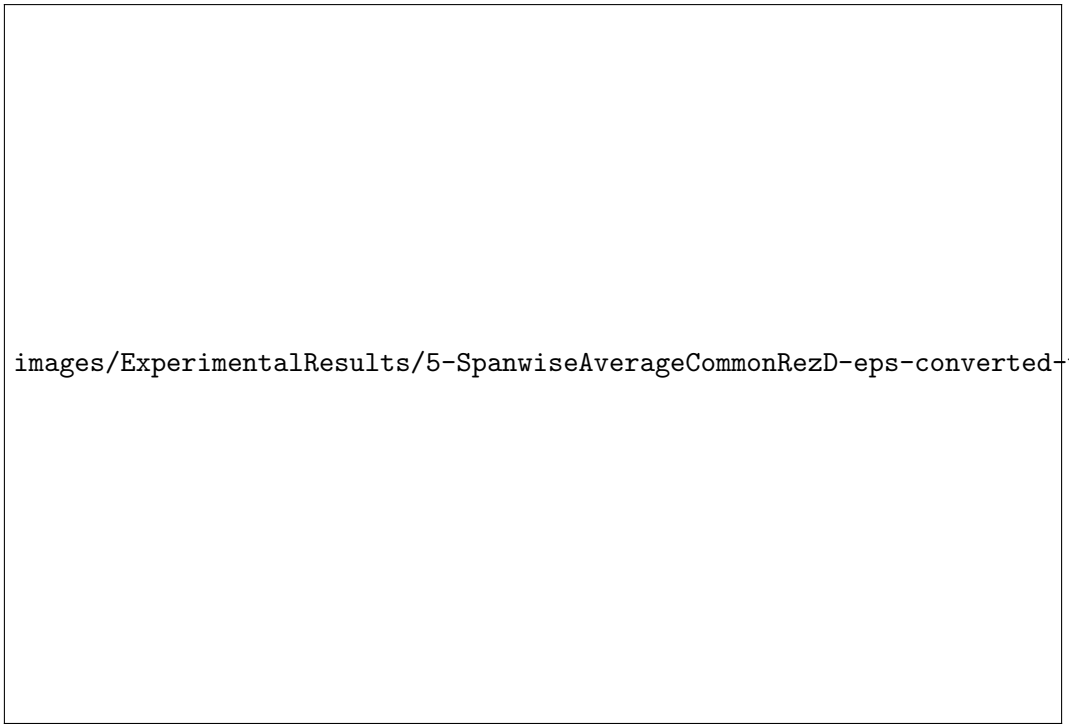


Figure 2.40: Re=4,000 - z/D=4 - Time and Span Averaged Nusselt Number - Experimental

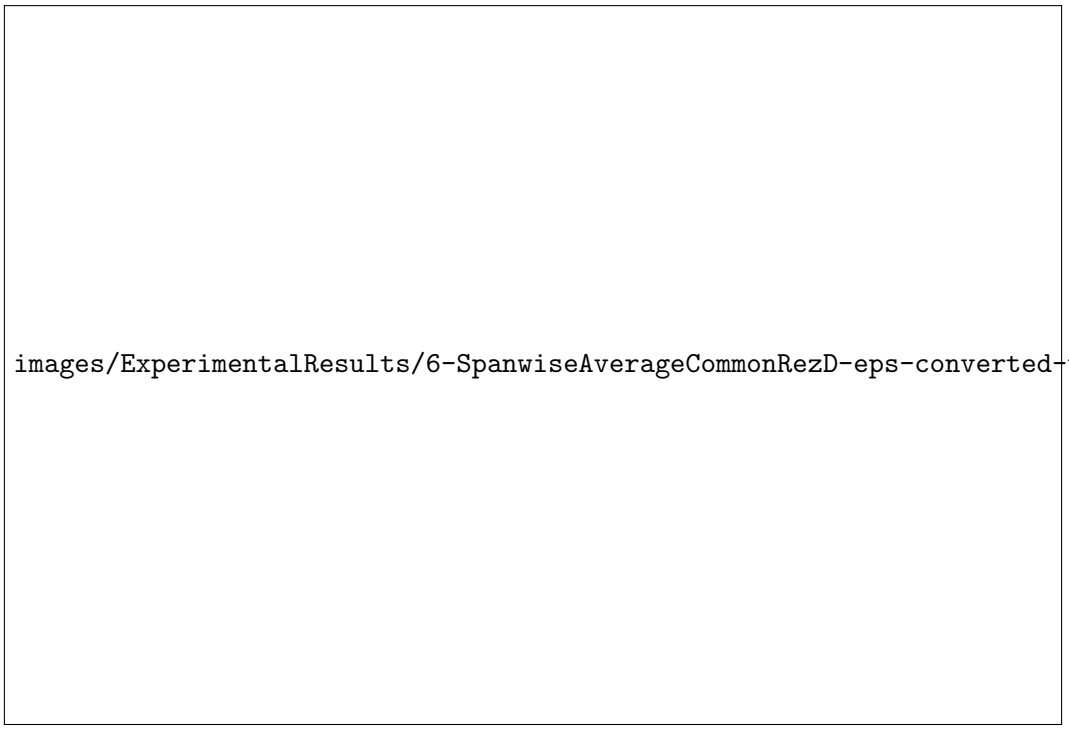


Figure 2.41: Re=8,000 - z/D=4 - Time and Span Averaged Nusselt Number - Experimental

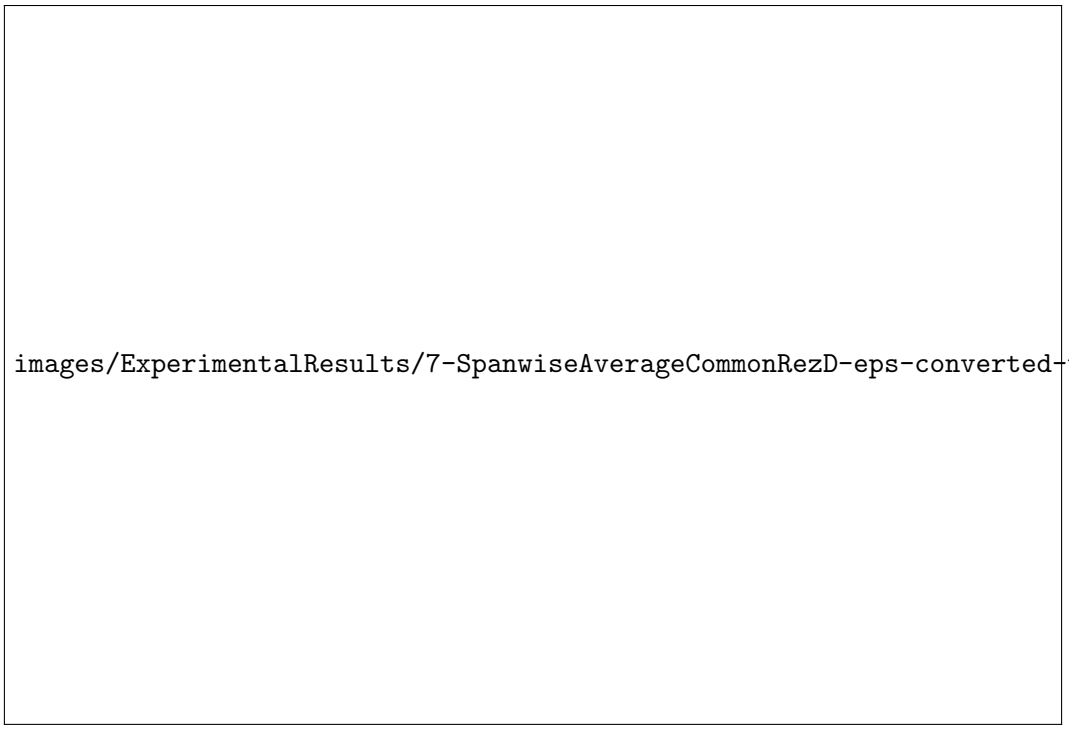


Figure 2.42: Re=12,000 - z/D=4 - Time and Span Averaged Nusselt Number - Experimental

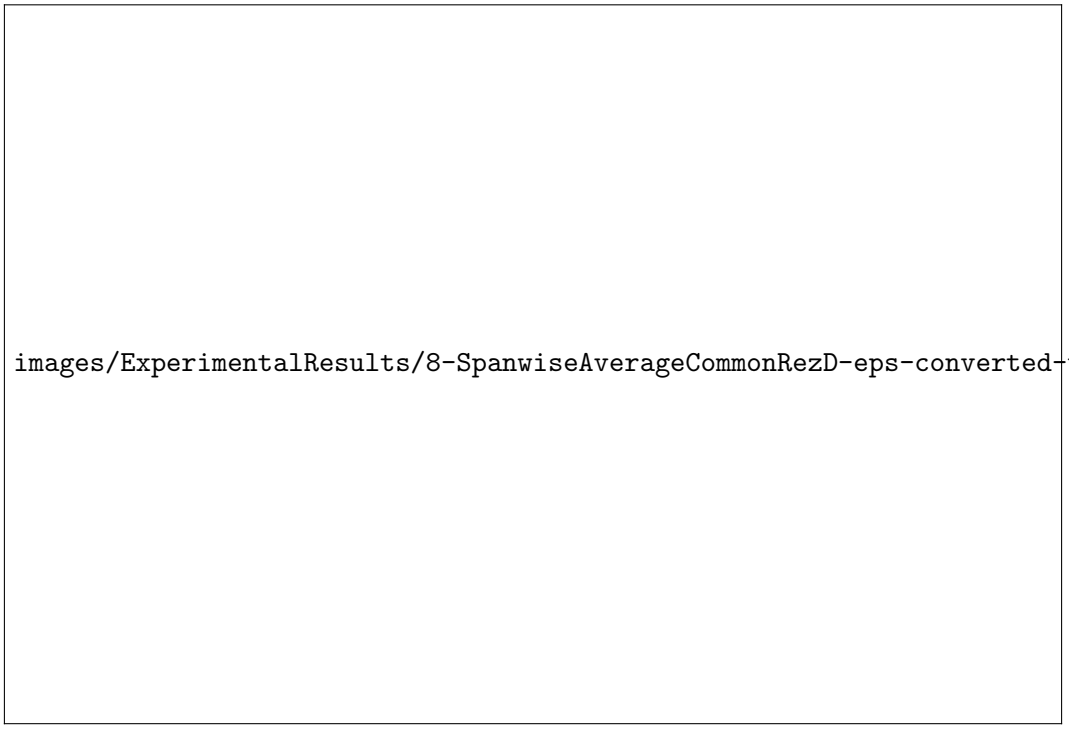


Figure 2.43: Re=15,000 - z/D=4 - Time and Span Averaged Nusselt Number - Experimental

2.4.3 Non-Dimensional Jet Standoff of z/D=5

Figures 2.44 through 2.57 show the time averaged Nusselt number and time and span averaged Nusselt number for a non-dimensional target spacing of z/D=5. As was with the non-dimensional target spacing of z/=4, the results appear qualitatively similar to cases with a closer target spacing. The cases with z/D=5 show a further reduction in time and span averaged Nusselt number when compared to the z/D=4 cases.

Double Exit - Time Averaged Nusselt Number

Figure 2.44: Re=4,000 - Double Exit - z/D=5 - Time Averaged Nusselt Number - Experimental

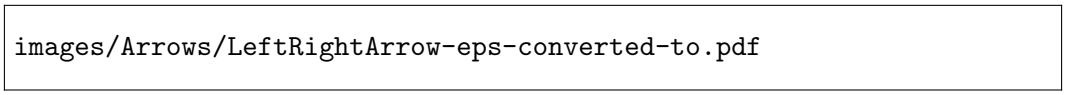


Figure 2.45: Re=8,000 - Double Exit - z/D=5 - Time Averaged Nusselt Number - Experimental

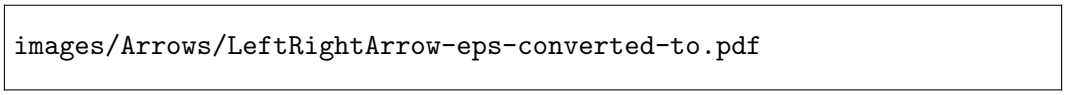


Figure 2.46: Re=12,000 - Double Exit - z/D=5 - Time Averaged Nusselt Number - Experimental

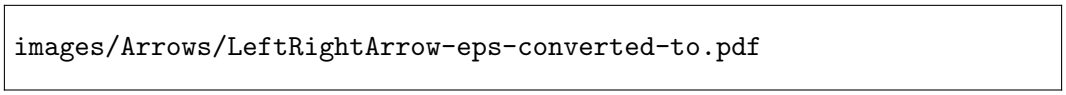


Figure 2.47: Re=15,000 - Double Exit - z/D=5 - Time Averaged Nusselt Number - Experimental

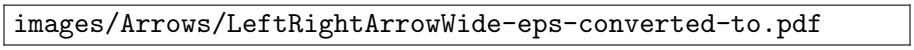


Figure 2.48: Double Exit - z/D=5 - Time and Span Averaged Nusselt Number - Experimental

Single Exit - Time Averaged Nusselt Number

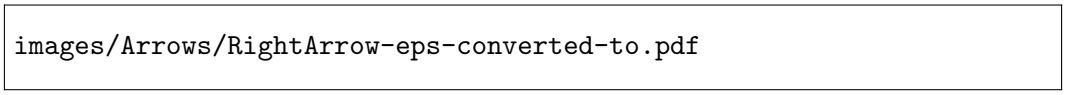


Figure 2.49: Re=4,000 - Single Exit - z/D=5 - Time Averaged Nusselt Number - Experimental

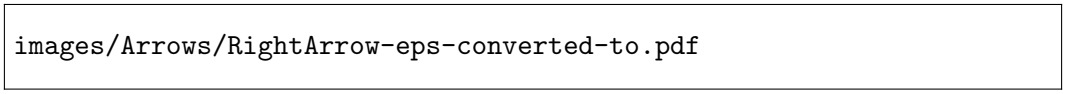


Figure 2.50: Re=8,000 - Single Exit - z/D=5 - Time Averaged Nusselt Number - Experimental

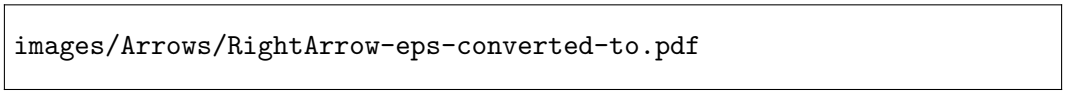


Figure 2.51: Re=12,000 - Single Exit - z/D=5 - Time Averaged Nusselt Number - Experimental

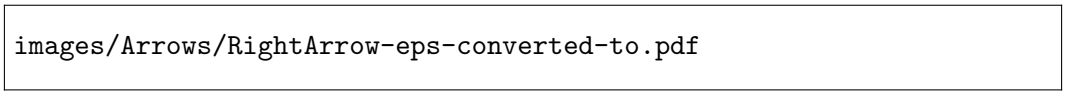
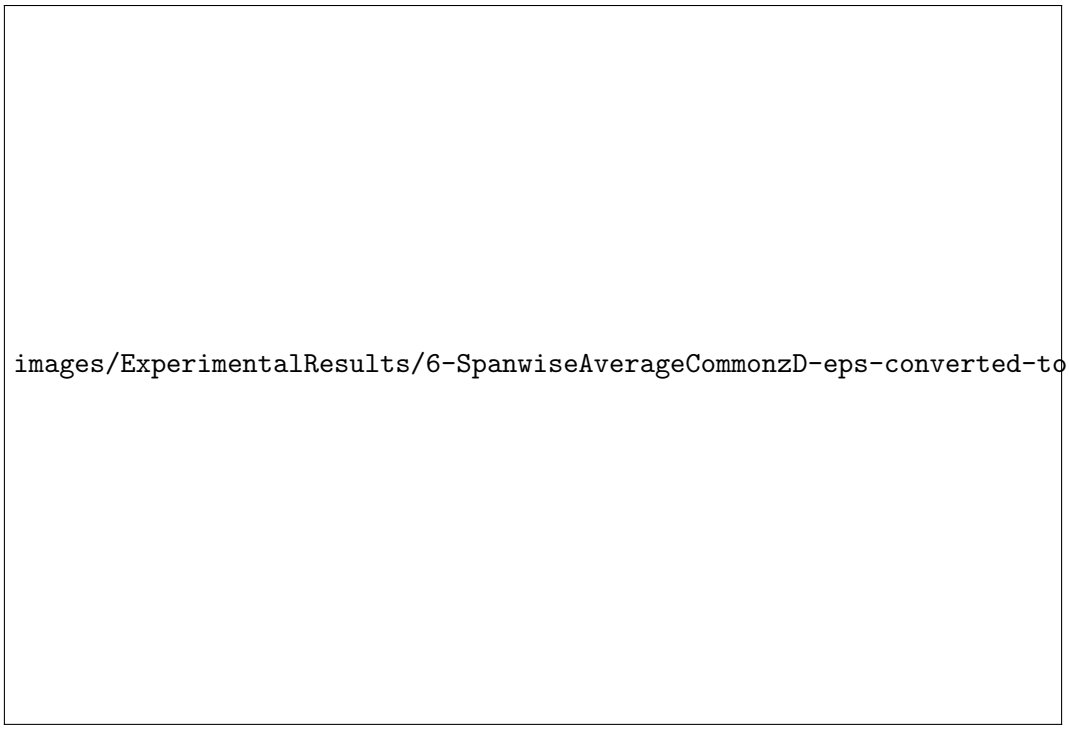


Figure 2.52: Re=15,000 - Single Exit - z/D=5 - Time Averaged Nusselt Number - Experimental



images/Arrows/RightArrowWide-eps-converted-to.pdf

Figure 2.53: Single Exit - z/D=5 - Time and Span Averaged Nusselt Number - Experimental

Single Exit vs Double Exit - Time and Span Averaged Nusselt Number

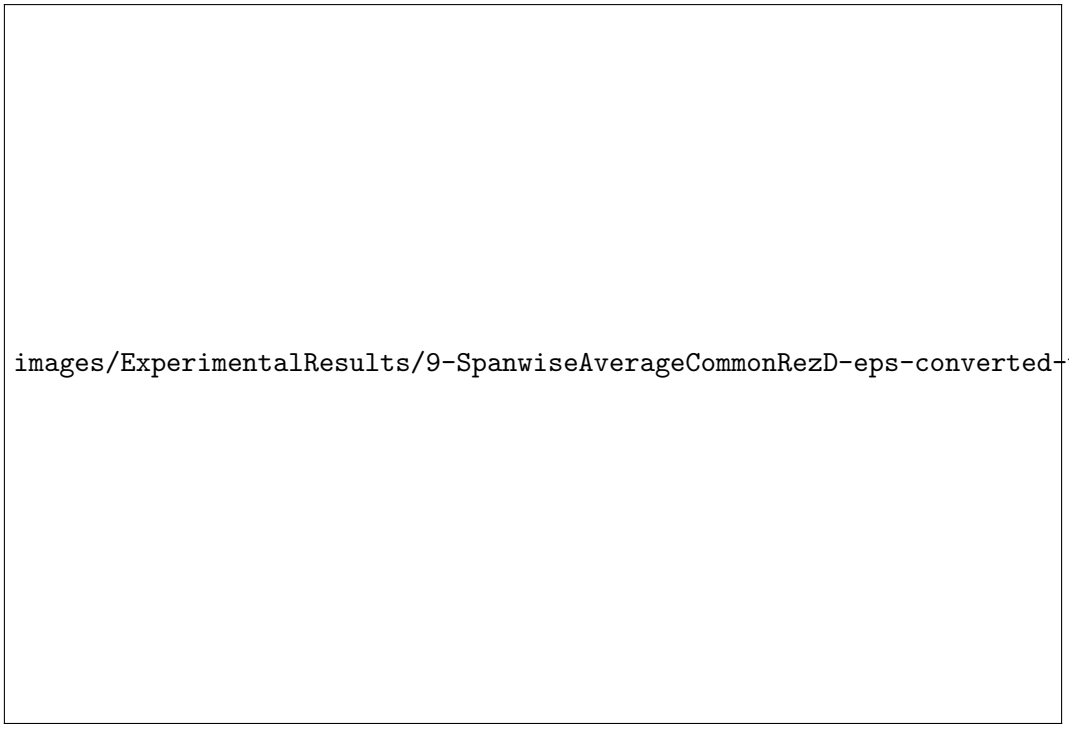


Figure 2.54: Re=4,000 - z/D=5 - Time and Span Averaged Nusselt Number - Experimental

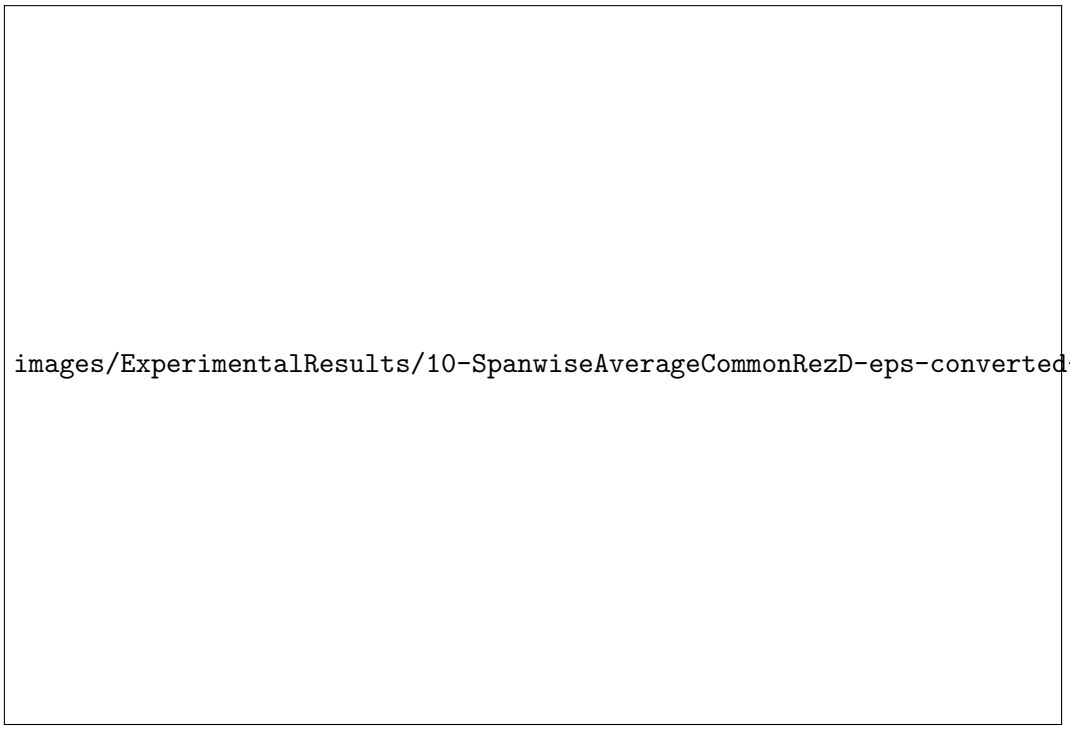


Figure 2.55: Re=8,000 - z/D=5 - Time and Span Averaged Nusselt Number - Experimental

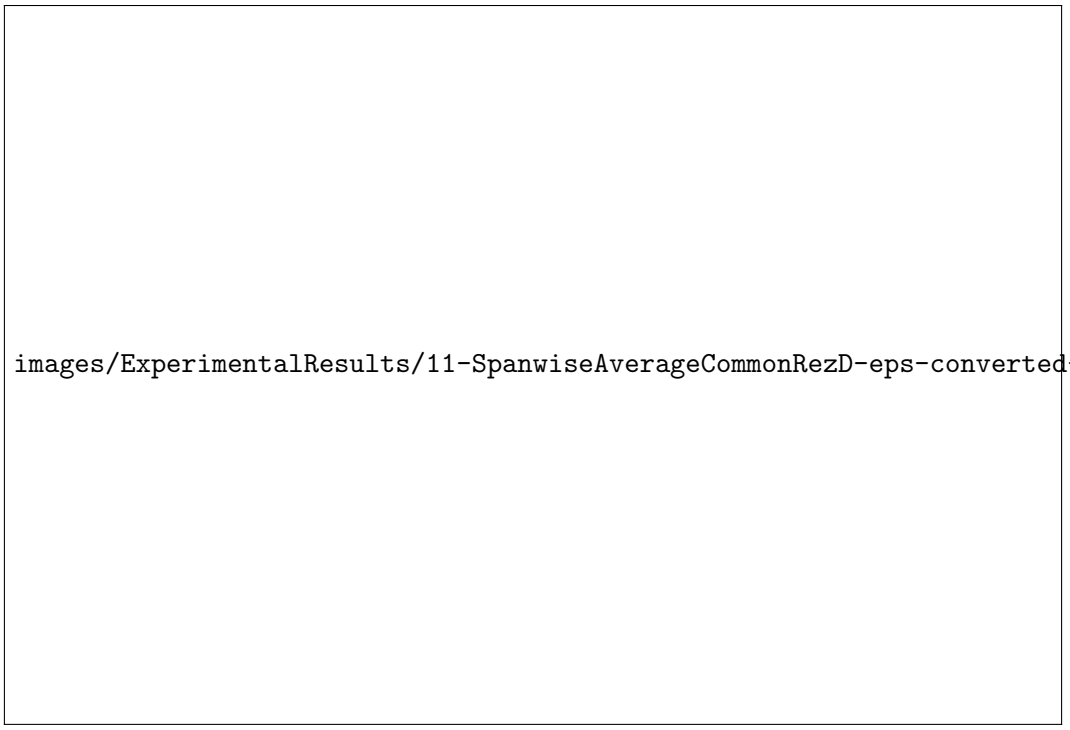


Figure 2.56: Re=12,000 - z/D=5 - Time and Span Averaged Nusselt Number - Experimental

Figure 2.57: Re=15,000 - z/D=5 - Time and Span Averaged Nusselt Number - Experimental

2.4.4 All Cases - Time and Area Averaged Nusselt Number

Figure 2.58 shows the time and overall surface area averaged Nusselt number (Nu_{mean}) for all experimental cases. Results are nearly linear with respect to jet Reynolds number. Time and area averaged Nusselt number is also dependent upon exit flow configuration and the target spacing, although more weakly than the jet Reynolds number within the ranges studied.

Figure 2.58: Experimental Results - Overall Averages

2.4.5 Agreement with Historical Results

Due to the unique nature of the geometry presently studied, no similar study has been identified in the public domain which could be used for detailed comparison of the results presented in Figures 2.14 to 2.58. As a result, only qualitative comparisons can be made. The spatial variation in Nusselt number seen in the present work shows qualitative similarities to studies presented by Huang et al., and Gao, among others, which studied the spatial variation in target surface Nusselt number[3, 14]. Historically, researchers have correlated the dependence of overall area averaged Nusselt number

with jet Reynolds number by

$$Nu_{mean} \propto Re_{jet}^{m} \tag{2.35}$$

where m is a coefficient dependent upon parameters such as the target spacing, hole pitch, and cross flow. Florschuetz et al. proposed a value of $m = .727$ as a reasonable value of m for a wide range of cases with minimal cross flow[15]. Within the narrow range of Reynolds numbers presently studied, the dependence upon Reynolds number is nearly linear, which is in reasonable agreement with the value of $m = .727$ proposed by Florschuetz et al.. Florschuetz et al., Kercher and Tabakoff, and Gao, among others, saw qualitatively similar effects on Nusselt number with changes in jet target spacing[1, 14, 15]. Huang et al. conducted a study, although geometrically different, varied the cross flow level by changing the number of exits, as was done in this study[3]. Their study exhibited qualitatively similar impacts on the overall area averaged Nusselt number when changing from a double exit to a single exit configuration.

Chapter 3

Numerical Study of Impingement Jet Heat Transfer

3.1 Computing Workflow

Preprocessing (mesh generation, case setup, and coarse grid simulations) was performed on a local workstation running the 64 bit Ubuntu 10.04 distribution of the Linux open source operating system. The system featured 12 gigabytes of random access memory and a quad core Intel i7 central processing unit operating at 2.8 gigahertz. The computational meshes were generated utilizing the commercial computational fluid dynamics (CFD) meshing software Pointwise, developed by Pointwise, Inc.. After mesh generation was complete, the mesh was exported to the Fluent case file format as an unstructured mesh. The ANSYS Fluent 12.1.2 CFD solver was used for all numerical simulations presented. Before each mesh could be used with the parallel Fluent solver, the meshes first had to be loaded using the serial version of Fluent, and re-saved. Although a simple task to perform, for larger meshes, the process required the use of a batch processing system in order to perform this conversion on a system with sufficient random access memory. An automated workflow was developed to transfer the mesh to the Hawk supercomputer at the Air Force Research Laboratory Department of Defense Supercomputing Resource Center (DSRC) (located at Wright Patterson Air Force Base in Dayton, Ohio) and schedule a job in the batch queuing system to perform this conversion process and place the mesh into the archival file system for long term

storage.

Once each mesh was converted to a format usable by the parallel version of Fluent, the solution process could then be initiated. An automated workflow was developed using MATLAB in order to generate the required batch submission scripts and Fluent journal file needed to conduct the simulation. The development of this MATLAB code did introduce some additional workload at the onset of this research, however, it was deemed necessary due to the cumbersome nature of the use of the batch queuing system, workspace file system, archive file system on the Hawk supercomputer, the extremely unreliable nature of the Fluent license server, as well as the size of the meshes being used and the number of CFD simulations that were conducted throughout the course of this study. In an effort to best imitate the flow conditions of the test rig, boundary conditions implemented in the numerical simulations were copied directly from the measured experimental test data files using the automated workflow. All other information related to the Fluent solver configuration, such as the mesh used, number of CPUs, run time, discretization schemes, etc. were derived from a case definition spreadsheet which the MATLAB script used as its input. In addition to the time savings achieved with this automated workflow, the case definition spreadsheet used to define solver settings for each CFD simulation also served as comprehensive documentation describing all simulations conducted. In addition to developing batch submission scripts and Fluent journal files for new CFD cases, the automated workflow was also able to continue a CFD simulation that was terminated either due to a system failure at the DSRC, or failure to request a long enough run time for the job. The source code for this automated workflow is included in Appendix A.3.1. With the appropriate batch submission scripts and Fluent journal files generated, these files could then be uploaded to the Hawk supercomputer and scheduled in the batch queuing system. The simulation then began at a later date as determined automatically by the batch queuing system based upon hardware and software license availability.

Once each simulation was complete, all files related to the run were copied to the archival file system at the DSRC for long term storage. Post processing was performed on the local workstation using MATLAB. The scripts generated for post processing are included in Appendix A. MATLAB was selected for post processing because of the vast mathematical functions available as well as its ability to tightly integrate the post processing of the numerical simulation results with the experimental results. Additionally, due the vast amount of data automation required in performing

time averaging as well as animating the solution data, the use of MATLAB exclusively was selected as the most efficient approach. In an effort to reduce the amount of data transferred, stored, and processed locally, solution data along two planes was exported to the ASCII format using Fluent. The first plane was the impingement jet target surface, and the second plane was at the centerline of the flow field (y=0). Unfortunately, the parallel version of Fluent does not have the capability to export solution data to ASCII format. As a result, all data had to be reread by the serial version of Fluent at a later time and then exported to ASCII format, rather than being exported during the solution process while all solution data was still in memory. This drawback increased the complexity of post processing as well as the post processing time dramatically. Because every time step had to be explicitly loaded in Fluent, and ASCII data exported, another MATLAB script was created to generate the required batch submission scripts and the corresponding Fluent journal file. The source code to the script can be found in Appendix A.4.1. With the appropriate batch submission scripts and Fluent journal files generated, these files were uploaded to the Hawk supercomputer and scheduled in the batch queuing system. The exporting then began at a later date as determined automatically by the batch queuing system based upon hardware and software license availability. Once this batch exporting process began, the ASCII data was also copied to the archival file system at the DSRC. When the process was complete, the ASCII data files were copied to the local workstation and post processed using MATLAB.

3.2 Steady Simulations of Single Impingement Jets and Single Impingement Jet Grid Dependency Study

In an effort to determine the baseline performance of a single jet and to validate the computational mesh topology developed, the numerical component of this study was initiated with a study of single impingement jets. A grid dependency study was conducted for a single jet configuration for jet Reynolds numbers of 4,000 and 15,000 and a non-dimensional jet target spacing (z/D) of 3. Due to the basic similarities in the flow physics found in multi jet configurations, a grid dependency study for a single jet configuration was deemed an acceptable technique which could be conducted with a significantly reduced computational expense.

3.2.1 Computational Mesh

In an effort to resolve flow all the way down to the viscous sublayer, the design of the baseline computational mesh targeted that the first grid point normal to each wall be located at a y^+ value of approximately 1. Initial estimates for the boundary layer point distribution were made through the use of experimental correlations for skin friction and boundary layer thickness of turbulent flow over a flat plate. Although the impingement jet nozzle was a cylinder rather than a flat plate, no correlations were identified for skin friction or boundary layer thickness in developing flow in a cylinder. Approximating the cylinder as a flat plate was a sufficient assumption for the grid point clustering process. Accordingly, y_L^+, the dimensionless distance at location L is defined as [16]

$$y_L^+ \equiv \frac{(u_{\tau,L})y_{N,L}}{\nu} \tag{3.1}$$

where ν is the kinematic viscosity of the fluid, $y_{N,L}$ is the distance normal to the surface at location L, and $u_{\tau,L}$ is the friction velocity at location L. The friction velocity, $u_{\tau,L}$, is defined as [16]

$$u_{\tau,L} \equiv \sqrt{\frac{\tau_{w,L}}{\rho_{fluid}}} \tag{3.2}$$

where $\tau_{w,L}$ is the local wall shear stress at location L. Substituting Equation (3.2) into Equation (3.1), setting $y_L^+ = 1$, and solving for $y_{N,L}$ yields

$$y_{N,L}\big|_{y^+=1} = \frac{\nu}{\sqrt{\frac{\tau_{w,L}}{\rho_{fluid}}}} \tag{3.3}$$

The local wall shear stress, $\tau_{w,L}$, was estimated using an experimentally determined correlation for local skin friction, $C_{f,L}$ of turbulent flow over a flat plate [17]

$$C_{f,L} = .0592 Re_L^{-1/5} \tag{3.4}$$

where Re_L is the Reynolds number based upon the distance, L, from the leading edge of the plate. The skin friction coefficient is defined in terms of the local wall shear stress ($\tau_{w,L}$), fluid density

(ρ_{fluid}), and free stream fluid velocity (U_∞) as

$$C_{f,L} \equiv \frac{\tau_{w,L}}{\rho_{fluid} U_\infty^2/2} \tag{3.5}$$

Combining equations (3.4) and (3.5) and solving for $\tau_{w,L}$ yields

$$\tau_{w,L} = .0592 Re_L^{-1/5} \rho_{fluid} U_\infty^2/2 \tag{3.6}$$

Combining equations (3.3) and (3.6) yields

$$y_{N,L}\big|_{y^+=1} = \frac{\nu}{\sqrt{.0592 Re_L^{-1/5} U_\infty^2/2}} \tag{3.7}$$

Equation (3.7) was evaluated at each jet Reynolds number (Re_{jet}) with location L equal to 10% of the total length of the nozzle (0.953mm) and the first grid point normal to the wall was placed at this location. At location L, and all downstream locations, flow should be resolved down to a y^+ value of approximately 1. The first axial grid point was placed at 10% of the total length of the nozzle. With this approach, the flow between the inlet to the nozzle and 10% of the nozzle length was not considered to be fully resolved. The choice of 10% of the nozzle length was based on engineering judgment. In the free stream, the grid point distribution was defined such that there were approximately 25 equally spaced grid points per mm, also based upon engineering judgment. At the outlet of the nozzle ($L = 9.525mm$), the boundary layer thickness (δ_L) was computed using an experimental determined correlation for boundary layer thickness of turbulent flow over a flat plate [17]

$$\delta_L = 0.37 L Re_L^{-1/5} \tag{3.8}$$

A single sided hyperbolic tangent interpolating function was used to distribute the grid points in the boundary layer based upon the initial clustering at the wall, final clustering at the edge of the boundary layer (which was equal to the free stream clustering of 25 grid points per mm), and the thickness of the boundary layer. Clustering at the outlet of the nozzle in the axial direction was defined such that the point spacing was two times the grid point spacing in the direction normal to the wall of the nozzle. A hyperbolic tangent interpolating function was also used to define the

axial grid point distribution in the nozzle. In the stagnation region, the same near wall normal grid point clustering was used in the stagnation region as was used normal to the wall in the nozzle. In the free stream, half way between the nozzle outlet and the impingement surface, the grid point distribution was defined to be 1 grid point per mm, based upon engineering judgment. All other grid point distributions were based upon engineering judgment. Single sided hyperbolic tangent interpolating functions were used to distribute points for all connectors (line segments) that had non uniform grid point spacing

A spreadsheet was developed to define the grid point count and length for all connectors in the computational mesh with equally spaced points, as well as the endpoint clustering and the total grid points on each connector in the mesh with non-uniform grid point spacing. The spreadsheet was also used to define a uniform refinement (rounded to the nearest grid point) of the mesh for each successive grid of the grid dependency study. For each successive grid used in the grid dependency study, the grid point density was increased by a multiple of 1.3, id est, the distance between grid points was successively reduced by a factor of 1/1.3. An alternative refinement scheme, which increases the number of grid points by a factor of 1.3 was explored, however, this approach was avoided due to the complexity in determining grid point spacing at the endpoints of the connectors when using the single sided hyperbolic tangent interpolating function. Both approaches produce similar refinement schemes when the number of grid points becomes large, however, changing the distance between the grid points by a factor of 1/1.3 rather than changing the number of grid points by a factor of 1.3 was selected due to an easier implementation of a uniform, systematic grid refinement scheme.

Utilizing this spreadsheet, a new mesh could be rapidly developed with appropriate grid point clustering for any jet Reynolds number. Due to the large number of grid points necessary in order to appropriately resolve the flow and heat transfer in impingement jets at high Reynolds numbers, a separate mesh was developed for each Reynolds number in order achieve appropriate clustering with minimum number of grid points. This technique is most suited for Reynolds Averaged Navier Stokes Equation (RANS) based models and for flows where high flow gradients are primarily due to boundary layers, rather than other flow phenomenona, such as massive flow separation. In the present study a RANS based solver was used, and all regions with high flow gradients were assumed to be associated with boundary layers, making this technique applicable.

Although the experimental test facility featured 55 impingement jets, rather than a single impingement jet, effort was made to design the computational mesh for the single jet case to replicate geometric features similar to that of the experimental facility. Accordingly, a cylindrical nozzle diameter of 4.7625mm was used. The length of the nozzle was 2 nozzle diameters. The jet standoff distance was defined to be 3 nozzle diameters. It should be noted that all dimensions in the test rig were assumed to be the nominal dimensions specified on all drawings which were provided to the manufacturer, all walls were assumed to be perfectly smooth, flat, and orthogonal to each other. The impingement jet nozzle was assumed to be a perfect circle. The heated foil impingement surface was considered to be of negligible thickness, and this thickness was not modeled in the numerical study.

The basic mesh topology consisted of that depicted in Figure 3.1. This topology represents 1/4 of an impingement jet nozzle and a heated region of the impingement jet target surface. This topology was then replicated, rotated, and extended to produce the mesh. The impingement jet target surface as well as the nozzle plate were square and had a length and width of 6 nozzle diameters. The computational mesh was generated using a block structured approach and was then converted to the unstructured Fluent case file format. Isometric, bottom, and side views of the computational mesh for the single jet case are shown in Figures 3.2, 3.3, and 3.4, respectively.

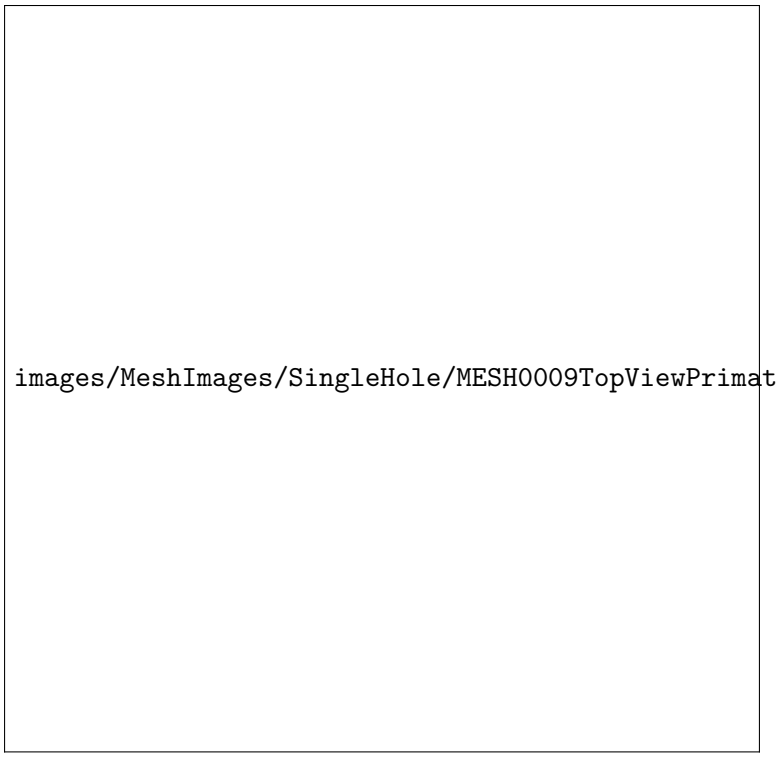


Figure 3.1: Basic Mesh Topology

Figure 3.2: Single Jet CFD Case - Isometric View of the Computational Mesh

Figure 3.3: Single Jet CFD Case - Bottom View of the Computational Mesh

3.2.2 Boundary Conditions

Due the the lack of a rigorous non-dimensionalization of all boundary conditions imposed in the test rig, boundary conditions imposed in the numerical simulations were based on dimensional values measured directly by the test rig instrumentation. Figures 3.3 and 3.4 also illustrate the boundary conditions imposed. When implementing the measured experimental boundary conditions in the numerical simulations, the same assumptions were made as when post processing the experimental data. All walls were assumed to maintain a no slip and adiabatic boundary condition, with the exception of the heated surface. The heated surface maintained a no slip condition with a constant surface heat flux, and no conduction. Although natural convection losses were identified and accounted for in the post processing of the experimental data, no natural convection was modeled numerically, and as a result, was not accounted for during the post processing of the numerical

data. The outlet boundary condition of the computational mesh was assumed to have a static gauge pressure of zero and a total temperature of 300K. The reference pressure was defined to be the atmospheric pressure that was measured during testing. The inlet temperature was assumed to be uniform, and the mass flow rate was defined to be constant. For the single jet cases, the constant mass flow rate at the inlet was defined based upon 1/55 of the area weighted average mass flow rate through all jets in the test rig. Upstream turbulence intensity was not measured in the test rig. A value of 0.7% was utilized for all simulations based upon the guidance of Dr. Shichuan Ou.

Figure 3.4: Single Jet CFD Case - Side View of the Computational Mesh

The length and width of the heated region of the single jet case was defined to be 3 nozzle diameters square, which was similar to the pitch used to distribute each of the 55 nozzles present in the experimental test rig. As shown in Figure 3.3, a region of an additional 1.5 nozzle diameters on all sides of the heated region was included in computational domain in order to ensure that the exit

boundary condition of zero static gauge pressure was valid at the exit of the computational domain, not causing any numerical instability, or affecting the accuracy of the entire flow field solution. This additional region was not heated because the convective heat transfer becomes very low far from the jet core, causing high temperature gradients which would have required an increased number of grid points in order to properly resolve the flow field without numerical instabilities.

For the single jet grid dependency study, a time independent flow configuration was desired. Boundaries on the mesh were defined such that there were four exits (i.e. the flow can exit in any direction), which results in steady flow because the flow does not interact with any sidewalls or neighboring jets. This steady flow was believed to have the highest heat transfer coefficient of any impingement jet configuration, and is useful because it also provides an estimate on the extent to which jet to jet interactions and cross flow interactions affect jet performance in an array of impingement jets.

3.2.3 Fluent Solver Configuration

Fluent's pressure based solver was used to solve the steady, 3D, compressible Reynolds Averaged Navier-Stokes (RANS) Equations without the inclusion gravity. As in the experimental facility, air was used as the working fluid in the numerical simulations of impingement jets. Although the flow Mach number was always less than 0.3, high impingement surface heat flux drove large temperature gradients in the flow, and as a result, significant fluid density variations existed, most prominently within the boundary layer. The ideal gas density model was selected in order to accommodate for the variable density of the working fluid. Additional fluid properties (thermal conductivity, viscosity, and specific heat) for air were derived from Fluent's integrated material property database and were assumed to be constant. In order to provide closure to the RANS Equations, the standard $k-\omega$ model was utilized.

3.2.4 Solution Initialization and Convergence

The flow field was initialized with a zero velocity and the temperature was initialized to that of the inlet temperature. Next, the mass flow rate boundary condition was activated. The solution began by solving the continuity, momentum and energy equations with a first order upwind spatial discretization scheme. No heat flux applied on the heated surface. After 50-500 iterations, the heat

flux was then applied. This technique was required to avoid the surface temperature from rising too quickly before the flow field was developed and before the impingement jet could begin to cool the heated surface. As mentioned in Section 2.2, the flow also had to be stabilized in the test rig before applying any heat flux, otherwise the facility would overheat. With the heat flux applied, the solution then progressed until the standard scaled residuals automatically computed by Fluent were less than 10^{-6}, for the x, y, and z momentum, as well as continuity, energy, k, and omega equations.

Several attempts were made to accelerate the convergence process. The first attempt initialized the velocity in the entire core of the jet to that of the free stream velocity, while leaving the rest of the flow field at zero velocity. This technique resulted in a slower convergence rate. The second attempt utilized a ramping of mass flow rate and heat flux boundary conditions up to the steady state conditions, rather than beginning the simulation with boundary conditions set at their steady state values. This technique resulted in virtually no acceleration in the convergence process and was disregarded due to the increased complexity in the solution process.

The solution obtained using first order accurate spatial discretization schemes was then used as an initial condition for a solution using second order accurate spatial discretization schemes. Several parameters were monitored in order to evaluate the convergence of the solution using second order accurate spatial discretization schemes. The standard scaled residuals automatically computed by Fluent (for the x, y, and z momentum, as well as continuity, energy, k, and omega) were monitored. The solution was assumed to be converged if all of the scaled residuals were less than 10^{-6}. If this was achieved, the solution was then converged for an additional 3,000 iterations in order to verify that the residuals continued to decrease and there was negligible change in the solution. For several of the coarser meshes, the default scaled residuals computed by Fluent could only be reduced to the order 10^{-3} with the present solution technique. For these cases, solution convergence was evaluated solely based upon monitoring changes in the solution. The maximum and average surface temperature were recorded and plotted during the convergence process. Animations of surface temperature distribution on the entire heated surface, as well as a temperature profile along the centerline of the heated surface were generated using images which were recorded every 10 iterations. The solution was assumed to be converged if, for the last 1500 iterations, the average surface temperature changed less than .025K, the maximum surface temperature changed less than

.0325K, and negligible temporal variation in surface temperature was observed when reviewing the animations of temperature vs iteration.

3.2.5 Post Processing Methodology

Post processing of the numerical simulation results for the single jet cases followed the same process as described in Section 2.3, with the exception that the term q''_{losses} was set to 0, and there was no variation in the results with time. The most basic MATLAB functions used to perform this processes can be found in Appendices A.4.5 and A.4.4. The top level script which generated all Figures in the remainder of Section 3.2 can be found in Appendix A.4.9.

3.2.6 Grid Dependency Study Results

A summary of all grid dependency study cases is presented in Table 3.1.

Jet Reynolds Number	Hole Spacing (x/D)	Hole Spacing (y/D)	Target Spacing (z/D)	Hole Pattern (x × y)	Exit Configuration	Grid Points (million)	Grid Point Spacing Multiple
4,000	N/A	N/A	3	1×1	Quadruple	1.2	$1/1.3^{-1}$
4,000	N/A	N/A	3	1×1	Quadruple	2.4	$1/1.3^{0}$ (baseline)
4,000	N/A	N/A	3	1×1	Quadruple	5.6	$1/1.3^{1}$
4,000	N/A	N/A	3	1×1	Quadruple	12.6	$1/1.3^{2}$
15,000	N/A	N/A	3	1×1	Quadruple	1.2	$1/1.3^{-1}$
15,000	N/A	N/A	3	1×1	Quadruple	2.6	$1/1.3^{0}$ (baseline)
15,000	N/A	N/A	3	1×1	Quadruple	5.9	$1/1.3^{1}$
15,000	N/A	N/A	3	1×1	Quadruple	13.2	$1/1.3^{2}$

Table 3.1: List of Grid Dependency Study Cases

Results from the grid dependency study at a jet Reynolds number of 4,000 are presented in Figures 3.5 to 3.10. Simulations were conducted using four different meshes. A baseline mesh was generated using the procedure laid out in Section 3.2.1. Two additional meshes were generated

by refining the baseline mesh, and the fourth mesh was generated by coarsening the baseline mesh. The spanwise average Nusselt number is shown in Figure 3.5. As is readily evident, all meshes yielded very similar results. The area weighted Nusselt number is presented in Figure 3.6. Although the variation with increase in grid points may appear large and erratic at first glance, it is important to note that the range of the vertical axis is very small. The absolute value of the percent change in area weighted average Nusselt number is presented in Figure 3.7. The absolute value of the percent change in the area weighted average solution is diminishing with increase in grid points, indicating the computational meshes compared are of sufficient distribution and point density. A more rigorous assessment is presented in Figure 3.8. In order to produce Figure 3.8, data for each grid refinement case was interpolated to a uniform, common, 100x100 grid. Next, a percent difference was calculated between each successive grid refinement case, for each of the uniform 100x100 grid point locations, and the absolute value was determined at each of these uniform grid point locations. The maximum absolute value was then located for each successive grid refinement case, and the results are presented in Figure 3.8. Figure 3.8 shows that even locally on the impingement jet target surface, the variation in surface Nusselt number is low, and the change with increasing grid points is diminishing.

Figure 3.5: Single Jet - Re=4,000 - z/D=3 - Spanwise Average Nusselt Number - Numerical

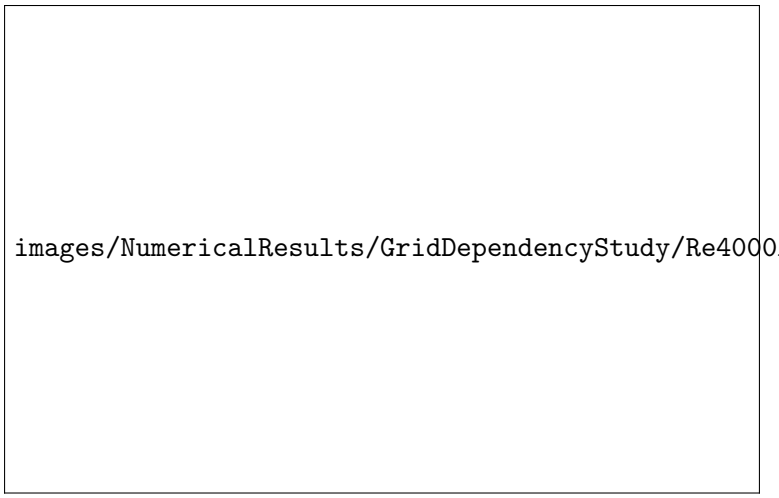


Figure 3.6: Single Jet - Re=4,000 - z/D=3 - Area Weighted Average Nusselt Number- Numerical

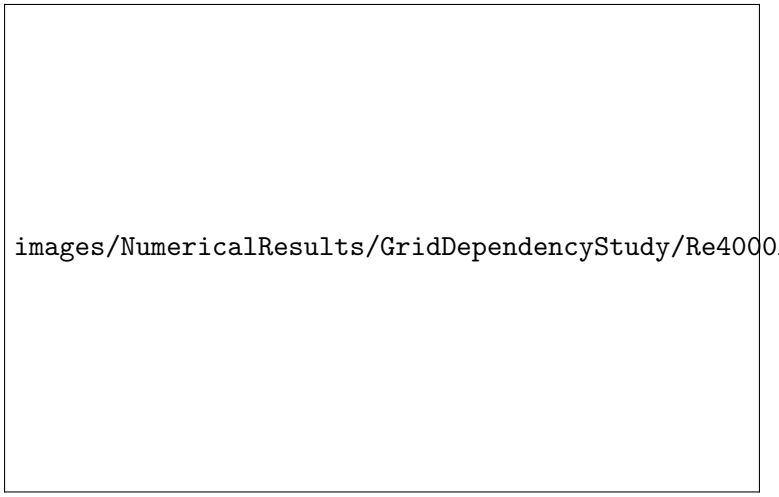


Figure 3.7: Single Jet - Re=4,000 - z/D=3 - Absolute Value of Percent Change in Area Weighted Average Nusselt Number - Numerical

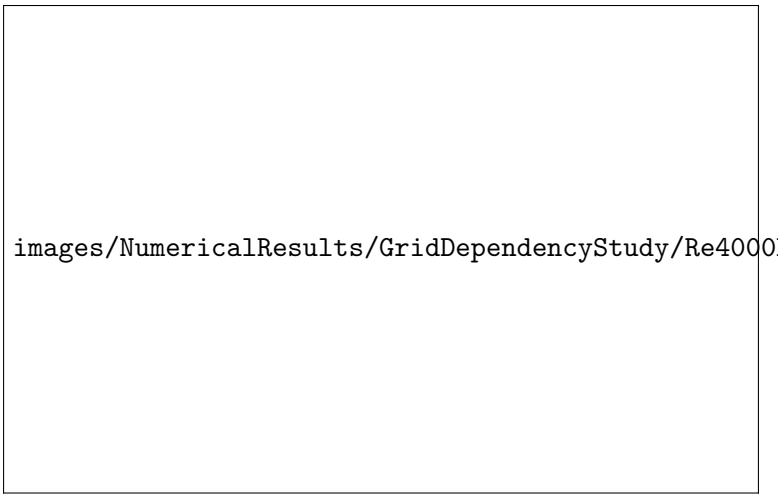


Figure 3.8: Single Jet - Re=4,000 - z/D=3 - Maximum Absolute Value of Percent Change in Local Nusselt Number - Numerical

Figures 3.9 and 3.10 show the y^+ value for the first grid point away from the target surface in the normal direction for the coarsest and finest meshes, respectively. With the goal for the first grid point normal to the target surface to be approximately at a y^+ value of 1, both meshes satisfied this requirement on the heated surface. Although not presented, in the impingement jet nozzle, y^+ values for the first grid point away from wall, in the normal direction were also of similar magnitude, except between the inlet of the nozzle, and the first axial grid point, a region which was not expected to be fully resolved to the viscous sublayer.

Figure 3.9: Single Jet - Re=4,000 - z/D=3 - y^+ Value for the First Grid Point Away From the Target Surface - Numerical (1.2 million grid points)

Figure 3.10: Single Jet - Re=4,000 - z/D=3 - y^+ Value for the First Grid Point Away From the Target Surface - Numerical (12.6 million grid points)

Figures 3.11 to 3.16 present results for the grid dependency study at a jet Reynolds number of 15,000 for the baseline mesh, coarsened mesh, and two refinements. As for the case with jet Reynolds number of 4,000, there was negligible change in the solution as the mesh was refined. Interestingly, the absolute value of the percent change in area weighted average Nusselt number increases with the number of grid points, as is evident in Figure 3.13, however, the absolute value of the percent change in local Nusselt number is still decreasing with an increase in grid points. This indicates that the solution is slowly changing locally in different directions, which can be observed in Figure 3.11, and that with an increase in grid points, there is a diminishing change in the local solution.

Figure 3.11: Single Jet - Re=15,000 - z/D=3 - Spanwise Average Nusselt Number - Numerical

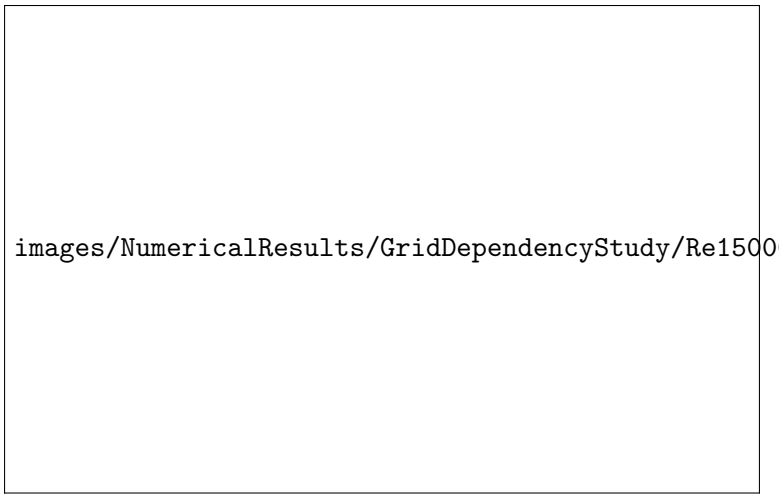


Figure 3.12: Single Jet - Re=15,000 - z/D=3 - Area Weighted Average Nusselt Number - Numerical

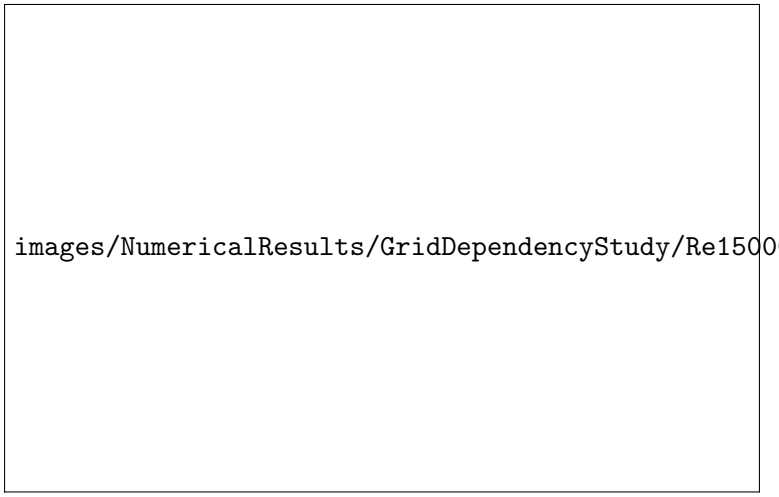


Figure 3.13: Single Jet - Re=15,000 - z/D=3 - Absolute Value of Percent Change in Area Weighted Average Nusselt Number - Numerical

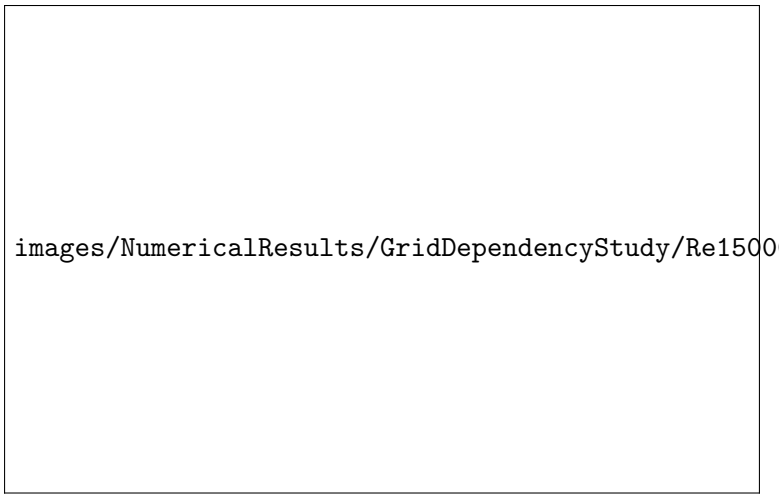


Figure 3.14: Single Jet - Re=15,000 - z/D=3 - Maximum Absolute Value of Percent Change in Local Nusselt Number - Numerical

Figure 3.15: Single Jet - Re=15,000 - z/D=3 - y^+ Value for the First Grid Point Away From the Target Surface - Numerical (1.2 million grid points)

Figure 3.16: Single Jet - Re=15,000 - z/D=3 - y^+ Value for the First Grid Point Away From the Target Surface - Numerical (13.2 million grid points)

As was shown in Figures 3.9 and 3.10, Figures 3.15 and 3.16 show that y^+ values for the first grid point away from the target surface in the normal direction for the coarsest and finest meshes are approximately 1 everywhere. In the impingement jet nozzle, y^+ values for the first grid point away from wall, in the normal direction were also not presented for cases with a jet Reynolds number of 15,000, however, they were also observed to be of similar magnitude, except between the inlet of the nozzle, and the first axial grid point, a region which was not expected to be fully resolved to the viscous sublayer.

With appropriate y^+ values, and negligible change in local solution as the mesh is refined for both jet Reynolds numbers studied, it has been shown that the the current mesh topology and grid point clustering is appropriate for simulating flow and heat transfer of impingement jets using a RANS based solver.

3.2.7 Single Jet Simulation Results at Re=4,000

Figures 3.17 through 3.22 present detailed results for the single jet case at a jet Reynolds number of 4,000 for the finest mesh, 12.6 million grid points. Figure 3.17 shows the spatial variation in

Nusselt number. For this single jet case with four exits, the variation in Nusselt number appears to be purely a function of radius. Figures 3.18 to 3.20 display the in plane velocity magnitude and velocity vectors on a plane taken about the centerline of the flow field. Figure 3.18 displays an overall velocity field, and Figures 3.19 and 3.20 show more densely placed velocity vectors in areas with boundary and free shear layers. It is interesting to observe that near the impingement surface, after the flow has turned, both a boundary layer and a free shear layer occur due to the high velocity fluid ("wall jet") that is traveling nearly parallel to the impingement surface.

Figure 3.17: Single Jet - Re=4,000 - z/D=3 - Nusselt Number - Numerical

Figure 3.18: Single Jet - Re=4,000 - z/D=3 - In Plane Velocity Magnitude and Velocity Vectors - Numerical

Figure 3.19: Single Jet - Re=4,000 - z/D=3 - In Plane Velocity Magnitude and Velocity Vectors (zoomed) - Numerical

Figure 3.20: Single Jet - Re=4,000 - z/D=3 - In Plane Velocity Magnitude and Velocity Vectors (zoomed) - Numerical

Figure 3.21 shows the normal component of vorticity about the plane at the centerline of the flow field. As can be seen, the flow is irrotational within the core of the jet, and in the free stream of the impingement region. Within the free shear layers and the boundary layers, the flow is highly rotational. The scale was saturated at $\pm 2500/s$ in order to make the vorticity throughout the entire flow field interpretable. The static temperature has also been plotted about this plane at the centerline of the flow and is shown in Figure 3.22. There is little variation between the inlet temperature and the free stream temperature in the impingement region. There are, however, high temperature gradients within the boundary layer on the impingement surface. These high temperature gradients are what mandated the use of the compressible Navier-Stokes equations.

Figure 3.21: Single Jet - Re=4,000 - z/D=3 - Normal Component of Vorticity - Numerical

Figure 3.22: Single Jet - Re=4,000 - z/D=3 - Static Temperature - Numerical

3.2.8 Single Jet Simulation Results at Re=15,000

Additional results for the single Jet case at a jet Reynolds number of 15,000 are presented in Figures 3.23 to 3.28 for the finest mesh, 13.2 million grid points. The results appear qualitatively similar to those presented in Figures 3.17 to 3.22.

Figure 3.23: Single Jet - Re=15,000 - z/D=3 - Nusselt Number - Numerical

Figure 3.24: Single Jet - Re=15,000 - z/D=3 - In Plane Velocity Magnitude and Velocity Vectors - Numerical

Figure 3.25: Single Jet - Re=15,000 - z/D=3 - In Plane Velocity Magnitude and Velocity Vectors (zoomed) - Numerical

Figure 3.26: Single Jet - Re=15,000 - z/D=3 - In Plane Velocity Magnitude and Velocity Vectors (zoomed) - Numerical

Figure 3.27: Single Jet - Re=15,000 - z/D=3 - Normal Component of Vorticity - Numerical

Figure 3.28: Single Jet - Re=15,000 - z/D=3 - Static Temperature - Numerical

3.3 Unsteady Simulations of Single Impingiment Jets with a Repeating Boundary Condition

In an effort to study jet to jet interactions without cross flow, two simulations were conducted of a single jet with a pair of repeating boundaries at Re=4,000, rather than four exits, as was described

in Section 3.2. The two repeating boundaries served to simulate an infinite row of impingement jet nozzles. This configuration will be referred to as "Single Jet Repeating". A summary of key parameters of each case is presented in Table 3.2.

Jet Reynolds Number	Hole Spacing (x/D)	Hole Spacing (y/D)	Target Spacing (z/D)	Hole Pattern (x × y)	Exit Configuration	Sample Time (s)	Grid Points (million)	Case Number
4,000	N/A	6 ("virtual")	3	1×∞	Double	11	2.4	I
4,000	N/A	3 ("virtual")	3	1×∞	Double	11	1.9	II

Table 3.2: List of Single Jet Repeating CFD Cases

3.3.1 Single Jet Repeating Case I: Computational Mesh, Boundary Conditions, Solver Settings, Solution Initialization and Solution Convergence

The computational mesh for the Single Jet Repeating Case I was identical to the baseline mesh that was used for the Re=4,000 in Section 3.2, except the boundary conditions at two of the outlets were changed to a pair of repeating boundaries. Figure 3.29 illustrates this. All other boundary conditions were identical to those imposed for the Re=4,000 case in Section 3.2. The solution initialization process was identical as well. After 5,000 iterations were conducted with a second order spatial discretization scheme, it was identified that this flow configuration was indeed physically unsteady. As a result, the simulation was changed from a steady state approach to a time accurate approach with a second order temporal discretization scheme. A time step of 0.022 seconds was used. Fluent's iterative time advancement scheme was utilized and 30 sub iterations were conducted for every time step. The solution progressed for 727 time steps, or 16 physical seconds. Solution data for the entire flow field was saved for each time step. During the solution process, the standard scaled residuals automatically computed by Fluent for the x, y, and z momentum, as well as continuity, energy, k, and omega equations were monitored and were reduced to an order of 10^{-3}. Average surface temperature was monitored. After 226 time steps (4.972 physical seconds), the average surface temperature was observed to have changed less

than 1.25K over the last 100 time steps (2.2 physical seconds), and was observed to be changing in primarily a cyclical manner, indicating that all startup transients effects due to the initialization of the flow with zero velocity should have negligible impact on the flow field after time step 226. Additionally, the velocity in the core of the jet was of order 15m/s, which would indicate that a flow particle would travel approximately 75 meters over the 4.972 second interval, a distance significantly larger than the distance from the inlet of the nozzle to the exhaust boundaries modeled in the computational mesh. Also indicating that any startup transient effects should have traveled to the exhaust boundaries by time step 226. All results presented used data from time steps number 227 to 727 (11 physical seconds), and have set t=0 seconds to refer to time step number 227.

Figure 3.29: Single Jet Repeating Case I - Bottom View of the Computational Mesh

3.3.2 Single Jet Repeating Case II: Computational Mesh, Boundary Conditions, Solver Settings, Solution Initialization and Solution Convergence

The computational mesh for the Single Jet Repeating Case II is similar to that of the Single Jet Repeating Case I except the repeating boundaries were moved to the edge of the heated region (closer to the impingement jet nozzle). The resultant mesh included 1.9 million grid points. This

change was equivalent to moving the impingement jet nozzles in this infinite row closer together. Figures 3.30 and 3.31 illustrate the computational domain setup for this case. The only other difference in the solution process for this case from the Single Jet Repeating Case I was that no second order steady solution was attempted because significant unsteadiness was observed with this configuration even with first order spatial discretization schemes and the steady state solver. The solution procedure transitioned directly from a steady state solution technique with first order accurate spatial discretization to an unsteady solution technique with second order accurate spatial and temporal discretization schemes after 36,960 iterations using the first order spatial discretization schemes. As with the Single Jet Repeating Case I, a time step of 0.022 seconds was used and Fluent's iterative time advancement scheme was utilized with 30 sub iterations conducted for every time step. Similar to the Single Jet Repeating Case I, all results presented used data from time steps number 227 to 727, and have set t=0 seconds to refer to time step number 227.

Figure 3.30: Single Jet Repeating Case II - Bottom View of the Computational Mesh

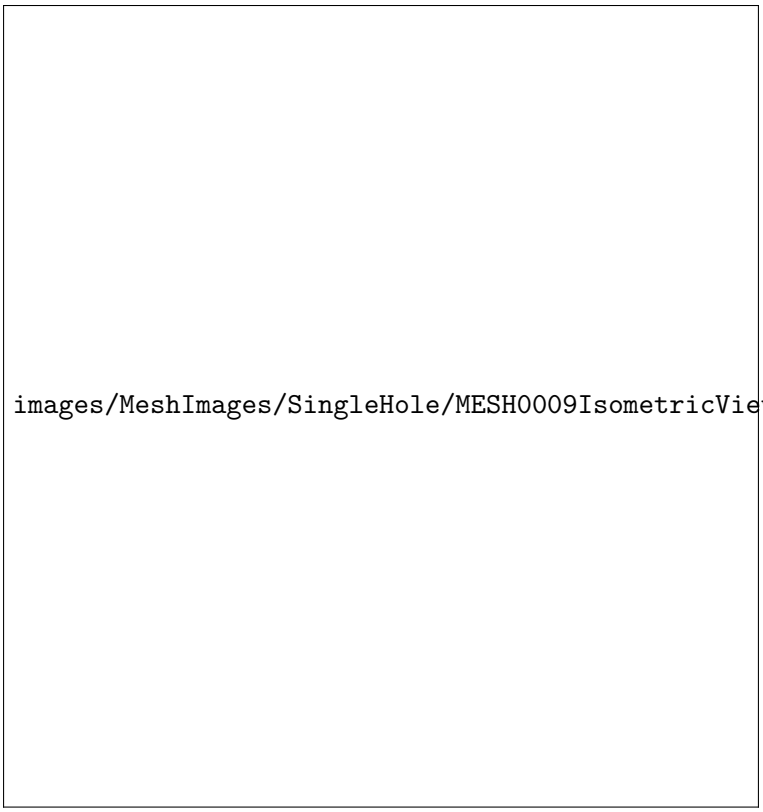


Figure 3.31: Single Jet Repeating Case II - Isometric View of the Computational Mesh

3.3.3 Post Processing Methodology

Post processing of the numerical simulation results for the single jet repeating cases followed the same process as described in Section 2.3, with the exception that the term q''_{losses} was set to 0. The most basic MATLAB functions used to perform this processes can be found in Appendices A.4.5 and A.4.4. The top level scripts which generated all Figures in the remainder of Sections 3.3.4 and 3.3.5 can be found in Appendices A.4.10 and A.4.11.

3.3.4 Single Jet Repeating Case I Results - Re=4,000, y/D=6

Results for the single jet case at a jet Reynolds number of 4,000, a non-dimensional target spacing of z/D=3, and repeating boundary conditions spaced at a non-dimensional distance of y/D=6 are presented in Figures 3.32 to 3.37. The time averaged Nusselt number, presented in Figure 3.32, appears to be a function of not only radius, but also weakly a function of angular position. An animation of the unsteady Nusselt number is presented in Figure 3.33. It is evident that significant

unsteadiness was stimulated by changing two of the exits to a set of repeating boundary conditions. Time averaged in plane velocity magnitude and normal component of vorticity are presented about a plane at the centerline of the flow (y=0), parallel to the repeating boundaries, in Figures 3.34 and 3.36, respectively. When comparing with Figures 3.18 and 3.21, the results appear qualitatively similar. However, when reviewing the animations of the unsteady in plane velocity magnitude and normal component of vorticity shown in Figures 3.35 and 3.37, a wobbling of the jet core can be observed. Results for time and spanwise average Nusselt number will be presented and discussed in Section 3.5.

Figure 3.32: Single Jet Repeating Case I - Re=4,000 - z/D=3 - Time-Averaged Nusselt Number

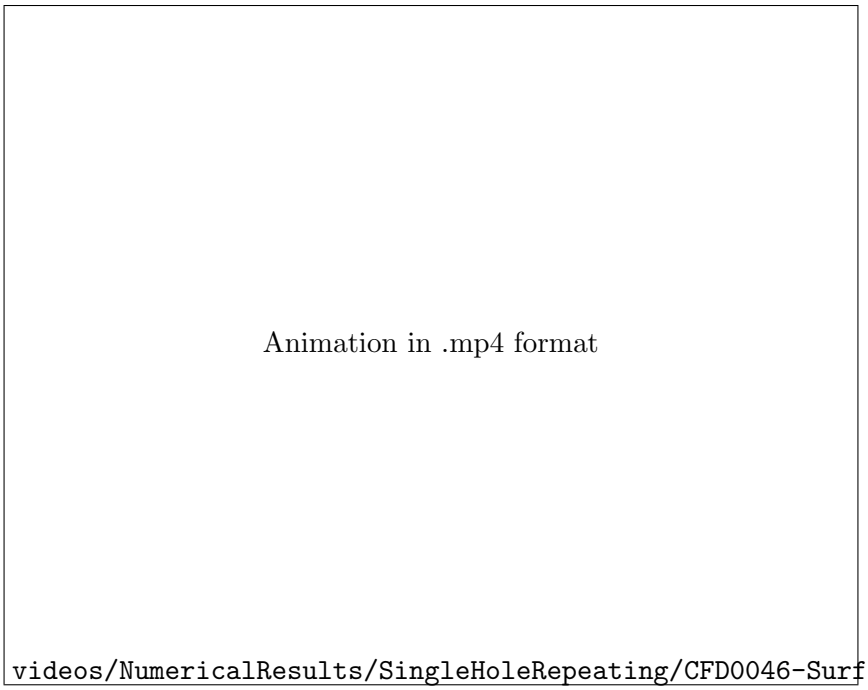


Figure 3.33: Single Jet Repeating Case I - Re=4,000 - z/D=3 - Animation of Unsteady Nusselt Number

Figure 3.34: Single Jet Repeating Case I - Re=4,000 - z/D=3 - Time-Averaged In Plane Velocity Magnitude

Figure 3.35: Single Jet Repeating Case I - Re=4,000 - z/D=3 - Animation of Unsteady in Plane Velocity Magnitude at the Center Line of the Flow

Figure 3.36: Single Jet Repeating Case I - Re=4,000 - z/D=3 - Time-Averaged Normal Component of Vorticity

Figure 3.37: Single Jet Repeating Case I - Re=4,000 - z/D=3 - Animation of Unsteady Normal Component of Vorticity at the Center Line of the Flow

3.3.5 Single Jet Repeating Case II Results - Re=4,000, y/D=3

Results for the single jet case at a jet Reynolds number of 4,000, a non-dimensional target spacing of z/D=3, and repeating boundary conditions spaced at a non-dimensional distance of y/D=3 are presented in Figures 3.38 to 3.43. The time averaged Nusselt number, presented in Figure 3.32 appears to be a function of not only radius, but also a strong a function of angular position, a significant difference between the single jet case with four exits and the single jet case with repeating boundaries placed at y/D=6. The unsteadiness which can be observed in Figure 3.39 is significantly more intense than that which was presented in Figure 3.33. Near the repeating boundaries there is a region with significantly lower convective heat transfer. This is due to the stagnation flow that occurs when the jet interacts with the nearby "virtual" jets, which are simulated by the repeating boundaries. When observing the time averaged in plane velocity magnitude (Figure 3.40) as well as the normal component of vorticity (Figure 3.42), the core of the jet appears to be weakened and widened. Animations of unsteady in plane velocity magnitude and normal component of vorticity, in Figures 3.41 and 3.43, respectively, show a wobbling of the jet with increased intensity when compared to Figures 3.35 and 3.37. Results for time and spanwise average Nusselt number will also be presented and discussed in Section 3.5.

Figure 3.38: Single Jet Repeating Case II - Re=4,000 - z/D=3 - Time-Averaged Nusselt Number

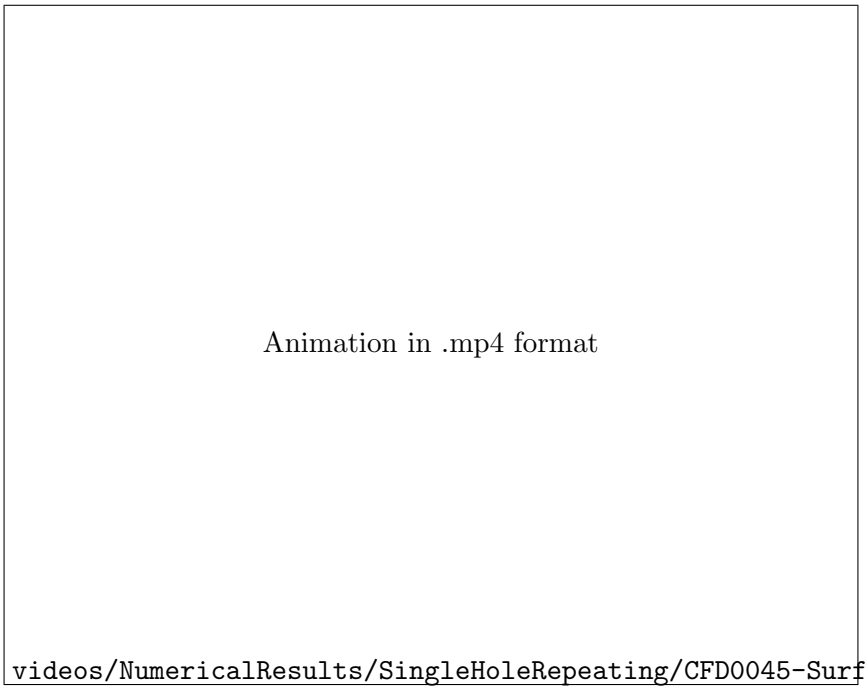


Figure 3.39: Single Jet Repeating Case II - Re=4,000 - z/D=3 - Animation of Unsteady Nusselt Number

Figure 3.40: Single Jet Repeating Case II - Re=4,000 - z/D=3 - Time-Averaged In Plane VelocityMagnitude

Figure 3.41: Single Jet Repeating Case II - Re=4,000 - z/D=3 - Animation of Unsteady in Plane Velocity Magnitude at the Center Line of the Flow

Figure 3.42: Single Jet Repeating Case II - Re=4,000 - z/D=3 - Time-Averaged Normal Component of Vorticity

Figure 3.43: Single Jet Repeating Case II - Re=4,000 - z/D=3 - Animation of Unsteady Normal Component of Vorticity at the Center Line of the Flow

3.4 Unsteady Simulations of Eleven Impingement Jets with a Repeating Boundary Condition

It was shown in Section 3.2 that the number of grid points required in order to accurately predict Nusselt number was considerably large. It was also shown in Sections 2.4.1 and 3.3 that impingement jet flow becomes unsteady when jets are forced to interact with each other. The experimental test rig was designed with 55 impingement jets. The rows of jets in the experimental facility were equally spaced, however, the spacing between the outer rows and the sidewalls did not match the pitch of the impingement jet array spacing (as was described in Section 2.1.1). With this difference in spacing between the holes and the sidewalls, the flow geometry did not facilitate any approximate simplifications through the use of repeating boundary conditions. Modeling a single row of holes with the same spacing as the test rig, with a repeating boundary condition does not account for the extra space between the outer rows and the sidewalls. This extra space in the test rig allows for cross flow air to bypass the impingement jets and migrate towards the exits without impacting

the performance of the downstream jets.

With the large mesh requirements for modeling a complete 55 jet case comes corresponding computing demands. Fluent software license availability at the DSRC was limited due to the popularity of the application which resulted in wait times of over one week before most large simulations could even begin executing. In addition, the Fluent license server and the DSRC resource checking proved to be extremely unreliable, resulting in effective wait times of many simulations to be beyond two weeks. This would occur because the simulation would attempt to begin running when no Fluent license servers would be available, resulting in the execution of the simulation being immediately terminated. When this would occur, the job would have to be re-queued and would have to wait in the queuing system again. Many times a job would be running for days, and then terminate due to an unexpected failure of the license server. When this happened, the failed case would have to be archived, and a new job initiated, which would typically require many more days/weeks of wait time, before the new job could start. In addition to these technical issues with the DSRC, the cumbersome nature of of the Fluent application greatly increased the cycle time for completing simulations with large meshes. As was mentioned in Section 3.1, many functions did not work while the Fluent application was running in parallel mode. The Fluent application would have to be relaunched in serial mode in order to perform these functions, greatly increasing the time to complete preprocessing, simulation, and post processing. With all of these challenges, simulations of only 11 impingement jets could be accomplished. Due to the fact that the flow geometry could not be modified to a smaller, but still representative flow geometry for numerical simulations, two 11 jet cases were simulated, in an attempt to simulate two possible extreme cases of cross flow, and jet to jet interactions. The first case incorporated repeating boundary conditions spaced at a non-dimensional distance of y/D=4.03, and the second incorporated repeating boundary conditions spaced at a non-dimensional distance of y/D=3. Both cases were conducted for a jet Reynolds number of 4,000, and a nozzle to target spacing of z/D=3. A summary of key parameters of each case is presented in Table 3.3.

Jet Reynolds Number	Hole Spacing (x/D)	Hole Spacing (y/D)	Target Spacing (z/D)	Hole Pattern (x × y)	Exit Configuration	Sample Time (s)	Grid Points (million)	Case Number
4,000	3	4.03 ("virtual")	3	11×∞	Double	11	16.9	I
4,000	3	3 ("virtual")	3	11×∞	Double	11	14.3	II

Table 3.3: List of Eleven Jet Repeating CFD Cases

3.4.1 Computational Mesh, Boundary Conditions, Solver Settings, Solution Initialization, Solution Convergence, and Post Processing

For cases with multiple jets, a fixed, equal mass flow rate could not be applied to the inlet of each nozzle, as was applied to the single jet cases. In the case of any real application of impingement jets, as well as the experimental test rig that was developed, all nozzles will be placed at approximately the same inlet pressure. Static gage pressure was measured inside of the pressure chamber in the experimental test rig. However, as was mentioned in Section 2.1.6, the static gage pressure within the pressure transducer was very low, on the order of 340Pa for the Re=4,000 case. During the design of the test rig, a low static pressure was not originally anticipated and a pressure transducer was installed which was calibrated with a full scale output of 17.24kPa. Due to this low quality measurement, an alternate technique was required in order to incorporate the inlet boundary condition.

A computational mesh was developed to also include a pressure chamber. When comparing Figures 2.4 and 3.46, it is evident that the cross section of the pressure chamber was slightly different than was used in the experimental test rig, however, the geometric differences were not anticipated to effect the results significantly due to low flow gradients expected in the pressure chamber. Modeling the pressure chamber facilitated capturing entrance effects in the impingement jet nozzles, and also allowed for the use of the mass flow rate measurement made with the venturi nozzle. With 11 nozzles, the mass flow rate was defined at the inlet of the pressure chamber to be 1/5 of the mass flow rate measured through the venturi nozzle for all 55 nozzles present in the test rig (rather than 1/55 which was used for the single jet cases). With a constant mass flow rate boundary condition defined across the inlet of the entire pressure chamber, the pressure inside

of the pressure chamber was approximately uniform throughout. This allowed for the numerical solution to automatically compute the variation in mass flow rate through the individual nozzles.

As well as the addition of the pressure chamber, the 11 jet computational mesh incorporated a longer unheated exit section when compared to the single jet cases, which matched the length of the exit section in the experimental test rig. The mesh topology and point clustering in the 11 jet case was based upon the 1.2 million grid point mesh for the single jet case at Re=4,000. Although this was the coarsest mesh used in the grid dependency study, it was selected in an effort to obtain a solution of reasonable fidelity, while minimizing all technical issues discussed when using Fluent at the DSRC. The grid point distribution in the pressure chamber and the exit sections were defined based upon engineering judgment and were clustered to avoid any discontinuities with the remainder of the mesh. In the pressure chamber upstream of the impingement jet nozzle inlets, a course grid point distribution was used due to the very low flow gradients expected. Due to the unsteady nature of the flow, the entire row of 11 nozzles was modeled, rather than implementing a symmetric boundary condition at the centerline. The case with the repeating boundaries placed at a non-dimensional distance of y/D=4.03 included 16.9 million grid points. The case with the repeating boundaries placed at a non-dimensional distance of y/D=3 included 14.3 million grid points.

Other than the changes in geometry mentioned, the solver configuration, solution process, judgment of solution convergence, and post processing were identical to that of the Single Jet Repeating Case II described in Section 3.3.2 except only 5,000 iterations were conducted with the first order spatial discretization schemes and the variation in average surface temperature from time step 126 to 226 was seen to vary up to 3K, rather than the 1.25K seen in Single Jet Repeating Cases I and II. As with Single Jet Repeating Cases I and II, all results presented used data from time steps number 227 to 727 (11 physical seconds), and have set t=0 seconds to refer to time step number 227.

Figure 3.44: Eleven Jet Repeating - Isometric View of the Computational Mesh

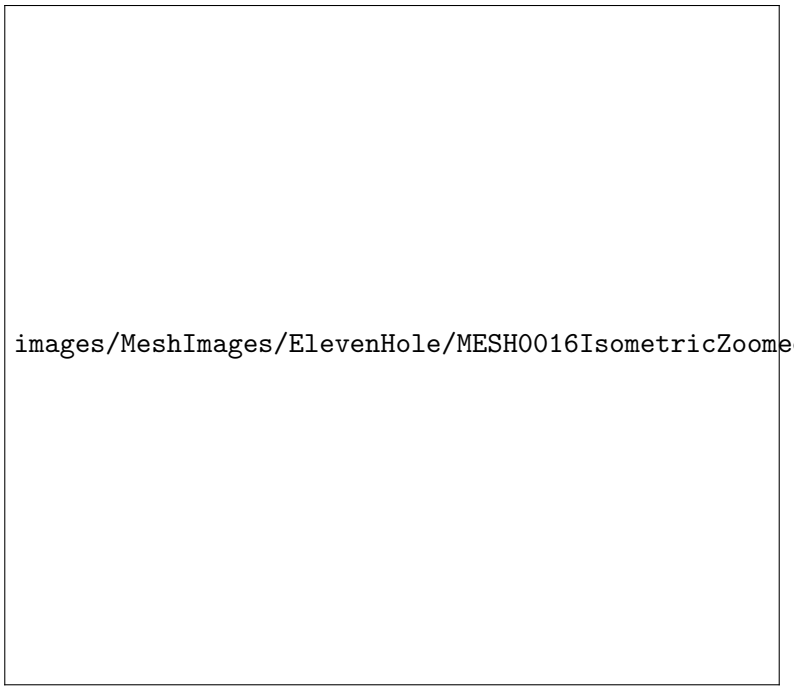


Figure 3.45: Eleven Jet Repeating - Isometric View of the Computational Mesh - Close Up

Figure 3.46: Eleven Jet Repeating - Side View of the Computational Mesh

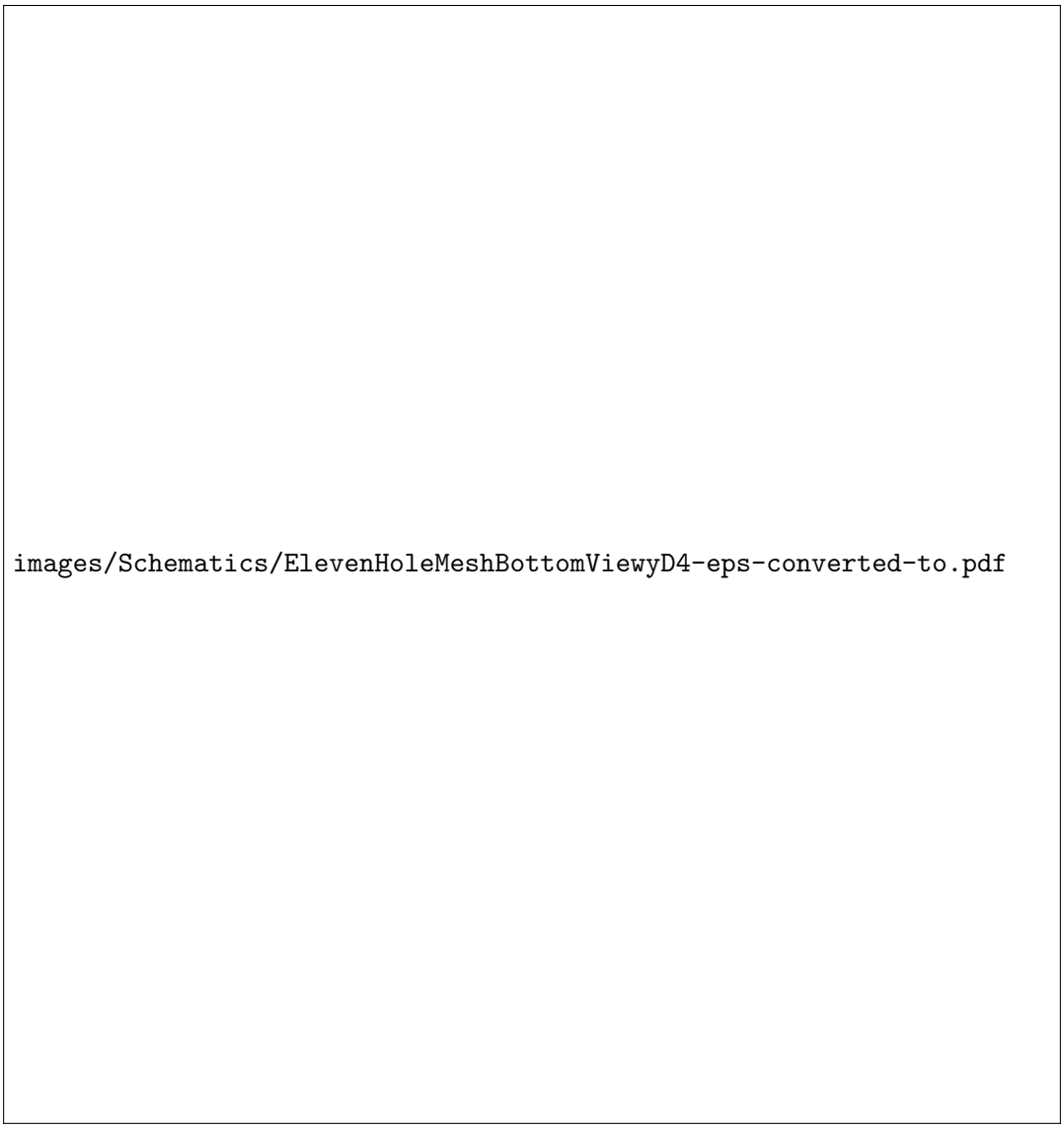


Figure 3.47: Eleven Jet Repeating Case I - y/D=4.03 - Bottom View of the Computational Mesh

Figure 3.48: Eleven Jet Repeating Case I - y/D=4.03 - Bottom View of the Computational Mesh (zoomed)

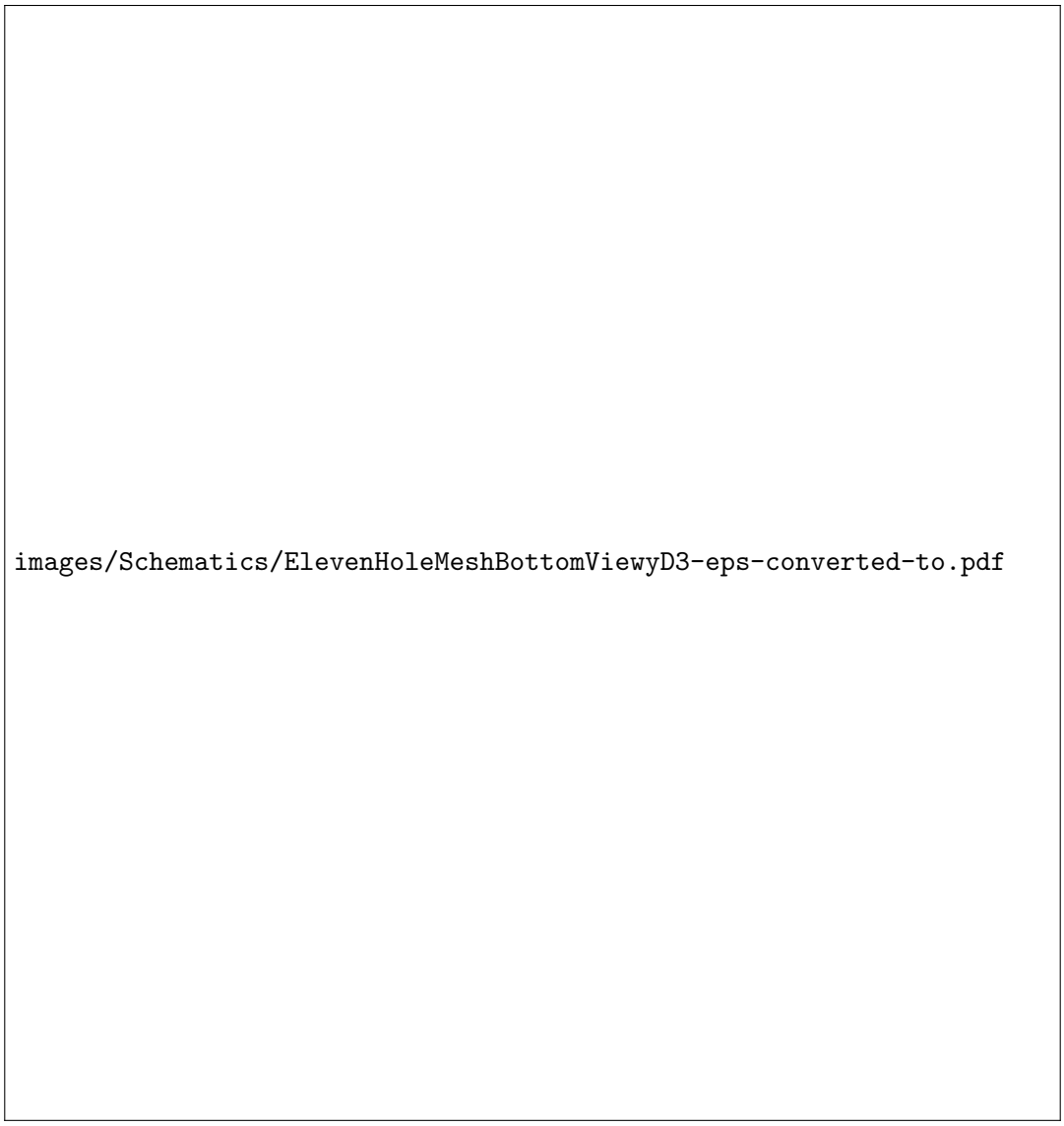


Figure 3.49: Eleven Jet Repeating Case II - y/D=3 - Bottom View of the Computational Mesh

3.4.2 Eleven Jet Repeating Case I Results - Re=4,000, y/D=4.03

Numerical results for the 11 jet case with a repeating boundary condition placed at y/D=4.03, a non-dimensional jet target spacing of z/D=3, and a jet Reynolds number of 4,000 are presented in Figures 3.50 to 3.60. Results for time averaged Nusselt number, shown in Figure 3.50, appear qualitatively the same as the center row in Figure 2.14. With the increased spatial resolution made possible via numerical simulation, sharper spatial gradients in time averaged Nusselt number are observed. Figure 3.51 presents the time maximum y^+ value for the first grid point normal to the wall of the impingement jet target surface. All y^+ values are of the order of, or less than 1, indicating that a sufficiently fine grid point clustering scheme was used near the impingement surface, despite the fact that the mesh was based upon the coarsest mesh in the grid dependency study presented in Section 3.2.6.

Figures 3.52 to 3.57 present time averaged results on a plane constructed about the centerline of the flow field (y=0). Figure 3.52 indicates the time averaged, in plane velocity magnitude. It is readily evident that the velocity in the pressure chamber was indeed very low. The velocity is highest in the impingement jet nozzles, and rapidly decreases as the jets approach and interact with the target surface. The downstream jets are severely bent, and the core of the downstream jets is significantly weakened prior to impacting the target surface. Figure 3.53 shows the normal component of vorticity about the plane placed at the centerline of the flow. In the pressure chamber, the velocity is very low and the fluid is nearly irrotational. Near the entrance to the impingement jet nozzles, and near every wall downstream, the flow is rotational due to the boundary layers that develop on the nozzle walls. In addition, in the free shear regions vorticity is very high. Counter rotating vortices form as a result of the stagnation point at the intersection of two nearby jets.

Figures 3.54 and 3.55 show the time averaged static and total pressure, respectively, at the centerline of the flow. Both the static pressure and total pressure are nearly uniform in the pressure chamber due to the low flow velocity in the pressure chamber. As the flow accelerates through the nozzles, the static pressure is reduced and so is the total pressure. Reduction in total pressure is due to viscous losses in the flow field. There is some asymmetry in the pressure distribution seen in Figures 3.54 and 3.55. This is believed to be due to the coarse point distribution utilized in the upstream portion of the pressure chamber. Because this asymmetry is far upstream of the

impingement jet target surface, it is not believed to effect the Nusselt number significantly.

Static temperature is presented in Figures 3.56 and 3.57. Throughout the bulk of the flow field, the static temperature is nearly uniform. There is only a slight rise in free stream temperature when comparing the static temperature in the pressure chamber to that of the impingement region. This justifies the use of the assumption that the upstream temperature was approximately equal to the free stream temperature when calculating the film temperature for evaluating the thermal conductivity for the determination of Nusselt number. Near the impingement jet target surface, however, large temperature gradients occurred within the flow field, as is evident in Figure 3.57, which shows a close up of the fluid temperature near the impingement jet target surface at the interaction of two nearby jets. These large temperature gradients were the dominant cause in density variation in the flow field, which mandated that the numerical simulations solve the compressible Navier-Stokes equations.

Figure 3.58 presents an animation of the unsteady Nusselt number. When comparing this animation qualitatively to the animation of the experimentally determined Nusselt number in Figure 2.19, it is evident that the unsteadiness captured via the the numerical simulation is of higher amplitude and frequency than that which was determined experimentally. It is believed that the thermal conductivity, specific heat, and thickness of the stainless steel heated foil may have damped the unsteadiness in the Nusselt number which was measured experimentally.

Figures 3.59 and 3.60 present animations of unsteady in plane velocity magnitude and normal component of vorticity on the same plane as in Figures 3.52 and 3.53. As can be seen, flow upstream of the impingement jet nozzles exhibits steady flow characteristics. Once the flow exits the impingement jet nozzles, it becomes highly unsteady.

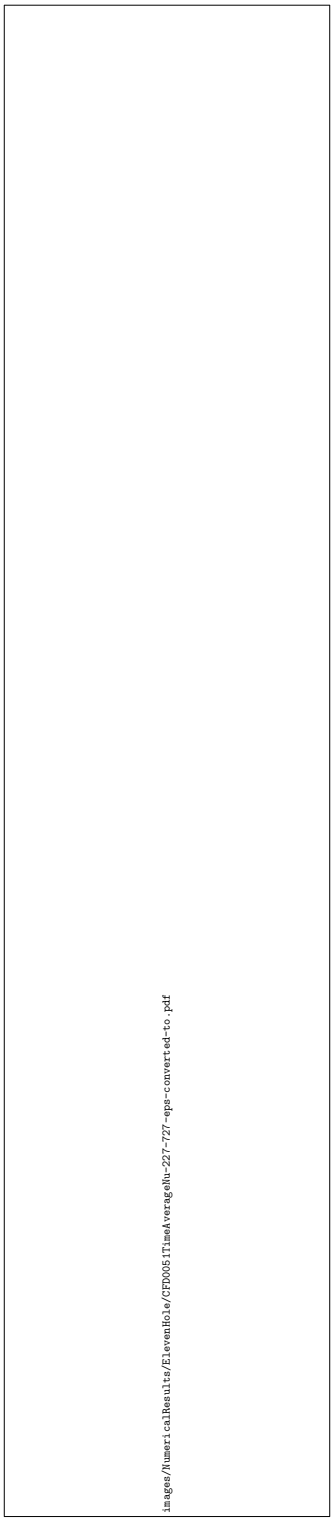


Figure 3.50: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Time-Averaged Nusselt Number - Numerical

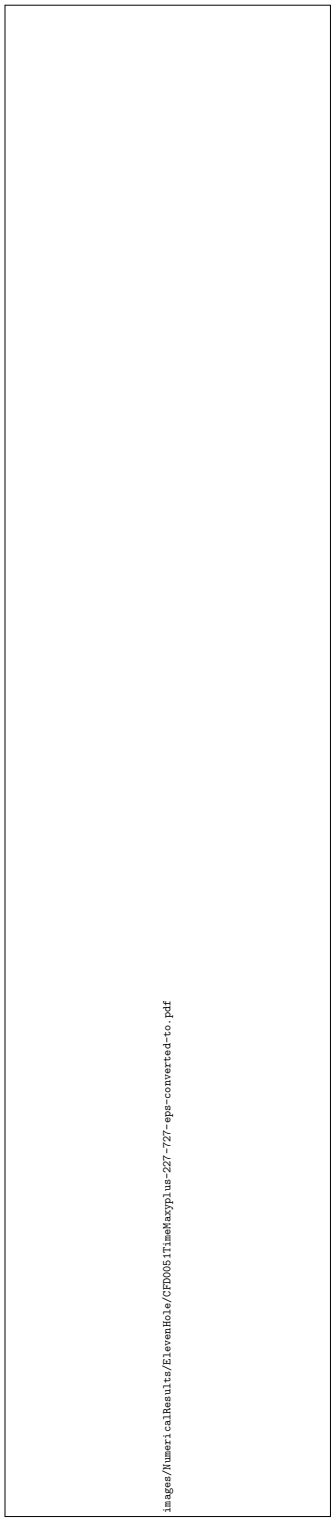


Figure 3.51: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Time-Maximum y^+ Value for the First Grid Point Away From the Target Surface - Numerical

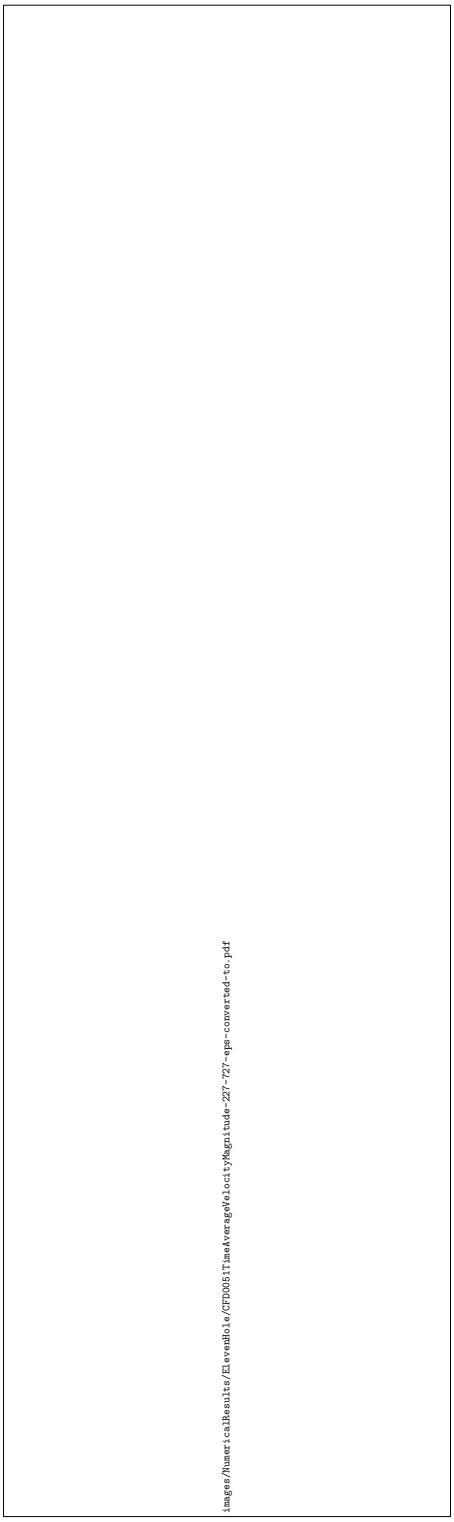


Figure 3.52: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Time-Averaged In Plane Velocity Magnitude - Numerical

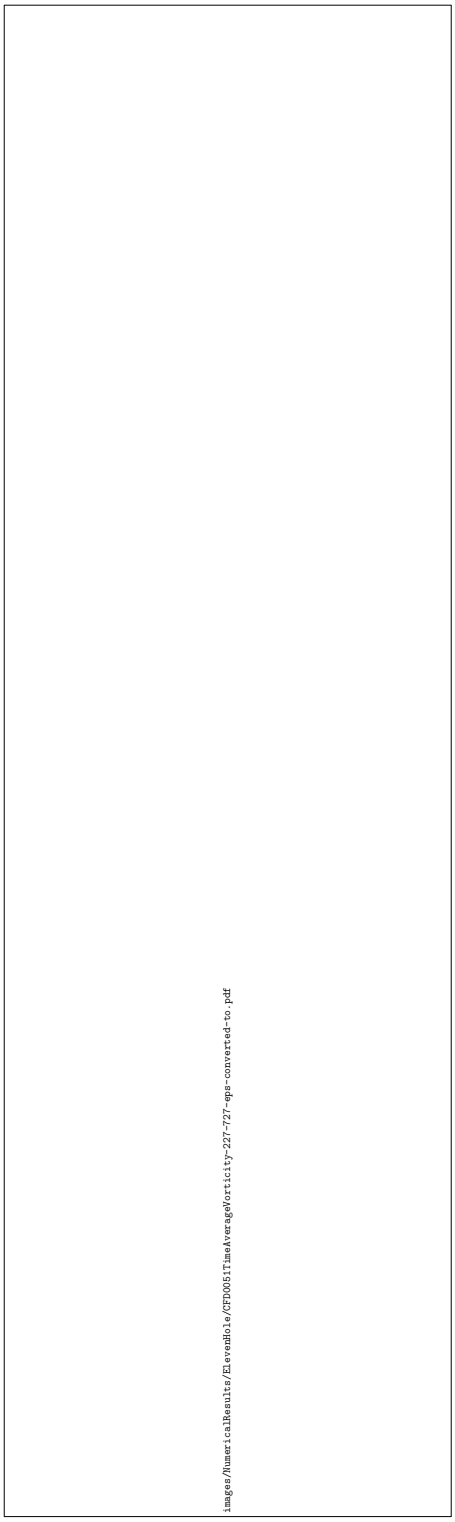


Figure 3.53: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Time-Averaged Normal Component of Vorticity - Numerical

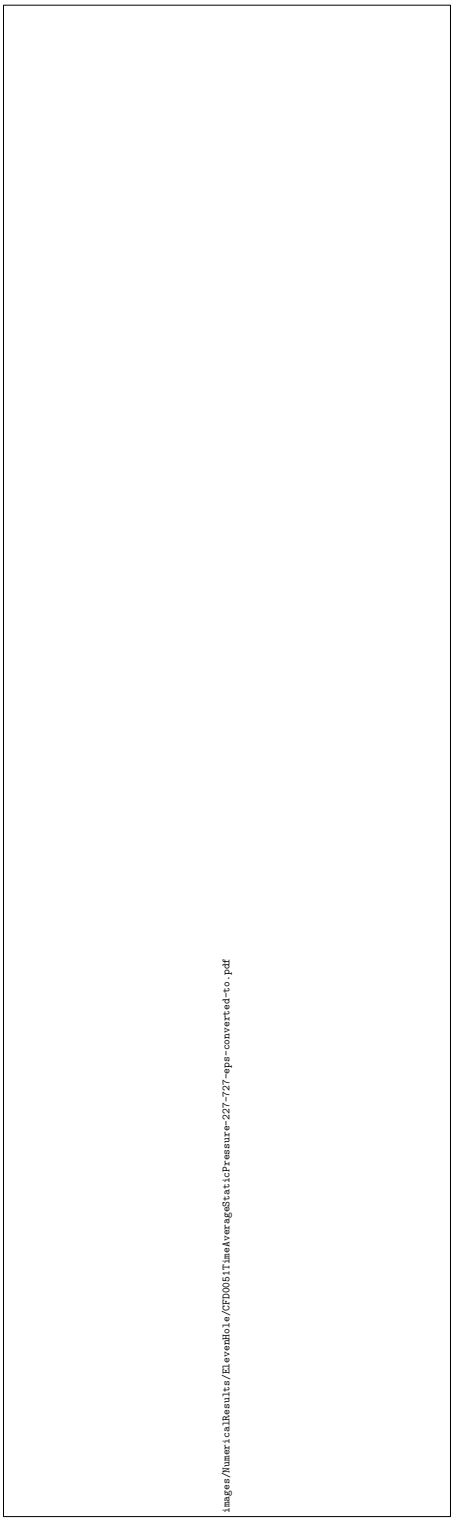


Figure 3.54: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Time-Averaged Static Gauge Pressure - Numerical

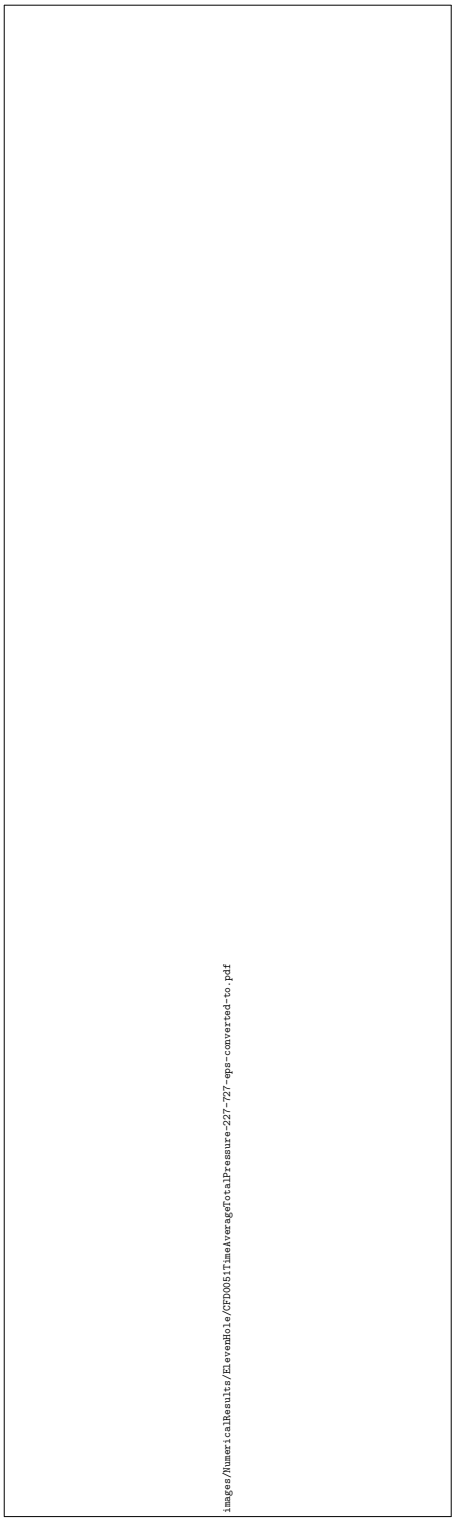


Figure 3.55: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Time-Averaged Total Pressure - Numerical

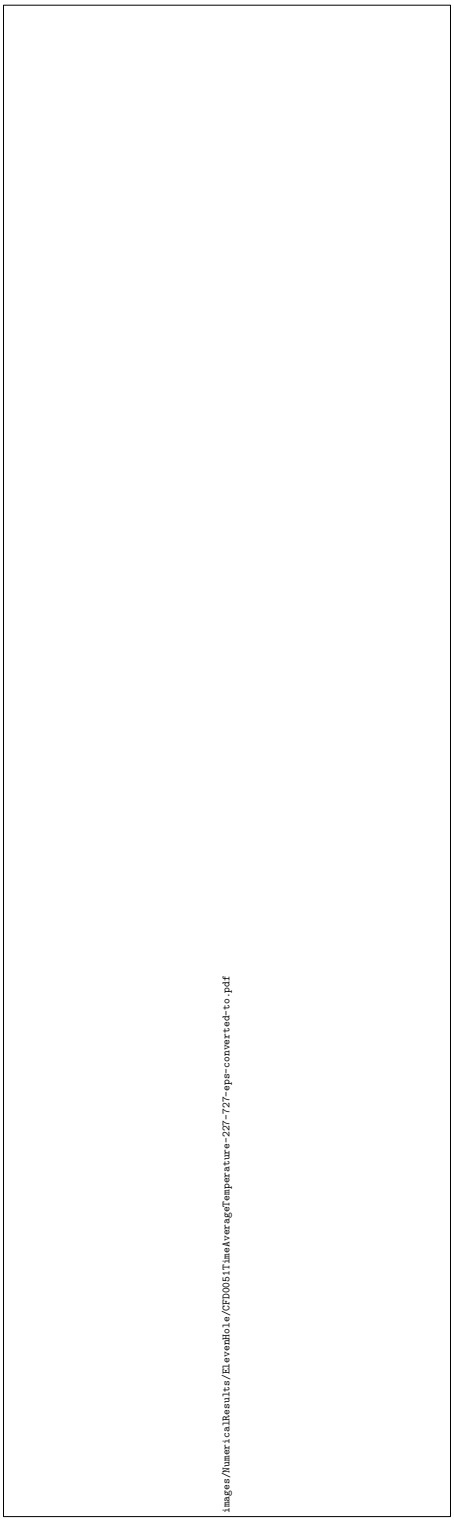


Figure 3.56: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Time-Averaged Static Temperature - Numerical

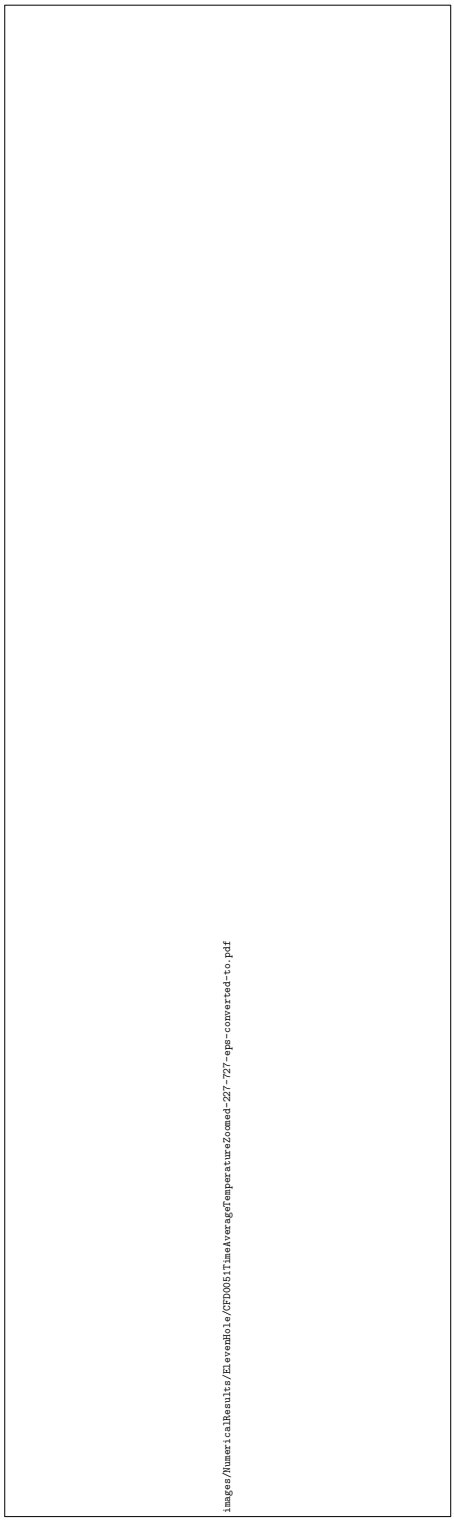


Figure 3.57: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Time-Averaged Static Temperature Zoomed to the Boundary Layer at the Interaction of Two Jets - Numerical

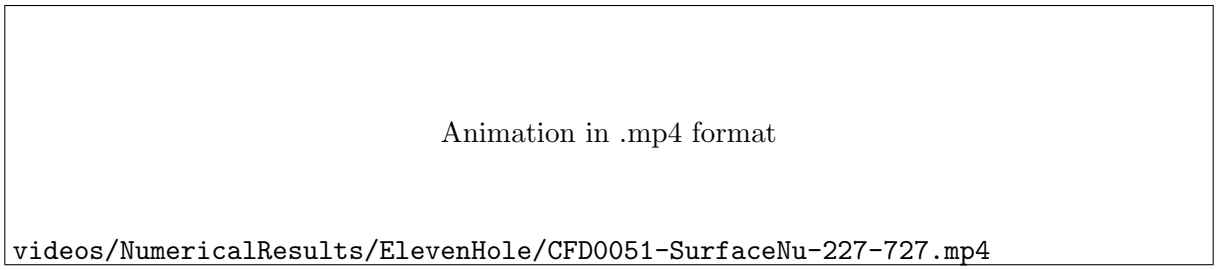


Figure 3.58: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Animation of Unsteady Surface Nusselt Number

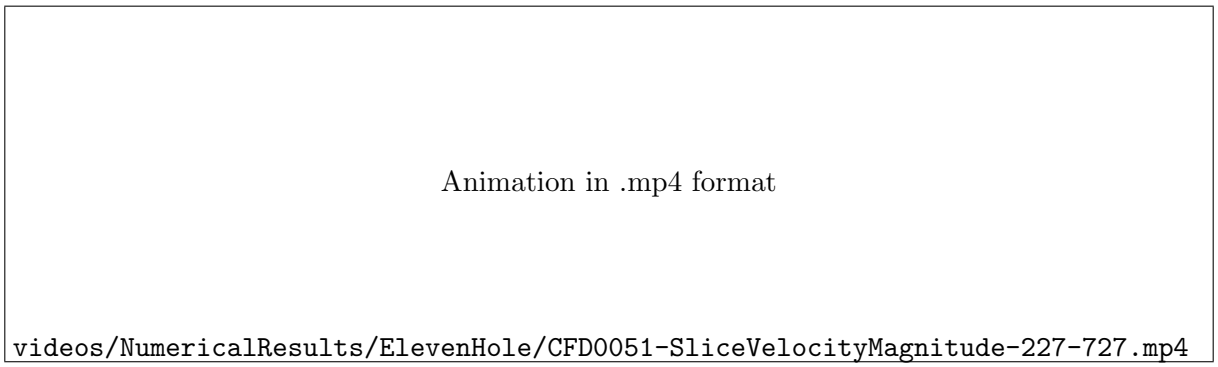


Figure 3.59: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Animation of Unsteady in Plane Velocity Magnitude at the Center Line of the Flow

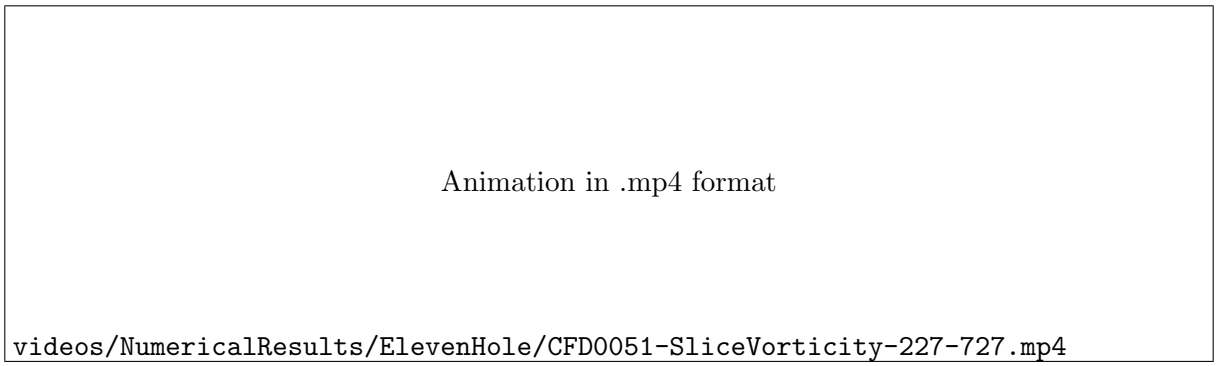


Figure 3.60: Eleven Jet Repeating Case I - Re=4,000 - z/D=3 - y/D=4.03 - Animation of Unsteady Normal Component of Vorticity at the Center Line of the Flow

3.4.3 Eleven Jet Repeating Case II Results - Re=4,000, y/D=3

Numerical results for the 11 jet case with a repeating boundary condition placed at non-dimensional spacing of y/D=3, a non-dimensional jet target spacing of z/D=3, and a jet Reynolds number of 4,000 are presented in Figures 3.61 to 3.71. These results are qualitatively similar to the results presented in Figures 3.50 to 3.60 for the previous case with the repeating boundaries placed at a non-dimensional spacing of y/D=4.03. This was expected due to the qualitative similarities in geometry between the two cases. Although qualitatively similar, the results do show some quantitative differences. The stagnation point due to the interaction of the nearby "virtual jets" has been shifted with the repeating boundaries closer to the center of the impingement jets. Due to this shift in the stagnation point, the corresponding reduction in convective heat transfer on the target surface has also been shifted. Additionally, the bending of the downstream jets is more severe due to the increased cross flow due to the reduced cross sectional area in the case with the repeating boundaries placed at a non-dimensional spacing of y/D=3. A more quantitative comparison of the two 11 jet cases will be presented in Section 3.5.

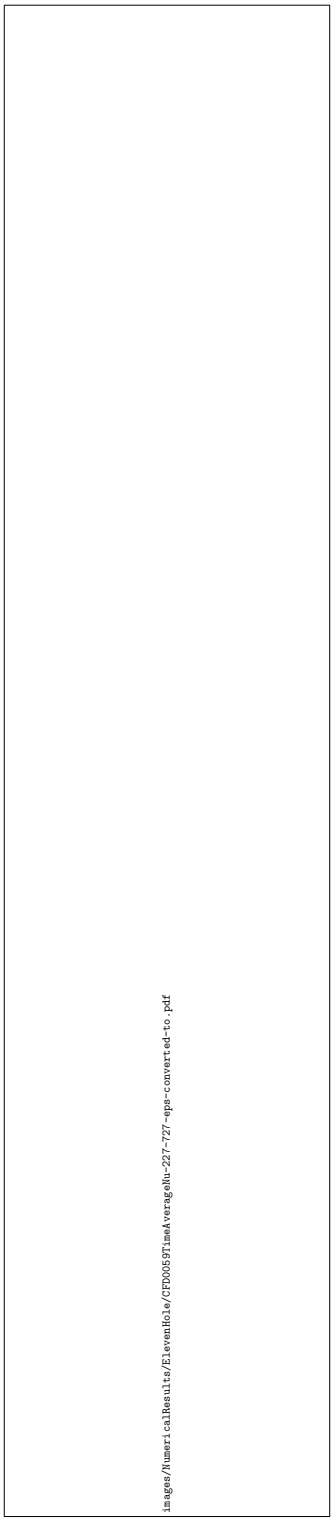


Figure 3.61: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Time-Averaged Nusselt Number - Numerical

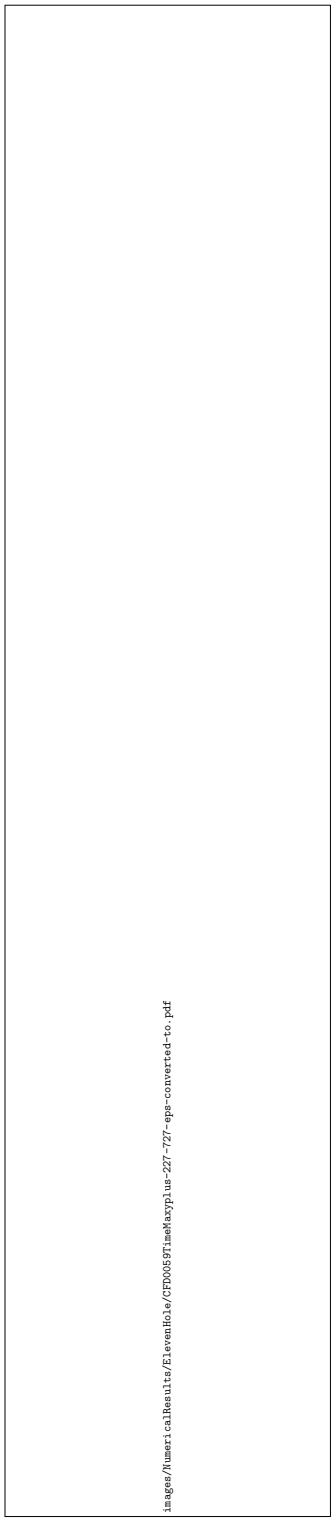


Figure 3.62: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Time-Maximum y^+ Value for the First Grid Point Away From the Target Surface - Numerical

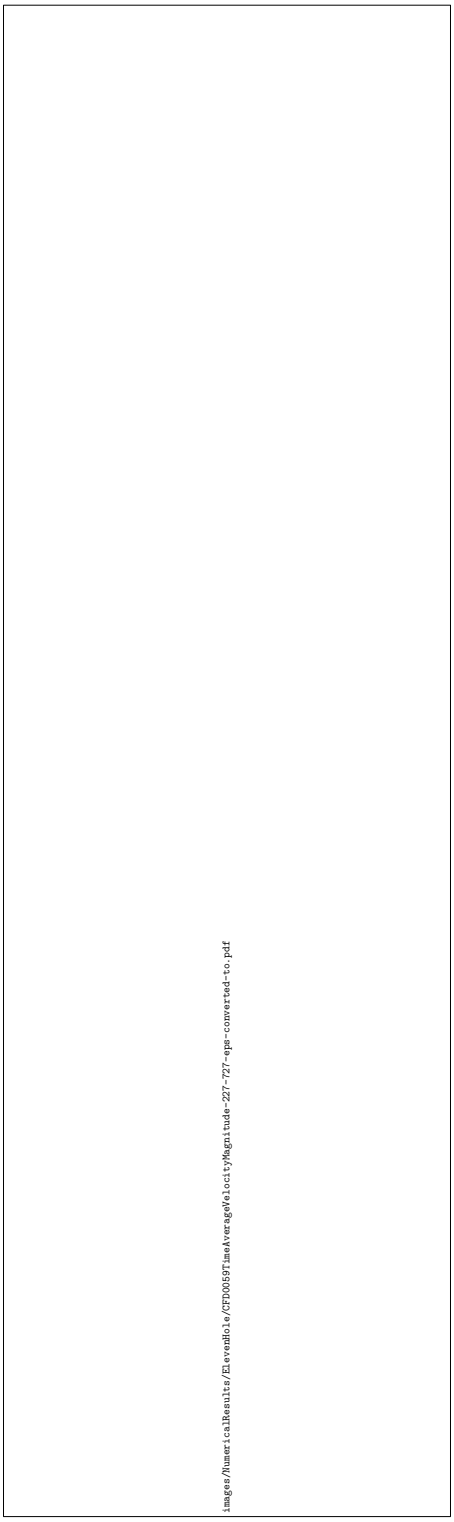


Figure 3.63: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Time-Averaged In Plane Velocity Magnitude - Numerical

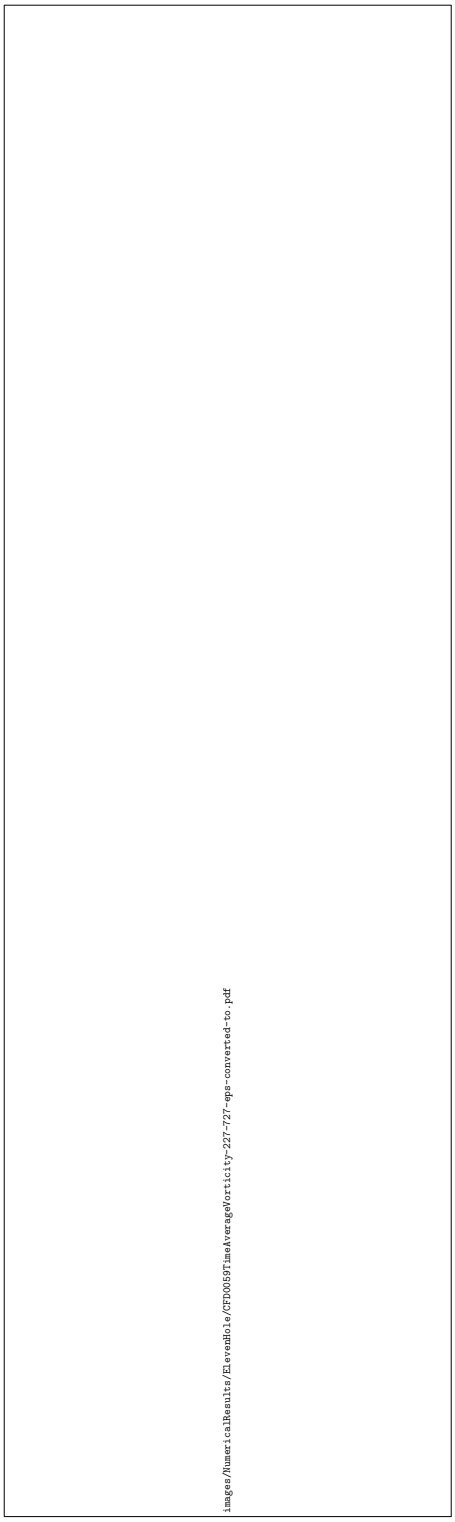


Figure 3.64: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Time-Averaged Normal Component of Vorticity - Numerical

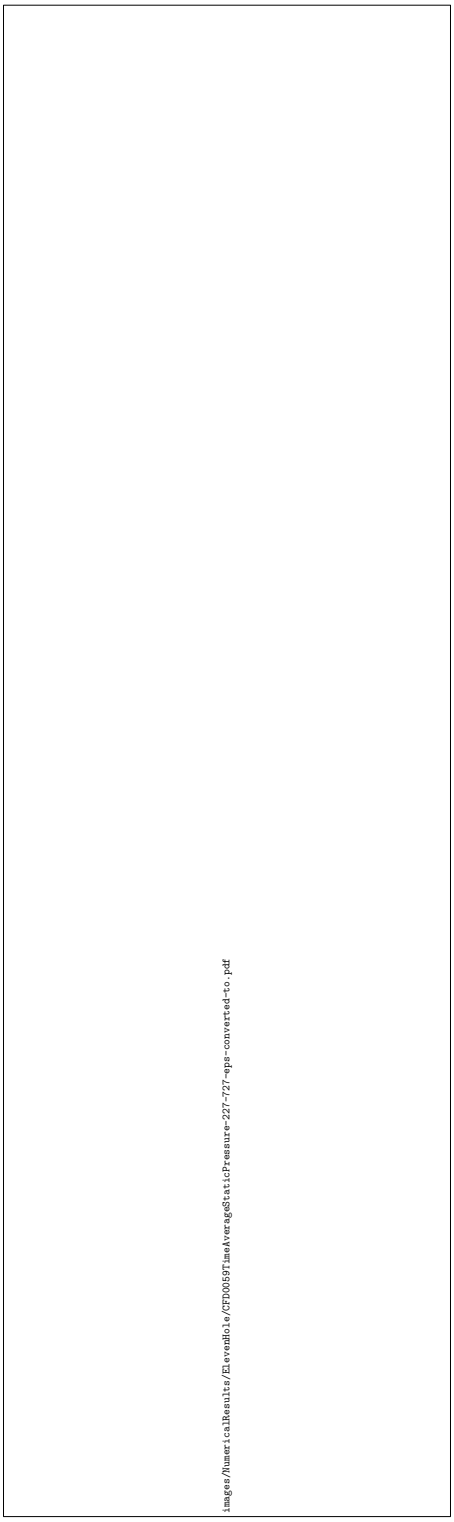


Figure 3.65: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Time-Averaged Static Gauge Pressure - Numerical

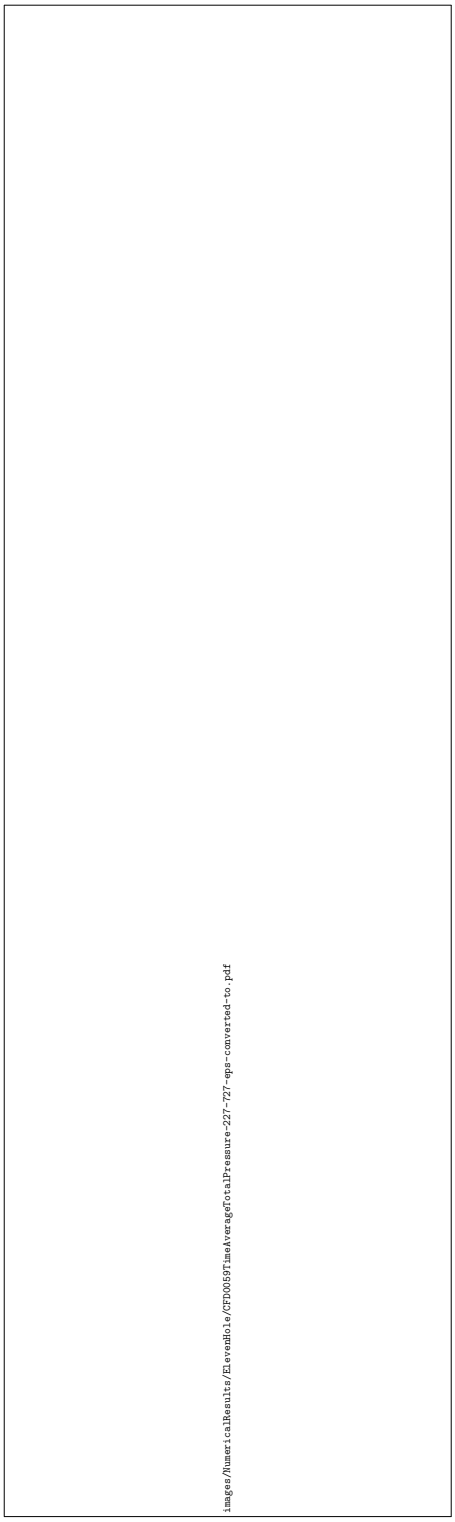


Figure 3.66: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Time-Averaged Total Pressure - Numerical

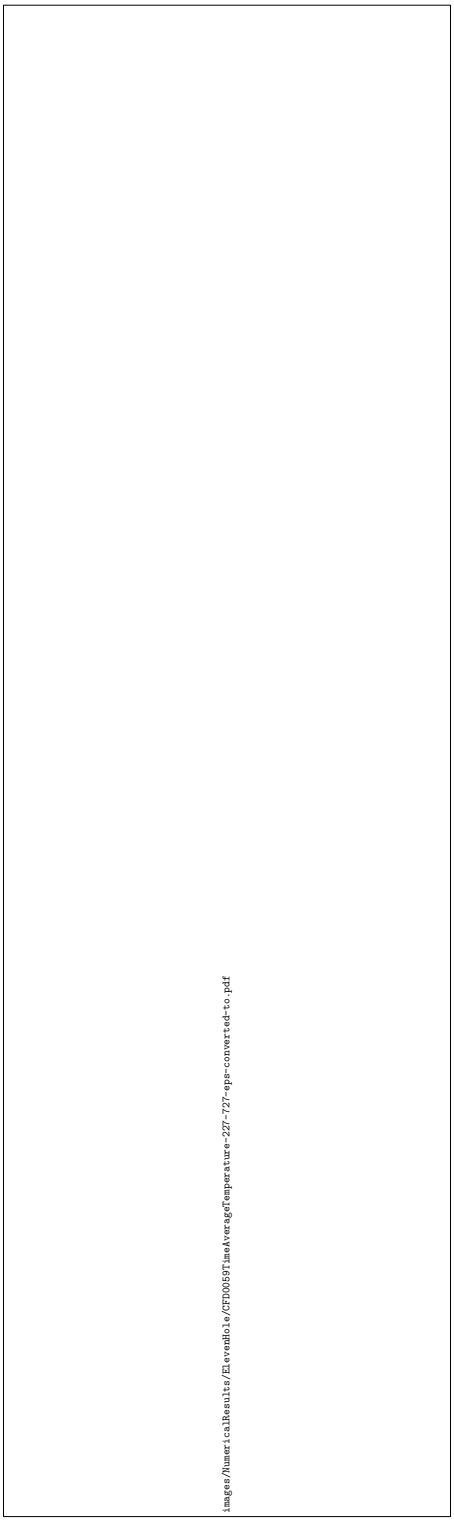


Figure 3.67: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Time-Averaged Static Temperature - Numerical

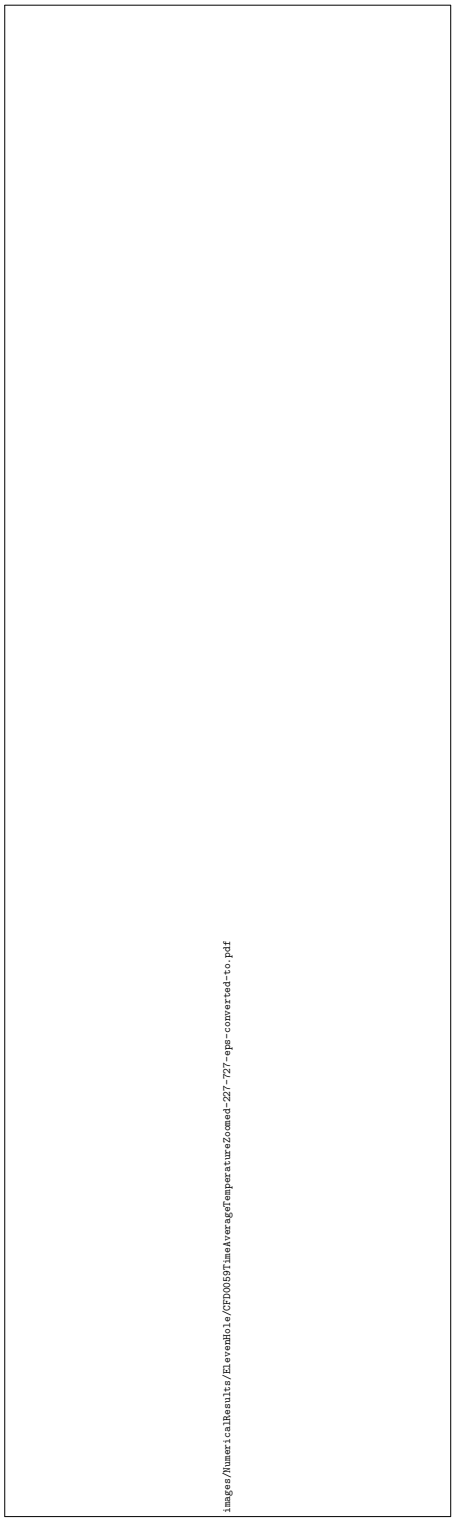


Figure 3.68: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Time-Averaged Static Temperature Zoomed to the Boundary Layer at the Interaction of Two Jets - Numerical

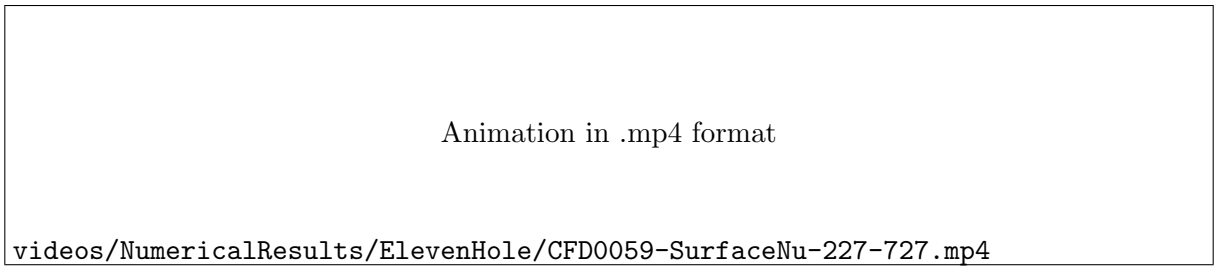


Figure 3.69: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Animation of Unsteady Surface Nusselt Number

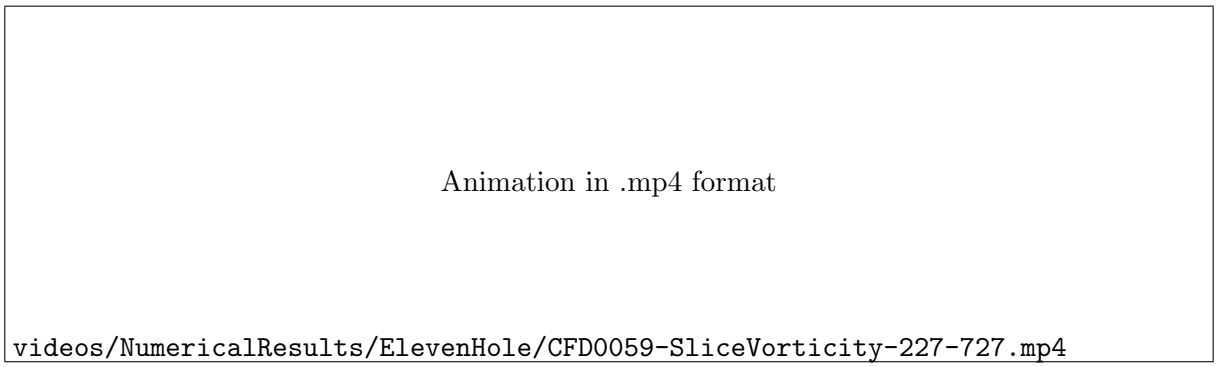


Figure 3.70: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Animation of Unsteady Normal Component of Vorticity at the Center Line of the Flow

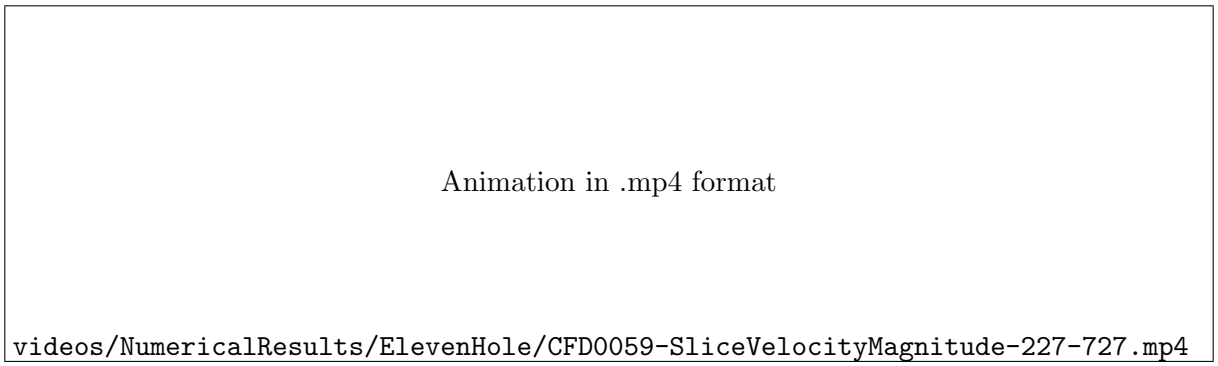


Figure 3.71: Eleven Jet Repeating Case II - Re=4,000 - z/D=3 - y/D=3 - Animation of Unsteady in Plane Velocity Magnitude at the Center Line of the Flow

3.5 Comparison of Spanwise Averaged Results

Figure 3.72 shows the time and spanwise average Nusselt number for the 11 jet numerical case with the repeating boundaries placed at a non-dimensional distance of y/D=4.03 (Eleven Jet Repeating Case I), the 11 jet numerical case with the repeating boundaries placed at a non-dimensional distance of y/D=3 (Eleven Jet Repeating Case II), and the center row of experimental results of an array of 55 impingement jets, all with a double exit configuration, non-dimensional target spacing of z/D=3, and a jet Reynolds number of 4,000. For the numerical case with y/D=4.03 and the experimental case, data was truncated to y/D=$\pm$1.5 for the calculation of time and span average. For all multi jet cases, the spanwise average was further truncated to x/D=$\pm$1.5 in Figure 3.73.

It should be noted that the experimental results compared for this double exit configuration, non-dimensional target spacing of z/D=3, and jet Reynolds number of 4,000, although of the same flow conditions, are from a different day of testing than the experimental results presented in Section 2.4. The boundary conditions for all numerical simulations presented in Figures 3.72 and 3.73 were copied from experimental data from the experimental case compared to in Figures 3.72 and 3.73, rather than from data from an experimental case presented in Section 2.4. Although seemingly inconsistent with the experimental component of this work, this approach was taken in order to maintain consistency within the numerical portion of the study. The results presented in section 2.4 were recorded after the numerical study was initiated, and rather than redo all numerical cases using the exact boundary conditions that were measured, the numerical study was continued using boundary conditions from the same experimental case as when the numerical study was initiated. No change was made in the experimental test procedure between the experimental results presented in Section 2.4 and the results presented in Figures 3.72 and 3.73, except for a change in the time sample recorded. For all unsteady cases presented in Figures 3.72 and 3.73, 11 physical seconds of data was time averaged. Although 34.26 physical seconds of data were averaged in Section 2.4, only 11.42 physical seconds were averaged in this section due to the increased computational demand with negligible change in solution.

As was discussed, neither 11 jet CFD case presented accurately reflected the flow geometry found in the experimental test rig, however, the CFD cases presented were anticipated to provide some upper and lower bounds on the time averaged Nusselt number. Reviewing Figure 3.72 shows some

agreement with these expectations. Both CFD cases were in close agreement with the experimental results near the center of the 3 central jets. The y/D=4.03 case showed an over prediction of spanwise average Nusselt number near the center of the outer impingement jets. This over prediction is expected because the y/D=4.03 case did not exhibit a stagnation region due to nearby jet interactions within the range of y/D=±1.5, but rather, experienced a stagnation region due to nearby jet interactions near y/D=±2.015, which was outside of the area used to compute the spanwise averages presented in Figures 3.72 and 3.73. The y/D=4.03 case was expected to have similar cross flow levels as was found in the test rig because the cross sectional area was matched to that of the test rig. With similar cross flow levels, the phase of the time and spanwise average Nusselt number more closely matched that of the experimental results.

The 11 jet case with y/D=3 featured similar nearby jet interactions as the experimental case, however, due to the reduced cross sectional area when compared to the experimental case, featured higher cross flow levels. These higher cross flow levels are clearly seen in Figures 3.63 and 3.64. The phase of the time and spanwise average Nusselt number is further shifted from that of the experimental results, and of the y/D=4.03 CFD case. Away from the center of the flow field, the time and spanwise average Nusselt number decreases more rapidly than the other cases.

Both CFD cases feature an increase in time and spanwise average Nusselt number at x/D=±6. The y/D=4.03 case also showed an increase in Nusselt number at x/D=±9 as well. Reviewing Figures 3.50 and 3.61, it appears as though the core of the jet is spread out in these areas. It is believed that this increase may be due to a critical level of cross flow. The experimental results also showed a minor increase near x/D=+6 and x/D=±9. Further three dimensional analysis of the flow field is required in order to provide a thorough explanation of this phenomenon.

All CFD results saw jagged and significantly lower troughs in the time and span averaged Nusselt number at the mid point between neighboring jets when compared to the experimental results. As can be seen in Figures 3.52, 3.53, 3.63, and 3.64, a stagnation region occurs where the neighboring jets interact. Two counter rotating vorticies are formed at these stagnation points. Also, in the stagnation region, the boundary layer is thickened and the heat transfer coefficient is reduced. In these regions with low heat transfer coefficient, the surface temperature is sharply elevated. It is believed that the negligible lateral conduction assumption stated in Section 2.1.2 may be invalid in these localized regions which could cause the post processing of the experimental

results to indicate a higher convective heat transfer coefficient. An attempt was made to model volumetric heating and three dimensional conduction in the heated foil using Fluent. However, this was not successful. After days of troubleshooting, it is believed that there is some error in Fluent's implementation of the volumetric heat generation boundary condition.

Figure 3.72: Time and Span Averaged Nusselt Number - 11 Jets, Re=4,000, y/D=3 - Numerical vs Experimental

Figure 3.73 presents the time and spanwise average Nusselt number for all cases presented which have featured at least 2 exits, a non-dimensional jet target spacing of z/D=3, and a jet Reynolds number of 4,000. As mentioned earlier, the multi jet CFD and experimental results were truncated to x/D=±1.5 and y/D=±1.5 in order to provide appropriate comparison to the single jet CFD cases. A summary of all cases compared in Figure 3.73 is presented in Table 3.3.

Jet Reynolds Number	Hole Spacing (x/D)	Hole Spacing (y/D)	Target Spacing (z/D)	Hole Pattern (x × y)	Exit Configuration	Sample Time (s)	Grid Points (million)	Case Number
4,000	3	4.03 ("virtual")	3	11×∞	Double	11	16.9	11 Jet Repeating Case I
4,000	3	3 ("virtual")	3	11×∞	Double	11	14.3	11 Jet Repeating Case II
4,000	3	3	3	11×5	Double	11.42	N/A	2010-11-23
4,000	N/A	6 ("virtual")	3	1×∞	Double	11	2.4	Single Jet Repeating Case I
4,000	N/A	3 ("virtual")	3	1×∞	Double	11	1.9	Single Jet Repeating Case II
4,000	N/A	N/A	3	1×1	Quadruple	N/A	12.6	$1/1.3^2$

Table 3.4: List of Spanwise Averages Compared

As can be expected, the single jet CFD case, with no cross flow and no jet to jet interactions featured the highest spanwise average Nusselt number within the jet core. The Single Jet Repeating Case I saw the second highest spanwise average Nusselt number. This case had no cross flow, and the periodic boundaries were placed at spacing of y/D=6, which was outside of the heated region, indicating that the nearby jet interactions simulated by the repeating boundary condition should have provided negligible effect on the Nusselt number (as was evident in Figure 3.32). The next highest case was the Single Jet Repeating Case II. This case had its repeating boundary conditions placed at a spacing of y/D=3. With this close placement of the repeating boundary condition (and

the associated "virtual" nearby jets), significant interactions between the neighboring jets were present within the heated region, as was evident in Figure 3.38.

The central jet of the multi jet cases had the lowest time and spanwise average Nusselt number of the numerical simulations. Although the central jet of the multi jet cases still should have experienced little or no cross flow, the spanwise average Nusselt number was still lower than the single jet cases with the repeating boundary conditions. This can be explained due to the additional nearby jet interactions that occur in the multi jet cases with the repeating boundary condition, as well as in the experimental results. The single jet cases with a repeating boundary condition experience interactions with nearby jets on two sides of the jet. The multi jet cases with a repeating boundary condition, and the experimental results experienced interactions with neighboring jets on all four sides. The multi jet simulation with the repeating boundaries placed at a non-dimensional distance of y/D=3 saw stronger interactions due to nearby jets, and the stagnation region near the repeating boundary was within the range of y/D=$\pm$1.5 in which the Nusselt number was averaged.

It should also be restated that the jet Reynolds numbers used when conducting numerical simulations for the single jet was based upon an area weighted average jet Reynolds number through all jets in the matching experimental test case. For the central jet of the multi jet cases, the jet Reynolds number through the central jet was slightly different than the jet Reynolds number of the single jet cases. Also, as was mentioned in Section 2.4, there was some uncertainty whether the offset of the center jet in the experimental results is due to a misalignment of the IR camera, or a physical characteristic of the flow. Other than this offset, the multi jet cases were in fairly close agreement with the experimental results near the jet core.

Figure 3.73: Time and Span Averaged Nusselt Number - Single/Center Jet, Re=4,000, y/D=3 - Numerical vs Experimental

Chapter 4

Conclusions

A new experimental test facility was designed, manufactured, and commissioned for studying heat transfer characteristics of an array of 55 impingement jets. Through the use of a steady state Infrared Thermography technique, unsteady convective heat transfer characteristics, although mild, could be observed. Results for target surface Nusselt number have been presented for jet Reynolds numbers of 4,000, 8,000, 12,000, and 15,000. Two cross flow configurations have been studied through the use of both a single exit, and a double exit configuration. For each Reynolds number and exit configuration, non-dimensional target spacings of 3, 4, and 5 have also been studied. The results show a strong dependency between the Nusselt number and the impingement jet Reynolds number. The Nusselt number is also impacted by the exit flow configuration, and by the nozzle to target spacing, although more weakly than jet Reynolds number for the ranges studied in the present work.

There are several areas in which future work with this test facility could be directed. The current test facility could be used to study the convective heat transfer characteristics throughout the entire range of jet Reynolds numbers that the facility has been designed for (1,000 to 30,000). Due to limitations of the present measurement technique, only 45 jets could be studied with the double exit configuration, and only 40 jets could be studied for the single exit configuration, rather than the entire 55 impingement jets present in the flow field. Some effort could be made to extend range of the measurement domain with this Infrared Thermography technique to include all 55 impingement jets in the flow field. The present results agree qualitatively with trends presented by previous researchers, however, no quantitative comparisons have been made. It may be worthwhile

to modify the test rig geometry to better match a historically tested flow configuration. Such an activity would allow for more detailed comparisons of results obtained using different measurement techniques.

Numerical simulations were also conducted for a smaller class of impingement jet heat transfer problems. Limitations due to number of licenses, and the reliability of the Fluent license server limited the complexity of the geometry that could be studied. As a result, the flow fields studied numerically were geometrically different than the experimental cases studied. Although of different geometry, much was still gained from these results which have been obtained numerically, as well as the process that was used to perform the simulations. Mesh topology and grid point clustering schemes have been developed and evaluated, and can be useful for future studies. Results for the baseline performance of single jets has been presented, and can be used as an upper limit on the convective heat transfer coefficients possible with impingement jets. Several cases were investigated with single impingement jets and a repeating boundary condition. These cases demonstrated the impacts of jet to jet interactions on the Nusselt number of impingement jets, without cross flow. The 11 jet case with y/D=3 demonstrated a configuration that was expected to have similar jet to jet interaction effects as the central row in the test rig. However, it was not expected to have the same cross flow effects. The 11 jet case with y/D=4.03 demonstrated a configuration that was expected to have similar cross flow effects, but significantly different jet to jet interaction effects. With all of these numerical simulations, further insight has been gained about flow and heat transfer of impingement jets. The simulation results have been able to provide vast amount of information regarding the three dimensional flow characteristics, data which was not obtainable with the present experimental technique.

For any continuation of this combined experimental and numerical study of impingement jet heat transfer, the experimental test facility should be modified such that representative geometry can be studied numerically with a reduced computational burden. The simplest modification to the test facility would be to manufacture a new impingement jet nozzle plate that features impingement nozzle spacings of x/D=3 and y/D=4.03. Doing so would allow experimental results to be compared to the current numerical results in a more meaningful way.

Some effort can be made to improve the boundary conditions implemented in the numerical simulations. It is hypothesized that some lateral conduction may have been present in the heated foil

which could have caused the experimental results to deviate from the numerical results, especially in areas with low convective heat transfer coefficient, the spatial temperature gradient may have been high, causing some lateral conduction in the foil. Further investigations should be performed to validate this hypothesis. Also, an inlet turbulence intensity of 0.7% was used based upon guidance of Dr. Shichuan Ou. As heat transfer can be significantly effected by turbulence intensity, it would be best to measure the turbulence intensity in the pressure chamber of the test rig so that the modeled turbulence intensity at the inlet can be defined to match the true turbulence intensity of the experimental procedure. A fixed temperature on the outlet boundaries of 300K was used for all cases. Although it is believed to have minimal impact on the convective heat transfer coefficient, in future simulations, it may be best to define the outlet boundaries to match the ambient temperature measured within the test cell.

The computational meshes developed were defined with a goal that on all walls, the first grid point normal to the surface was placed at a maximum y^+ value of approximately 1. Although this goal was achieved, no effort was made, however, to validate that the law of the wall was indeed maintained within the turbulent boundary layers present in these numerical simulations of impingement jets. Further effort could be conducted to validate that the solution technique did indeed satisfy the law of the wall.

Transitioning to a more versatile, lower cost CFD solver, designed for high performance computing, rather than Fluent, would allow for larger meshes featuring more complex geometry to be studied. With the ability to solve more complex flow fields, the simulation of flow and heat transfer in entire impingement jet arrays may be possible in a feasible amount of time. In addition to the fact that the current study was not able to simulate the same geometry as was studied experimentally, the use of repeating boundary conditions can have difficulty capturing the true unsteadiness in such a complex flow field. Modeling the entire geometry present in the test rig will allow for the unsteady effects to be more accurately resolved.

Using both experimental and numerical techniques, characteristics of impingement jet heat transfer have been studied. In all cases studied, highly localized heat transfer characteristics were observed. Due to the complexity of the flow field and the impacts of cross flow on convective heat transfer characteristics, any heating/cooling system designer is encouraged to use the results presented as a guide during the design process, however, detailed experimental and/or numerical

studies of the exact flow configuration should be conducted prior to placing a part featuring an impingement jet heating/cooling system into service. Due to the authors lack of experience in numerical solution techniques at the onset of this study, some challenges driven by the complexity of the desired flow configuration were not identified until after the test facility was completely designed, commissioned, and operational, which impacted the ability to make meaningful comparisons between the results obtained using both techniques. Prior to initiating any future studies of impingement jet heat transfer, one is encouraged to more thoroughly understand the implications of the proposed flow configuration on both experimental and numerical methods.

References

[1] DM Kercher and W. Tabakoff. Heat transfer by a square array of round air jets impinging perpendicular to a flat surface including the effect of spent air. *J. Eng. Power*, 92(1):73–82, 1970.

[2] LW Florschuetz, DE Metzger, DI Takeuchi, and RA Berry. Multiple jet impingement heat transfer characteristic: Experimental investigation of in-line and staggered arrays with cross-flow. 1980.

[3] Y. Huang, S.V. Ekkad, and J.C. Han. Detailed heat transfer distributions under an array of orthogonal impinging jets. *Journal of Thermophysics and Heat Transfer*, 12(1):73–79, 1998.

[4] K. Kanokjaruvijit and R.F. Martinez-botas. Jet impingement on a dimpled surface with different crossflow schemes. *International Journal of Heat and Mass Transfer*, 48(1):161–170, 2005.

[5] T. Wang, M. Lin, and R.S. Bunker. Flow and heat transfer of confined impingement jets cooling using a 3-D transient liquid crystal scheme. *International Journal of Heat and Mass Transfer*, 48(23-24):4887–4903, 2005.

[6] EI Esposito, SV Ekkad, Y. Kim, and P. Dutta. Novel Jet Impingement Cooling Geometry for Combustor Liner Backside Cooling. *Journal of Thermal Science and Engineering Applications*, 1:021001, 2009.

[7] P.R. Robertson. The design and validation of an impinging jet test facility. Thesis, Baylor University, Department of Mechanical Engineering, 2006.

[8] W. Sutherland. LII. The viscosity of gases and molecular force. *Philosophical Magazine Series 5*, 36(223):507–531, 1893.

[9] J. Hilsenrath, CW Beckett, WS Benedict, L. Fano, HJ Hoge, JF Masi, RL Nuttall, YS Touloukian, and HW Woolley. Tables of thermodynamic and transport properties. *US Nat. Bur. Stand. Circ*, 564, 1955.

[10] J.D. Anderson. *Modern compressible flow: with historical perspective*. McGraw-Hill Science/Engineering/Math, 2003.

[11] Colorado Engineering Experiment Station, Inc. *Recommended Procedure for the Use of Critical Flow Venturi/Nozzle Calibration Data, CEESI Technical Note TN-013*, August 1994.

[12] MatWeb, LLC. Westlake Plastics Zelux W Window Grade Polycarbonate Material Properties, 2011. `http://www.matweb.com/`.

[13] A. Bejan. *Convection heat transfer*, page 198. Wiley, 2004.

[14] L. Gao. Effect of Jet Hole Arrays Arrangement on Impingement Heat Transfer. Master's thesis, Louisiana State University, 2003.

[15] LW Florschuetz, CR Truman, and DE Metzger. Streamwise flow and heat transfer distributions for jet array impingement with crossflow. *Journal of Heat transfer*, 103:337, 1981.

[16] D.C. Wilcox. *Turbulence modeling for CFD*, page 16. DCW Industries, 2006.

[17] F.P. Incropera and D.P. DeWitt. *Fundamentals of heat and mass transfer*, page 395. J. Wiley, 2002.

Appendix A

Matlab Script Source Code

A.1 Common Scripts

A.1.1 PlotNuContour.m

```
function [ContourFigure,ColorbarHandle]=PlotNuContour(Nu,X,Y,D,TargetRe,MinXD,MaxXD,MinYD,
    MaxYD,MinNu,MaxNu,NuStep,HoverD,ContourScaleLabel,Time)
    %plot results in two dimensions

    %make the aspect ratio of the figure match that of the graph so minimal whitespace
        exists in the exported images, but still not too large for the screen
    MaxFigureHeight=900;
    MaxFigureWidth=1700;

    FigureHeight=MaxFigureHeight;
    FigureWidth=floor(FigureHeight*(MaxXD-MinXD)/(MaxYD-MinYD));
    if FigureWidth>MaxFigureWidth
            FigureWidth=MaxFigureWidth;
            FigureHeight=floor(FigureWidth*(MaxYD-MinYD)/(MaxXD-MinXD));
            if FigureHeight<400
                    FigureHeight=400;
            end
    end

    XZero=60;
    YZero=XZero;
    ContourFigure=figure('OuterPosition',[XZero YZero FigureWidth+XZero FigureHeight+
        YZero],'Name',['Re=',num2str(TargetRe)]);       %big figure

    contourf(X/D,Y/D,Nu,120,'LineStyle','none');
    hold all
    %contour(X/D,Y/D,Nu,60); %this line doesn't seem to work for some reason
    colormap('gray')
    caxis([MinNu, MaxNu])

    FontSize=40;
    LineWidth=3;
    ColorbarHandle=colorbar;
    if ~exist('ContourScaleLabel','var')
            ContourScaleLabel='Nu';
    end
    set(get(ColorbarHandle,'ylabel'),'String', sprintf(ContourScaleLabel), 'Rotation',
        90, 'VerticalAlignment', 'Bottom','FontSize',FontSize)
    set(gca,'FontSize',FontSize);
    set(get(gca,'XLabel'),'FontSize',FontSize);
    set(get(gca,'YLabel'),'FontSize',FontSize);
    set(gca,'LineWidth',LineWidth)
    set(ColorbarHandle,'LineWidth',LineWidth);
    set(ColorbarHandle,'ytick', [MinNu:NuStep:MaxNu])
```

```
41          %adjust the location of the colorbar
42          LocateColorbarLabel = get(ColorbarHandle,'ylabel');
43          pos = get(LocateColorbarLabel,'position');
44          pos(1,1) = pos(1,1)+10;
45          set(LocateColorbarLabel,'Position',pos);
46
47          xlabel('x/D');
48          ylabel('y/D');
49          daspect([1 1 1])
50
51          if exist('Time','var')
52                  title(['Re=',thousands(TargetRe),'  z/D=',num2str(HoverD),'  t=',num2str(
                        Time,'%07.4f'),'s'])
53          else
54                  title(['Re=',thousands(TargetRe),'  z/D=',num2str(HoverD)])
55          end
56          xlim([MinXD MaxXD])
57          ylim([MinYD MaxYD])
58          set(get(ContourFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','
                YMinorTick','on','XTick',[-15:3:15],'YTick',[-6:3:6])
59          set(gcf,'Color','w')
60
61
62
63  end
```

A.1.2 PlotNuSpanwiseAverage.m

```
1   function [SpanwiseAverageFigure]=PlotNuSpanwiseAverage(SpanwiseTimeAverageNu,X,D,TargetRe,
        MinXD,MaxXD,Marker,LabelText,TitleText)
2           global SpanwiseAverageFigure
3
4           %plot results in 1 dimension
5           if exist('SpanwiseAverageFigure')
6                   if ishandle(SpanwiseAverageFigure)
7                           figure(SpanwiseAverageFigure)
8
9                           %for some reason the following few lines
10                          %didn't allow the plot to keep alternating the line colors
11                          %and hold all works since the color map is closed every time
12                          %plotting to this figure
13                          %if ishold~=1
14                          %       hold(get(SpanwiseAverageFigure,'CurrentAxes'))
15                          %end
16                          hold all
17                  else
18                          %SpanwiseAverageFigure=figure('OuterPosition',[30 30 1600 900]);
                                    %big figure
19                          SpanwiseAverageFigure=figure('OuterPosition',[30 30 622 500]);
                                    %small figure
20
21                  end
22          end
23
24          plot(X/D,SpanwiseTimeAverageNu,'Marker',Marker);
25          xlim([MinXD MaxXD])
26
27          xlabel('x/D');
28          ylabel('Nu');
29          title(['Time and Span Averaged Nusselt Number',TitleText]);
30          set(get(SpanwiseAverageFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick',
                'on','YMinorTick','on','XTick',[-15:3:15])
31          [~,~,~,currentlegend]=legend;
32          currentlegend{1,size(currentlegend,2)+1}=sprintf(LabelText);
33          legend(currentlegend,'Location','NorthEast');
34  end
```

A.1.3 thousands.m

```
function output = thousands(input)
%add a thousands separator to a number to make it more readable
import java.text.*
value = DecimalFormat;
output = char(value.format(input));
```

A.1.4 ReadExperimentalCaseListSpreadsheet.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%read additional descriptive data for this run
filehandle=fopen(PathToCaseListDataFile);
%note: if the number of delimiters is less than the total number of fields,
%matlab just breaks the current row at the last defined delimiter
%such that the next field is on a new row.
%because of this, extra delmiters are put in so that if a new field is
%added, it will still work okay
CaseListData = textscan(filehandle, '%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s','CollectOutput',1,'delimiter','\t');
fclose(filehandle);     %close file
CaseListData=CaseListData{1,1};  %simplify variable a little

%define variable and column mappings
CaseListDataVariables={
        'Field Name','Column Number','Units','String?'
        'CaseNumber',1,'','no'  %make sure to change this below in for loop also if column number changes
        'FirstTimeStep',9,'','no'
        'LastTimeStep',10,'','no'
        'D',11,'m','no'
        'FoilWidth',14,'m','no'
        'FoilLength',15,'m','no'
        'CameraAlignmentDate',13,'','no'
        'FoilEmissivity',16,'','no'
        'HoverD',5,'','no'
        'Misc',8,'','yes'
        'ExitConfiguration',7,'','yes'
        'ActualH',6,'mm','no'
        'ChannelWidth',17,'m','no'
        'MinXD',18,'','no'
        'MaxXD',19,'','no'
        'TargetRe',4,'','no'
        };



%double check values are mapped properly
disp(' ')
disp('Check and make sure each field is properly mapped to the correct variable')
for CaseListDataVariableCount=2:size(CaseListDataVariables,1)
        %double check values are mapped properly
        disp([CaseListDataVariables{CaseListDataVariableCount,1},'[',CaseListDataVariables{CaseListDataVariableCount,3}, '] <------------> ',CaseListData{1,CaseListDataVariables{CaseListDataVariableCount,2}}]);
end
disp('please wait')
for CaseListDataCount=2:size(CaseListData,1)
        if strcmp(CaseListData(CaseListDataCount,1),ExperimentalCaseNumber)
                for CaseListDataVariableCount=2:size(CaseListDataVariables,1)
                        if strcmp(CaseListDataVariables{CaseListDataVariableCount,4},'no')
                                eval([CaseListDataVariables{CaseListDataVariableCount,1},'=str2num(CaseListData{CaseListDataCount,CaseListDataVariables{CaseListDataVariableCount,2}});']);
                        else
                                eval([CaseListDataVariables{CaseListDataVariableCount,1},'=CaseListData{CaseListDataCount,CaseListDataVariables{CaseListDataVariableCount,2}};']);
```

```
                    end
            end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.1.5 ReadLabViewData.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%read LabView data for case
filehandle = fopen([RawExperimentalDataPath,'/',ExperimentalCaseNumber,'/',
    ExperimentalCaseNumber,'.txt']);     %open file
%note: if the number of delimiters is less than the total number of fields,
%matlab just breaks the current row at the last defined delimiter
%such that the next field is on a new row.
%because of this, extra delmiters are put in so that if a new field is added, it will still
    work okay
LabViewData = textscan(filehandle, '%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s
    %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s','CollectOutput',1,'delimiter'
    ,'\t');
fclose(filehandle);      %close file
LabViewData=LabViewData{1,1};  %simplify variable a little

%define variable and column mappings
LabViewDataVariables={
        'Field Name','Column Number','Units'
        'UpstreamAverageTemperature',7,'F'
        'FoilVoltageDrop',13,'V'
        'FoilCurrent',12,'A'
        'AmbientTemperature',6,'F'
        'ActualExperimentalRe',9,''
        'mjets',8,'lbs/s'
        'AtmosphericPressure',4,'psia'
        };

%double check values are mapped properly
disp(' ')
disp('Check and make sure each field is properly mapped to the correct variable')

for LabViewDataVariableCount=2:size(LabViewDataVariables,1)
        %double check values are mapped properly
        disp([LabViewDataVariables{LabViewDataVariableCount,1},'[',LabViewDataVariables{
            LabViewDataVariableCount,3}, '] <-----------> ',LabViewData{1,LabViewDataVariables
            {LabViewDataVariableCount,2}}]);
        %assign the variable
        eval([LabViewDataVariables{LabViewDataVariableCount,1},'=str2num(LabViewData{2,
            LabViewDataVariables{LabViewDataVariableCount,2}});']);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%convert variables to SI units
UpstreamAverageTemperature=(UpstreamAverageTemperature-32)*5/9+273;      %K
AmbientTemperature=(AmbientTemperature-32)*5/9+273;      %K
AtmosphericPressure=AtmosphericPressure/(1.4504*10^-4); %Pa
FoilHeatFlux=(FoilVoltageDrop*FoilCurrent)/(FoilWidth*FoilLength)           %W/m^2
mjets=mjets*.4536;                     %kg/s
```

A.2 Experimental Post Processing Scripts

A.2.1 plotmultiplecases.m

```
clc;
clear;
close all;
clear global

%add folder to the path where additional scripts are located
```

```
7  addpath ../ExperimentalDataCopy/impingement/analysis/
8
9  %define paths to read data from
10 BasePath='../../';
11 CaseListDataFile='CaseList.txt';
12 ExperimentalData='/Experimental/ExperimentalDataCopy/impingement/';
13 RawExperimentalDataPath=[BasePath,ExperimentalData,'/RawData/'];
14 PathToCaseListDataFile=[BasePath,ExperimentalData,'/analysis/',CaseListDataFile];
15 OutputImageFolder=[BasePath,'/Experimental/post/'];
16
17 %list the case numbers to plot, group them, and define a marker for each grouping
18 CaseNumbers={
19         %'casenumber', first spanwise average figure number/overall average curve number,'
               marker', second spanwise average figure number,'marker', third spanwise average
               figure number, 'marker'
20         '0023R4HD3E1H55',2,'+',2,'p',1,'+'
21         '0022R4HD3E2H55',1,'p',1,'p',1,'o'
22         '0024R8HD3E1H55',2,'o',3,'p',2,'+'
23         '0025R8HD3E2H55',1,'*',4,'p',2,'o'
24         '0027R12HD3E1H55',2,'x',6,'p',3,'+'
25         '0026R12HD3E2H55',1,'.',5,'p',3,'o'
26         '0028R15HD3E1H55',2,'s',7,'p',4,'+'
27         '0029R15HD3E2H55',1,'d',8,'p',4,'o'
28
29         '0031R4HD4E1H55',4,'+',2,'+',5,'+'
30         '0030R4HD4E2H55',3,'p',1,'+',5,'o'
31         '0032R8HD4E1H55',4,'o',3,'+',6,'+'
32         '0033R8HD4E2H55',3,'*',4,'+',6,'o'
33         '0035R12HD4E1H55',4,'x',6,'+',7,'+'
34         '0034R12HD4E2H55',3,'.',5,'+',7,'o'
35         '0036R15HD4E1H55',4,'s',7,'+',8,'+'
36         '0037R15HD4E2H55',3,'d',8,'+',8,'o'
37
38         '0040R4HD5E1H55',6,'+',2,'o',9,'+'
39         '0039R4HD5E2H55',5,'p',1,'o',9,'o'
40         '0041R8HD5E1H55',6,'o',3,'o',10,'+'
41         '0042R8HD5E2H55',5,'*',4,'o',10,'o'
42         '0044R12HD5E1H55',6,'x',6,'o',11,'+'
43         '0043R12HD5E2H55',5,'.',5,'o',11,'o'
44         '0045R15HD5E1H55',6,'s',7,'o',12,'+'
45         '0046R15HD5E2H55',5,'d',8,'o',12,'o'
46         };
47
48 OvarallAverageConfigurationsLabels={
49         %'Configuration Name','marker'
50         'z/D=3 - Double Exit','+'
51         'z/D=3 - Single Exit','p'
52         'z/D=4 - Double Exit','o'
53         'z/D=4 - Single Exit','*'
54         'z/D=5 - Double Exit','x'
55         'z/D=5 - Single Exit','.'
56         };
57
58
59 %add a chance to tweek the plot a little for each case
60 ScaleTweaksFirstSpanwiseAverageFigure={
61         'ylim([min(get(gca,''YTick''))-(max(get(gca,''YTick''))+20)])'
62         ''
63         'ylim([min(get(gca,''YTick''))-(max(get(gca,''YTick''))+20)])'
64         ''
65         'ylim([min(get(gca,''YTick''))-(max(get(gca,''YTick''))+20)])'
66         ''
67         };
68
69 ScaleTweaksSecondSpanwiseAverageFigure={
70         'ylim([min(get(gca,''YTick''))-(max(get(gca,''YTick''))+10)])'
71         ''
72         ''
```

```
        'ylim([min(get(gca,''YTick''))-(max(get(gca,''YTick''))+10)])'
        'ylim([min(get(gca,''YTick''))-(max(get(gca,''YTick''))+10)])'
        ''
        ''
        'ylim([min(get(gca,''YTick''))-(max(get(gca,''YTick''))+10)])'
        };


%first plot spanwise averages of the same exit and z/D
for FirstSpanwiseAverageFigureNumber=1:max(cell2mat(CaseNumbers(:,2)))
        CurrentlyFoundCases=0;
        for CurrentCase=1:size(CaseNumbers,1)
                if CaseNumbers{CurrentCase,2}==FirstSpanwiseAverageFigureNumber
                        CurrentlyFoundCases=CurrentlyFoundCases+1;
                        [OverallAverageNu(CaseNumbers{CurrentCase,2},CurrentlyFoundCases,2),
                            OverallAverageNu(CaseNumbers{CurrentCase,2},CurrentlyFoundCases
                            ,1)]=ProcessDatausingPlotFunction(CaseNumbers{CurrentCase,1},
                            RawExperimentalDataPath,PathToCaseListDataFile,CaseNumbers{
                            CurrentCase,3},'yes',OutputImageFolder,'yes');
                end
        end
        eval(ScaleTweaksFirstSpanwiseAverageFigure{FirstSpanwiseAverageFigureNumber})
        hgexport(gcf,[OutputImageFolder,num2str(FirstSpanwiseAverageFigureNumber),'-
            SpanwiseAverageCommonzD.eps'])
        close all
end




%plot the overall averages
close all
figure('OuterPosition',[30 30 622 500]);
hold all
for CurrentConfiguration=1:size(OverallAverageNu,1)
        plot(squeeze(OverallAverageNu(CurrentConfiguration,:,1)),squeeze(OverallAverageNu(
            CurrentConfiguration,:,2)),'Marker',OvarallAverageConfigurationsLabels{
            CurrentConfiguration,2})
        [~,~,~,currentlegend]=legend;
        currentlegend{1,size(currentlegend,2)+1}=sprintf(OvarallAverageConfigurationsLabels{
            CurrentConfiguration,1});
        legend(currentlegend,'Location','NorthWest');
end
XLabels=[4000,8000,12000,15000];
for x=1:size(XLabels,2)
        XlabelsFormatted{x}=thousands(XLabels(x));
end
set(gca,'XTick',XLabels)
set(gca,'XTickLabel',XlabelsFormatted)
xlim([min(XLabels), max(XLabels)])
xlabel('Reynolds Number')
ylabel('Nusselt Number')
set(gca,'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick','on')
title(['Time and Area Averaged Nusselt Number']);
hgexport(gcf,[OutputImageFolder,'OverallAverageNu.eps'])
close all



%next plot spanwise averages of the same exit and Re
for SecondSpanwiseAverageFigureNumber=1:max(cell2mat(CaseNumbers(:,4)))
        for CurrentCase=1:size(CaseNumbers,1)
                if CaseNumbers{CurrentCase,4}==SecondSpanwiseAverageFigureNumber
                        ProcessDatausingPlotFunction(CaseNumbers{CurrentCase,1},
                            RawExperimentalDataPath,PathToCaseListDataFile,CaseNumbers{
                            CurrentCase,5},'yes',OutputImageFolder);
                end
```

```
131         end
132         eval(ScaleTweaksSecondSpanwiseAverageFigure{SecondSpanwiseAverageFigureNumber})
133         hgexport(gcf,[OutputImageFolder,num2str(SecondSpanwiseAverageFigureNumber),'-
                SpanwiseAverageCommonRe.eps'])
134         close all
135 end
136
137 %next plot spanwise averages of the same Re and z/D
138 for ThirdSpanwiseAverageFigureNumber=1:max(cell2mat(CaseNumbers(:,6)))
139         for CurrentCase=1:size(CaseNumbers,1)
140                 if CaseNumbers{CurrentCase,6}==ThirdSpanwiseAverageFigureNumber
141                         ProcessDatausingPlotFunction(CaseNumbers{CurrentCase,1},
                                RawExperimentalDataPath,PathToCaseListDataFile,CaseNumbers{
                                CurrentCase,7},'yes',OutputImageFolder,'yes','yes');
142                 end
143         end
144         ylim([min(get(gca,'YTick')) (max(get(gca,'YTick'))+5)])
145         hgexport(gcf,[OutputImageFolder,num2str(ThirdSpanwiseAverageFigureNumber),'-
                SpanwiseAverageCommonRezD.eps'])
146         close all
147 end
148
149
150
151 %pause
152
153
154
155
156
157
158
159
160
161
162
163
164
165 %load('../literature/han_bothexits_spanwiseaverage.mat')
166 %hold all
167 %plot(Re4850(:,1)-Re4850(115,1),Re4850(:,2))
168 %[n,n,n,currentlegend]=legend;
169 %currentlegend{1,size(currentlegend,2)+1}=sprintf(['Han - Re=4850 H/D=3\nBoth Exits']);
170 %legend(currentlegend,'Location','EastOutside');
171
172 % plot(Re9550(:,1)-Re9550(118,1),Re9550(:,2))
173 % [n,n,n,currentlegend]=legend;
174 % currentlegend{1,size(currentlegend,2)+1}=sprintf(['Han - Re=9550 H/D=3\nBoth Exits']);
175 % legend(currentlegend,'Location','EastOutside');
```

A.2.2 ProcessDatausingPlotFunction.m

```
1  function [OverallAverageNu,TargetRe,X,Y,D,MinXD,MaxXD,MeasurementDomainHeight,TimeAverageNu,
       Nu]=ProcessDatausingPlotFunction(ExperimentalCaseNumber,RawExperimentalDataPath,
       PathToCaseListDataFile,Marker,PlotResults,OutputImageFolder,zOverDinSpanwiseTitle,
       ReinSpanwiseTitle)
2           global SpanwiseAverageFigure
3
4           disp(' ')
5           disp('------------------------------------------------------------------------')
6           disp(['Starting calculations for case #',ExperimentalCaseNumber])
7
8           %call a script that reads in data from the experimental case list spreadsheet/log
                for the current case
9           ReadExperimentalCaseListSpreadsheet
10
11          %override if debuging to speed up.
12          %LastTimeStep=2
```

```matlab

    %call a script that reads in lab view data for the current caes
    ReadLabViewData

    %call a script which defines all of the alignment parameters
    eval(['CameraAlignment',num2str(CameraAlignmentDate)]);


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %define constants and perform unit conversions

    %define the acceleration due to gravity
    g=9.807;        %m/s^2, Incropera and DeWitt 5th edition

    %specific ideal gas constant for air
    R=287;  %J/(kg*K), anderson, modern compressible flow, page 21

    %specific heat of air at 300 Kelvin
    cp=1007;        %J/(kg*K), Incropera and DeWitt 5th edition

    StefanBoltzmannConstant=5.670*10^-8;     %W/(m^2*K^4), Incropera and DeWitt 5th edition

    %sutherland viscocity law constants, see reference in master thesis writeup
    T0=273;
    S=111;
    mu0=1.716*10^-5;

    %define thermal conductivity of air as a function of temperature at atmospheric
    %pressure (Incropera and DeWitt 5th edition table A.5
    %K, W/(m*K)
    k_T=[
        100,.00934
        150,.0138
        200,.0181
        250,.0223
        300,.0263
        350,.0300
        ];



    %preallocate for speed
    SurfaceTemperature=zeros((LastTimeStep-FirstTimeStep+1),(YBottomEdge-YTopEdge+1),(XRightEdge-XLeftEdge+1));

    %read in all data
    for TimeStep=FirstTimeStep:LastTimeStep
        load([RawExperimentalDataPath,'/',ExperimentalCaseNumber,'/matlab/matlab_',ExperimentalCaseNumber,'_',num2str(TimeStep),'.MAT'], '-mat');
        eval(['SurfaceTemperature(TimeStep,:,:)=(matlab_',ExperimentalCaseNumber,'_',num2str(TimeStep),'([YTopEdge:YBottomEdge],[XLeftEdge:XRightEdge]));'])
        ;
        eval(['clear matlab',ExperimentalCaseNumber,'_',num2str(TimeStep),';']);
    end


    %throw away data outside of the X/D range of interest so that it speeds
    %up computation and also allows for proper averaging
    FoundLeftEdge=0;
    FoundRightEdge=0;
    for LoopXLocation=1:(XRightEdge-XLeftEdge+1)
        if X(LoopXLocation)/D>MinXD
            if FoundLeftEdge==1
                if X(LoopXLocation)/D>MaxXD
                    if FoundRightEdge==0
                        FoundRightEdge=1;
```

```
                                        RightEdgeLocation=LoopXLocation          %edge of the domain of interest (has nothing to do with alignment)
                                    end
                                end
                            else
                                FoundLeftEdge=1;
                                LeftEdgeLocation=LoopXLocation          %edge of the domain of interest (has nothing to do with alignment)
                            end
                        end
            end
            X=X(LeftEdgeLocation:RightEdgeLocation);
            SurfaceTemperature=SurfaceTemperature(:,:,LeftEdgeLocation:RightEdgeLocation);
            Tfilm=(SurfaceTemperature+UpstreamAverageTemperature)/2;          %film temperature on the forced convection side

            %calculate and print out values for reference purposes
            MaxSurfaceTemperature=max(max(max(SurfaceTemperature)))-273      %C
            MinSurfaceTemperature=min(min(min(SurfaceTemperature)))-273      %C

            ActualExperimentalRe            %print the current Reynolds number


            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %perform calculations related to natural convection losses
            %note that correlations for laminar flow with uniform, constant surface temperature are
            %utilized. this is an approximation but considered the best we can do.
            %values for h are computed using the spanwise average surface
            %temperature at each time. using this value of h calculated with the
            %spanwise average surface temperature, the actual surface temperature
            %is then used to calculate the convectiv heat loss as a function of
            %position and time. it is a little messed up of an approach but not
            %sure what else we can really do.
            SpanAveragedSurfaceTemperatureTemporary=squeeze(mean(SurfaceTemperature,2));     %K
            YPositionNatural=zeros(size(SurfaceTemperature));        %preallocate for speed
            SpanAveragedSurfaceTemperature=zeros(size(SurfaceTemperature)); %preallocate for speed
            for LoopYLocation=1:size(SurfaceTemperature,2)
                    SpanAveragedSurfaceTemperature(:,LoopYLocation,:)=SpanAveragedSurfaceTemperatureTemporary;      %set all y values to the spanwise average at each x location and at each time
                    YPositionNatural(:,LoopYLocation,:)=Y(LoopYLocation)+abs(min(Y));        %set the y location at each x value and at each time
            end


            TfilmNatural=(SpanAveragedSurfaceTemperature+AmbientTemperature)/2;      %film temperature on the natural convection side of the foil, K
            beta=1./TfilmNatural;   %expansion coefficient for an ideal gas, 1/K

            DensityNatural=AtmosphericPressure./(R.*TfilmNatural);
            %calculate thermal conductivity based upon the film temperature for the
            %natural convection calculation. do a manual, linear interpolation between 350K and 300K
            kNatural=((k_T(6,2)-k_T(5,2))/(k_T(6,1)-k_T(5,1)))*(TfilmNatural-k_T(6,1))+k_T(6,2);

            alpha=kNatural./(DensityNatural*cp);

            %calculate viscocity using sutherland viscocity law
            RelativeViscocityNatural=(TfilmNatural/T0).^(3/2)*(T0+S)./(TfilmNatural+S);
            ViscocityNatural=mu0*RelativeViscocityNatural;

            nu=ViscocityNatural./DensityNatural;

            Ra=(g*beta.*abs(SpanAveragedSurfaceTemperature-AmbientTemperature).*YPositionNatural
```

```
        .^3)./(alpha.*nu);
133
134     if max(max(max(Ra)))>10^9
135         disp('warning, natural convection is turbulent and correlations don''t apply
                ')
136         pause
137     end
138
139     NuNatural=.387*Ra.^.25;          %bejan, convection heat transfer, third edition,
            page 198
140     hNatural=NuNatural.*kNatural./YPositionNatural;
141
142     %because heat transfer coefficient at y=0 is infinity, just set it
143     %equal to the value right after y=0
144     hNatural(:,size(hNatural,2),:)=hNatural(:,size(hNatural,2)-1,:);
145
146     LossesNatural=hNatural.*(SurfaceTemperature-AmbientTemperature);
147
148     clear hNatural NuNatural Ra nu ViscocityNatural RelativeViscocityNatural alpha
            kNatural DensityNatural beta TfilmNatural                          %clear
            up some memory
149
150     %end of natural convection loss calculations
151     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
152     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
153     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
154
155     %calculate values
156     RadiationLosses=FoilEmissivity.*StefanBoltzmannConstant.*(SurfaceTemperature.^4-
            AmbientTemperature.^4); %W/m^2, radiationlosses
157     h=(FoilHeatFlux-RadiationLosses-LossesNatural)./(SurfaceTemperature-
            UpstreamAverageTemperature);
158
159     %calculate and print out some values for reference purposes
160     MaxNaturalLossPercent=max(max(max(LossesNatural)))/FoilHeatFlux*100
161     MeanNaturalLossPercent=mean(mean(mean(LossesNatural)))/FoilHeatFlux*100
162     MeanNaturalandRadiationLossPercent=mean(mean(mean(LossesNatural+RadiationLosses)))/
            FoilHeatFlux*100
163     MaxRadiationLossPercent=max(max(max(RadiationLosses)))/FoilHeatFlux*100
164     MeanRadiationLossPercent=mean(mean(mean(RadiationLosses)))/FoilHeatFlux*100
165
166     clear RadiationLosses LossesNatural                                %clear up some
            memory
167
168     %calculate thermal conductivity based upon the film temperature
169     %do a manual, linear interpolation between 350K and 300K
170     k=((k_T(6,2)-k_T(5,2))/(k_T(6,1)-k_T(5,1)))*(Tfilm-k_T(6,1))+k_T(6,2);
171     %%%%%%%%%this is not used because it is way too slow in MATLAB and max of Tfilm is
172     %%%%%%%%%less than 350K anyway
173     %k=zeros(size(Tfilm));  %preallocate for speed
174     %for looptime=1:size(Tfilm,1)
175     %       for loopY=1:size(Tfilm,2)
176     %               for loopX=1:size(Tfilm,3)
177     %                       k(looptime,loopY,loopX)=interp1(k_T(:,1),k_T(:,2),Tfilm(
            looptime,loopY,loopX));
178     %               end
179     %       end
180     %end
181
182     clear Tfilm RadiationLosses                                %clear up some memory
183
184     Nu=h*D./k;
185
186     clear h                                    %clear up some memory
187
188     TimeAverageNu=squeeze(mean(Nu));
189     SpanwiseTimeAverageNu=mean(TimeAverageNu);
190
```

```
191         if (exist('PlotResults','var')==1)&&(strcmp(PlotResults,'yes'))
192                 MinYD=-MeasurementDomainHeight/2/D;
193                 MaxYD=MeasurementDomainHeight/2/D;
194                 [ContourFigure,ColorbarHandle]=PlotNuContour(TimeAverageNu,X,Y,D,TargetRe,
                        MinXD,MaxXD,MinYD,MaxYD,0,100,20,HoverD);
195
196                 %CurrentImage = getframe(ContourFigure);
197                 %imwrite(CurrentImage.cdata, [ExperimentalCaseNumber,'.png']); %this
                        command doesn't seem to work right when specifying the output resolution
198
199                 %export as an eps image. for some reason it works okay with this contour
                        plot... not sure why I was having problems before.
200                 hgexport(ContourFigure,[OutputImageFolder,ExperimentalCaseNumber,'.eps']);
                                %this command doesn't seem to work right when specifying
                        any other format than eps? also don't remember why it worked better than
                         the print command?
201
202                 disp(' ')
203                 disp('check out the figure and press any key to continue')
204                 %pause
205                 close(ContourFigure)
206
207                 if ((exist('ReinSpanwiseTitle','var'))&&(strcmp(ReinSpanwiseTitle,'yes')))
                        &&((exist('zOverDinSpanwiseTitle','var'))&&(strcmp(zOverDinSpanwiseTitle
                        ,'yes')))
208                         SpanwiseAverageTitleText=['  Re=',thousands(TargetRe),'  z/D=',
                                num2str(HoverD)];
209                         SpanwiseAverageLegendText=ExitConfiguration;
210                 elseif ((exist('zOverDinSpanwiseTitle','var'))&&(strcmp(
                        zOverDinSpanwiseTitle,'yes')))
211                         SpanwiseAverageTitleText=['  z/D=',num2str(HoverD)];
212                         SpanwiseAverageLegendText=['Re=',thousands(TargetRe)];
213                 else
214                         SpanwiseAverageTitleText=['  Re=',thousands(TargetRe)];
215                         SpanwiseAverageLegendText=['z/D=',num2str(HoverD)];
216                 end
217
218                 [SpanwiseAverageFigure]=PlotNuSpanwiseAverage(SpanwiseTimeAverageNu,X,D,
                        TargetRe,MinXD,MaxXD,Marker,SpanwiseAverageLegendText,
                        SpanwiseAverageTitleText);
219         end
220
221
222         %calculate the overall averaged Nu for the specified domain
223         OverallAverageNu=mean(SpanwiseTimeAverageNu)
224         OveralAverageNuCorrelatedHistorical=OverallNuCorrelation(D,HoverD,mjets,
                ActualExperimentalRe,ExitConfiguration,ChannelWidth,MinXD,MaxXD)
225         PercentDiffferenceOverallNu=(OverallAverageNu/OveralAverageNuCorrelatedHistorical-1)
                *100
226
227 end
```

A.2.3 OverallNuCorrelation.m

```
1  function OveralAverageNuCorrelated=OverallNuCorrelation(D,HoverD,mjets,Re,ExitConfiguration,
       ChannelWidth,MinXD,MaxXD)
2          %try to calculate average heat transfer coefficient using corelations
3          %of Florschuetz et all, 1981
4
5          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6          %text rig geometry that is fixed unless the top plate is changed
7          NumberofColumns=5;
8          xnoverD=3;
9          ynoverD=3;
10         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13         %run specific information
```

```
14          %set weighting for the first column to be .5 if it is a line of symmetry
15          if strcmp(ExitConfiguration,'Single_Exit')
16                  Weighting=[1,1,1,1,1,1,1,1];
17                  InitialCrossFlowWeighting=1;
18                  XValues=(MinXD+2):3:(MaxXD-2);
19          else
20                  Weighting=[.5,1,1,1,1];
21                  InitialCrossFlowWeighting=0;
22                  XValues=0:3:(MaxXD-3);
23          end
24
25          mjet=mjets/55;                          %kg/s
26          NumberofRows=size(Weighting,2);
27
28          %%%%%%%%%%%%%%%%%%%%%%%%%%%%
29          % correlation constants
30          CA=1.18;
31          nxA=-.944;
32          nyA=-.642;
33          nzA=.169;
34
35          Cm=.612;
36          nxm=.059;
37          nym=.032;
38          nzm=-.022;
39
40          CB=.437;
41          nxB=-.095;
42          nyB=-.219;
43          nzB=.275;
44
45          Cn=.092;
46          nxn=-.005;
47          nyn=.599;
48          nzn=1.04;
49
50          A=CA*(xnoverD^nxA)*(ynoverD^nyA)*(HoverD^nzA);
51          m=Cm*(xnoverD^nxm)*(ynoverD^nym)*(HoverD^nzm);
52          B=CB*(xnoverD^nxB)*(ynoverD^nyB)*(HoverD^nzB);
53          n=Cn*(xnoverD^nxn)*(ynoverD^nyn)*(HoverD^nzn);
54          %end correlation constants
55          %%%%%%%%%%%%%%%%%%%%%%%%%%%%
56
57          %%%%%%%%%%%%%%%%%%%%%%%%%%%%
58          %other constants
59          Pr=.7;
60          %%%%%%%%%%%%%%%%%%%%%%%%%%%%
61
62
63          %%%%%%%%%%%%%%%%%%%%%%%%%%%%
64          %calculations
65          Anozzle=(D/2)^2*pi;
66          rhoVjet=mjet/Anozzle;                   %not sure if this is calculated
                correctly according to the paper
67          Achannel=D*HoverD*ChannelWidth;
68          rhoVchannel=cumsum(mjet*NumberofColumns*[InitialCrossFlowWeighting,Weighting(1:(
                NumberofRows-1))])/Achannel;    %not sure if this is calculated correctly
                according to the paper
69          AverageNu=A*Re^m*(1-B*(HoverD*rhoVchannel/rhoVjet).^n)*Pr^(1/3);
70
71          %hold all
72          %bar(XValues,AverageNu,1,'FaceColor','none','LineWidth',2)
73          OveralAverageNuCorrelated=sum(AverageNu.*Weighting)/sum(Weighting);
74          %%%%%%%%%%%%%%%%%%%%%%%%%%%%
75
76
77  end
```

A.2.4 CameraAlignment20100824.m

```



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Camera Alignment
%note, this alignment process was performed by aligning to the nozzle plate
%surface, and then the top plate surface, and then interpolating the foil
%surface position based upon the spacer thickness. there may be some
%discrepancy in this interpolation process (it may not be linear), but the
%difference in alignment is not huge, so it is not considered a big deal
%
%see the following spreadsheets for more information
%       2010.08.24 - alignment via hole pattern - final.xls
%       2010.08.24 - alignment of foil surface - final.xls

%width based upon the nominal/design hole patter dimensions
MeasurementDomainWidth=0.562*.0254*11;   %m
MeasurementDomainHeight=0.562*.0254*5;   %m

%mapping of the nozzle plate surface
XCenter=160;     %pixel
YCenter=108;     %pixel
NozzePlateLeftVerticalEdge=25;
NozzePlateBottomHorizontalEdge=169;

%mapping of the test section top/upper plate outer surface
TopPlateLeftVerticalEdge=15;
TopPlateBottomHorizontalEdge=174;

%actual measured thickness of the top plate (not the design thickness)
TopPlateThickness=.794*25.4;     %mm

%taken from the mapping from the nozzle plate surface to the top plate
%surface back to the foil surface
XLeftEdge=round((NozzePlateLeftVerticalEdge-TopPlateLeftVerticalEdge)/(TopPlateThickness+
    ActualH)*ActualH+TopPlateLeftVerticalEdge)
YBottomEdge=round((NozzePlateBottomHorizontalEdge-TopPlateBottomHorizontalEdge)/(
    TopPlateThickness+ActualH)*ActualH+TopPlateBottomHorizontalEdge)

%assume the centering of the camera was correct
XRightEdge=XLeftEdge+(XCenter-XLeftEdge)*2
YTopEdge=YBottomEdge-(YBottomEdge-YCenter)*2

%Interpolate all values based upon the mapping defined abolve
X=interp1([XLeftEdge,XCenter],[-MeasurementDomainWidth/2,0],[XLeftEdge:XRightEdge],'linear',
    'extrap');
Y=interp1([YBottomEdge,YCenter],[-MeasurementDomainHeight/2,0],[YTopEdge:YBottomEdge],'
    linear','extrap');
%end of camera alignment
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.2.5 GenerateExperimentalNuAnimation.m

```
clc;
clear;
close all;
clear global

%add folder to the path where additional scripts are located
addpath ../ExperimentalDataCopy/impingement/analysis/

%define paths to read data from
BasePath='../../';
CaseListDataFile='CaseList.txt';
ExperimentalData='/Experimental/ExperimentalDataCopy/impingement/';
RawExperimentalDataPath=[BasePath,ExperimentalData,'/RawData/'];
```

```
PathToCaseListDataFile=[BasePath,ExperimentalData,'/analysis/',CaseListDataFile];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
%define the current case
%CaseNumberSelected='0022R4HD3E2H55'
%CaseNumberSelected='0029R15HD3E2H55'
%CaseNumberSelected='0023R4HD3E1H55'
CaseNumberSelected='0028R15HD3E1H55'

HoverD=3; %need to hard code this in because never made ProcessDatausingPlotFunction return HoverD
FramesPerSecond=500/11.42       %frames/second, need to hard code this in because it was never recorded in the case list spreadsheet
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


OutputImageFolder=[BasePath,'/Experimental/post/',CaseNumberSelected,'-SurfaceNuTimeSteps/'];



[OverallAverageNu,TargetRe,X,Y,D,MinXD,MaxXD,MeasurementDomainHeight,TimeAverageNu,Nu]=ProcessDatausingPlotFunction(CaseNumberSelected,RawExperimentalDataPath,PathToCaseListDataFile,'+','no');

mkdir([OutputImageFolder])                          %make the directory if it doesn't already exist
delete([OutputImageFolder,'/*.*'])        %delete any files that were in the directory, if it already existed


for TimeStep=1:size(Nu,1)
        [ContourFigure,ColorbarHandle]=PlotNuContour(squeeze(Nu(TimeStep,:,:)),X,Y,D,TargetRe,MinXD,MaxXD,-MeasurementDomainHeight/2/D,MeasurementDomainHeight/2/D,0,100,20,HoverD,'Nu',(TimeStep-1)/FramesPerSecond);

        CurrentImage = getframe(ContourFigure);
        imwrite(CurrentImage.cdata, [OutputImageFolder,'/',CaseNumberSelected,'-',num2str(TimeStep,'%05.0f'),'.png']);  %this command doesn't seem to work right when specifying the output resolution
        close(ContourFigure)
end



%first clear out an environmental variable matlab sets that messes this program up
LD_LIBRARY_PATH_original=getenv('LD_LIBRARY_PATH');
setenv('LD_LIBRARY_PATH','');

%delete the video if one has already been attempted for the same time steps
delete([OutputImageFolder,'/../',CaseNumberSelected,'-SurfaceNu-1-',num2str(TimeStep),'.mp4'])
delete([OutputImageFolder,'/../',CaseNumberSelected,'-SurfaceNu-1-',num2str(TimeStep),'.mpg'])

%now generate a video in two formats
eval(['!ffmpeg -r ',num2str(FramesPerSecond,3),' -sameq -i ',OutputImageFolder,'/',CaseNumberSelected,'-%05d.png',' ',OutputImageFolder,'/../',CaseNumberSelected,'-SurfaceNu-1-',num2str(TimeStep),'.mp4'])
eval(['!ffmpeg -r ',num2str(FramesPerSecond,3),' -sameq -i ',OutputImageFolder,'/',CaseNumberSelected,'-%05d.png',' ',OutputImageFolder,'/../',CaseNumberSelected,'-SurfaceNu-1-',num2str(TimeStep),'.mpg'])

%now change the environmental variable back
setenv('LD_LIBRARY_PATH',LD_LIBRARY_PATH_original)
```

A.3 CFD Preprocessing Scripts

A.3.1 BatchWriter.m

```
clc;
clear all;
clear global;
close all;


CFDCase='CFD0059'









BasePath='../../';
RawMeshes='RawMeshes';
RawMacros='RawMacros';
ExperimentalData='/Experimental/ExperimentalDataCopy/impingement/';
CaseListDataFile='CaseList.txt';

CFDCases='CompletedCFDCases';
OutputFolder='local_workspace';
UploadFolder='hawk_archive_SmallFiles-outbox';


%general paths
PathToRawMeshes=[BasePath,RawMeshes,'/Serial/'];
PathToRawMacros=[BasePath,RawMacros,'/'];
PathToOutputFolder=[BasePath,OutputFolder,'/'];
PathToUploadFolder=[BasePath,UploadFolder,'/'];
RawExperimentalDataPath=[BasePath,ExperimentalData,'/RawData/'];
PathToCaseListDataFile=[BasePath,ExperimentalData,'/analysis/',CaseListDataFile];









%begin generic code
PathToWorkingDirectory=[PathToOutputFolder,CFDCase,'/'];
if exist(PathToWorkingDirectory,'dir')==7
        disp('folder already exists and you requested to start a new case delete or rename
            case first')
        break
else
        mkdir(PathToWorkingDirectory)
        mkdir([PathToWorkingDirectory,'/output/images/Residuals/'])
        %note: the rest of the image folders are generated below when the macro script is
            written
        mkdir([PathToWorkingDirectory,'/output/csv/'])
        mkdir([PathToWorkingDirectory,'/output/cas-dat/'])
        mkdir([PathToWorkingDirectory,'/output/status/'])
        mkdir([PathToWorkingDirectory,'/input/'])
        mkdir([PathToWorkingDirectory,'/post/'])
end









```

```
%include some code that reads in data from spreadsheets
addpath /home/andy/Desktop/CFD/CFDScripts/       %this is used because the run command changes the path and messes all other paths defined up
addpath /home/andy/Desktop/CFD/Experimental/ExperimentalDataCopy/impingement/analysis/
addpath /home/andy/Desktop/CFD/Experimental/scripts/
ReadCFDCaseListSpreadsheet       %note: can't put a .m on the end of the sript name because it thinks the . is an operator or something
ReadExperimentalCaseListSpreadsheet

eval(['!cp ./BatchWriter.m ',PathToWorkingDirectory,'/input/'])
eval(['!cp ',PathToCFDCaseListDataFilePrefix,'.* ',PathToWorkingDirectory,'/input/'])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%generate journal file
journalfile=fopen([PathToWorkingDirectory,'/input/journalfileMATLAB.jou'],'w');

%print out the date and time for reference
fprintf(journalfile,['!date',char(10)]);
%print out bandwidth information for reference
fprintf(journalfile,['/parallel/bandwidth',char(10)]);

%set settings that have to be set every time fluent is launched
fprintf(journalfile,['/display/set/windows/text/company_no',char(10)]);
fprintf(journalfile,['/display/set/colors/color-scheme_classic',char(10)]);

reply = input('Start running in Fluent (1) locally inside of MATLAB or generate a batch submission script for hawk (2)?[2]: ', 's');
if isempty(reply)
    reply = '2';
end

%if running on hawk, start writing the batch script
if strcmp(reply,'1')==0
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
        %generate batch script for submitting job to DSRC
        BatchScriptFile=fopen([PathToWorkingDirectory,'/input/',CFDCase,'.BatchSubmission'],'w');

        fprintf(BatchScriptFile,['#!/bin/csh -x',char(10)]);
        fprintf(BatchScriptFile,['#PBS -l job_type=''MPI''',char(10)]);

        %this is a really stupid thing to have to do but no shell variables work for these parameters
        fprintf(BatchScriptFile,['#PBS -o hawk-0:andy/BatchScriptStandardErrorAndOutput/',CFDCase,'.StandardOut',char(10)]);
        fprintf(BatchScriptFile,['#PBS -e hawk-0:andy/BatchScriptStandardErrorAndOutput/',CFDCase,'.StandardError',char(10)]);

        fprintf(BatchScriptFile,['#PBS -A WPFAFPRW29132P33',char(10)]);

        fprintf(BatchScriptFile,['#PBS -m abe',char(10)]);
        fprintf(BatchScriptFile,['#PBS -M andy@schroder.net',char(10)]);

        fclose(BatchScriptFile);

        eval(['!cp ',PathToWorkingDirectory,'/input/',CFDCase,'.BatchSubmission ',PathToWorkingDirectory,'/input/',CFDCase,'.DebugBatchSubmission']);

        BatchScriptFile=fopen([PathToWorkingDirectory,'/input/',CFDCase,'.BatchSubmission'],'a');
        fprintf(BatchScriptFile,['#PBS -N aus',CFDCase,char(10)]);
        fprintf(BatchScriptFile,['#PBS -l select=ncpus=',num2str(NumberofCPUs+2),char(10)]);
        fprintf(BatchScriptFile,['#PBS -l fluent=',num2str(NumberofCPUs),char(10)]);
        fprintf(BatchScriptFile,['#PBS -l walltime=',WallTime,char(10)]);
        fprintf(BatchScriptFile,['#PBS -q regular',char(10)]);
        fprintf(BatchScriptFile,['set FluentCPUs=',num2str(NumberofCPUs),char(10)]);      %this is hoaky but makes the creation of two similiar batch scripts easier
        fprintf(BatchScriptFile,['set ExtraText=""',char(10)]);
```

```
121         fclose(BatchScriptFile);
122
123         BatchScriptFileDebug=fopen([PathToWorkingDirectory,'/input/',CFDCase,'.
                DebugBatchSubmission'],'a');
124         fprintf(BatchScriptFileDebug,['#PBS -N dbg',CFDCase,char(10)]);
125         fprintf(BatchScriptFileDebug,['#PBS -l select=ncpus=22',char(10)]);
126         fprintf(BatchScriptFileDebug,['#PBS -l fluent=21',char(10)]);
127         fprintf(BatchScriptFileDebug,['#PBS -l walltime=50:00',char(10)]);
128         fprintf(BatchScriptFileDebug,['#PBS -q debug',char(10)]);
129         fprintf(BatchScriptFileDebug,['set FluentCPUs=20',char(10)]);   %this is hoaky but
                makes the creation of two similiar batch scripts easier
130         fprintf(BatchScriptFileDebug,['set ExtraText="Debug-"',char(10)]);
131         fclose(BatchScriptFileDebug);
132
133         PartialBatchScriptFile=fopen([PathToWorkingDirectory,'/input/',CFDCase,'.
                PartialBatchSubmission'],'w');
134
135         fprintf(PartialBatchScriptFile,['echo "starting ',CFDCase,'" | mail -s "starting ',
                CFDCase,' - $ExtraText$PBS_JOBID" andy@schroder.net',char(10)]);
136         fprintf(PartialBatchScriptFile,['module load CFD/fluent12.1.2',char(10)]);       %not
                sure if this does anything but terry says to try it anyway
137         fprintf(PartialBatchScriptFile,['limit stacksize unlimited',char(10)]); %not sure if
                this does anything but terry says to try it anyway
138         fprintf(PartialBatchScriptFile,['mkdir -p $WORK_DIR/',CFDCase,'/output',char(10)]);
                %tag on the output so that the standard output file can actually be written
                somewhere.
139         fprintf(PartialBatchScriptFile,['cd $WORK_DIR/',CFDCase,char(10)]);
140         fprintf(PartialBatchScriptFile,['date >>&"./output/',CFDCase,'-$ExtraText$PBS_JOBID.
                FluentStandardOutput"',char(10)]);
141         fprintf(PartialBatchScriptFile,['/usr/bin/rcp -r ${msas}:inbox/SmallFiles/',CFDCase,
                '.input.tar $WORK_DIR >>&"./output/',CFDCase,'-$ExtraText$PBS_JOBID.
                FluentStandardOutput"',char(10)]);
142         fprintf(PartialBatchScriptFile,['tar -C $WORK_DIR -xf $WORK_DIR/',CFDCase,'.input.
                tar >>&"./output/',CFDCase,'-$ExtraText$PBS_JOBID.FluentStandardOutput"',char
                (10)]);
143 end
144
145 %now see how to copy files.
146 %note, don't think this section can't be combined with if strcmp(CFDCaseToContinue,'N/A')
        statements below because wanted to write the batch submission script at the same time as
        copying files
147 %(just to keep things together) and don't always need to copy files so if you are going to
        do both at the same time so if you are going to do it together then do it before
        starting to write the
148 %journal file
149 if (strcmp(reply,'1'))&&(strcmp(CFDCaseToContinue,'N/A')==0)
150         %if not a starting from a fresh mesh and running locally or files are no longer on
                the hawk workspace then do a local copy and upload manually
151         eval(['!cp ',CFDCaseToContinuePathOutputData,'/output/cas-dat/',DataFileName,' ',
                PathToWorkingDirectory,'/input/']);
152         eval(['!cp ',CFDCaseToContinuePathOutputData,'/output/cas-dat/',CaseFileName,' ',
                PathToWorkingDirectory,'/input/']);
153         eval(['!cp ',CFDCaseToContinuePathSmallFiles,'/input/*.macro ',
                PathToWorkingDirectory,'/input/']);
154         eval(['!cp -r ',CFDCaseToContinuePathSmallFiles,'/output/images ',
                PathToWorkingDirectory,'/output/']);
155         eval(['!cp -r ',CFDCaseToContinuePathSmallFiles,'/output/csv ',
                PathToWorkingDirectory,'/output/']);
156         eval(['!cp -r ',CFDCaseToContinuePathSmallFiles,'/output/status ',
                PathToWorkingDirectory,'/output/']);
157 elseif (strcmp(CFDCaseToContinue,'N/A')==0)
158         %if not starting from a fresh mesh and files are on hawk workspace and not running
                locally
159         %copy the required files using the batch script
160
161         fprintf(journalfile,['!/usr/bin/rcp -r ${msas}:CompletedCFDCases/SmallFiles/',
                CFDCaseToContinue,'.tar $WORK_DIR',char(10)]);% >>&"./output/',CFDCase,'-
                $ExtraText$PBS_JOBID.FluentStandardOutput"',char(10)]);
```

```
162         fprintf(journalfile,['!tar -C $WORK_DIR -xf $WORK_DIR/',CFDCaseToContinue,'.tar',
                char(10)]);% >>&"./output/',CFDCase,'-$ExtraText$PBS_JOBID.FluentStandardOutput
                "',char(10)]);
163
164         fprintf(journalfile,['!/usr/bin/rcp -r ${msas}:CompletedCFDCases/OutputData/',
                CFDCaseToContinue,'/output/cas-dat/',CaseFileName,' ./input/',char(10)]);%
                >>&"./output/',CFDCase,'-$ExtraText$PBS_JOBID.FluentStandardOutput"',char(10)]);
165         fprintf(journalfile,['!/usr/bin/rcp -r ${msas}:CompletedCFDCases/OutputData/',
                CFDCaseToContinue,'/output/cas-dat/',DataFileName,' ./input/',char(10)]);%
                >>&"./output/',CFDCase,'-$ExtraText$PBS_JOBID.FluentStandardOutput"',char(10)]);
166
167         fprintf(journalfile,['!cp $WORK_DIR/',CFDCaseToContinue,'/input/*.macro ./input/',
                char(10)]);% >>&"./output/',CFDCase,'-$ExtraText$PBS_JOBID.FluentStandardOutput
                "',char(10)]);
168         fprintf(journalfile,['!cp -r $WORK_DIR/',CFDCaseToContinue,'/output/images ./output/
                ',char(10)]);% >>&"./output/',CFDCase,'-$ExtraText$PBS_JOBID.
                FluentStandardOutput"',char(10)]);
169         fprintf(journalfile,['!cp -r $WORK_DIR/',CFDCaseToContinue,'/output/csv ./output/',
                char(10)]);% >>&"./output/',CFDCase,'-$ExtraText$PBS_JOBID.FluentStandardOutput
                "',char(10)]);
170         fprintf(journalfile,['!rm -r $WORK_DIR/',CFDCaseToContinue,char(10)]);%'>>&"./output
                /',CFDCase,'-$ExtraText$PBS_JOBID.FluentStandardOutput"',char(10)]);
171         %print out the date and time for reference
172         fprintf(journalfile,['!date',char(10)]);
173         eval(['!cp -r ',CFDCaseToContinuePathSmallFiles,'/output/status ',
                PathToWorkingDirectory,'/output/']);
174 elseif (strcmp(reply,'1'))
175         %if running locally and a new mesh, get it out of the local mesh folder
176         %copy some files
177         eval(['!cp ',PathToRawMeshes,'/',CaseFileName,'/',CaseFileName,'.cas ',
                PathToWorkingDirectory,'/input/'])                %add .cas for file copy since
                it isn't included in the variable for new meshes
178
179         %only do this for running locally since on hawk need to do as a two step process
                because of the license server being stupid
180         WriteMultiCoreMesh=fopen([PathToWorkingDirectory,'/input/WriteMultiCoreMesh.jou'],'w
                ');
181         fprintf(WriteMultiCoreMesh,['/file/read-case ./input/',CaseFileName,char(10)]);
                                                                                %see comment
                about no .cas in this variable for new meshes
182         %convert the foil to a solid, even though this was already defined as solid in
                pointwise. for simulations run on hawk,
183         %this is done with the make parallel mesh script because need to know the boundary
                names in order to run. when running locally,
184         %need to just cancel the script after the first run and then reupdate the
                spreadsheet with the correct boundary names, then run again.
185         if strcmp(WallConduction,'yes')
186                 fprintf(WriteMultiCoreMesh,['/define/boundary-conditions/modify-zones/zone-
                        type foil solid',char(10)]);
187         end
188         fprintf(WriteMultiCoreMesh,['/file/write-case ./input/',CaseFileName,' yes',char(10)
                ]);
189         fprintf(WriteMultiCoreMesh,['exit',char(10)]);
190         fclose(WriteMultiCoreMesh);
191 else
192         %if running on hawk and a new mesh, get it out of the archives
193         fprintf(journalfile,['!/usr/bin/rcp -r ${msas}:RawMeshes/Parallel/',CaseFileName,'/'
                ,CaseFileName,'.cas ./input/',char(10)]);% >>&"./output/',CFDCase,'-
                $ExtraText$PBS_JOBID.FluentStandardOutput"',char(10)]);
194 end
195
196 disp('If you want to manually remove any status files now to force restarting at');
197 disp('earlier point in the simulation, do that now and then press any key to continue');
198 disp('if doing this make sure to delete any images or csv files generated. csv files will');
199 disp('be appended to and images maybe will not be overwritten?, so things will turn out very
        weird');
200 disp('press any key to continue');
201 pause
```

```
202
203 %now finish writing the batch script if running on hawk
204 if strcmp(reply,'1')==0
205         fprintf(journalfile,['!date',char(10)]);% >>&"./output/',CFDCase,'-
                $ExtraText$PBS_JOBID.FluentStandardOutput"',char(10)]);
206
207         %now doing this separately and not automatically because the license server is
                stupid.
208         %if strcmp(CFDCaseToContinue,'N/A')      %if this is a raw mesh then you need to also
                 read in the mesh and resave it because of a multi-cpu bug with fluent or
                pointwise.
209         %       fprintf(journalfile,['!fluent 3ddp -gu -driver null -i ./input/
                WriteMultiCoreMesh.jou',char(10)]);% >>&"./output/',CFDCase,'-
                $ExtraText$PBS_JOBID.FluentStandardOutput"',char(10)]);
210         %end
211
212         fprintf(PartialBatchScriptFile,['fluent -paltix -mpi=sgi 3ddp -gu -t $FluentCPUs -
                driver null -i ./input/journalfileMATLAB.jou >>&"./output/',CFDCase,'-
                $ExtraText$PBS_JOBID.FluentStandardOutput"',char(10)]);
213         fprintf(PartialBatchScriptFile,['cat "./output/',CFDCase,'-$ExtraText$PBS_JOBID.
                FluentStandardOutput"| mail -s "complete ',CFDCase,' - $ExtraText$PBS_JOBID"
                andy@schroder.net',char(10)]);
214         fclose(PartialBatchScriptFile);
215         eval(['!cat ',PathToWorkingDirectory,'/input/',CFDCase,'.PartialBatchSubmission>>',
                PathToWorkingDirectory,'/input/',CFDCase,'.DebugBatchSubmission']);
216         eval(['!cat ',PathToWorkingDirectory,'/input/',CFDCase,'.PartialBatchSubmission>>',
                PathToWorkingDirectory,'/input/',CFDCase,'.BatchSubmission']);
217         eval(['!rm ',PathToWorkingDirectory,'/input/',CFDCase,'.PartialBatchSubmission']);
218 end
219
220
221
222 %read in necessary files
223 fprintf(journalfile,['/file/read-case ./input/',CaseFileName,char(10)]);
224 %note for the new mesh the .cas was not included in the CaseFileName variable because
        variable is needed to reference the file and folder.
225 %fluent automatically appends this if not present
226 fprintf(journalfile,['/file/read-macros ./input/save_figures.macro',char(10)]);
227 fprintf(journalfile,['/file/read-macros ./input/save_residuals.macro',char(10)]);
228 fprintf(journalfile,['/file/read-macros ./input/save_current_residuals.macro',char(10)]);
229
230 if strcmp(CFDCaseToContinue,'N/A')       %setup the case from a raw mesh exported from
        pointwise.
231
232         ReadLabViewData          %include some code that reads labview data and calculates a
                few derived quantities
233         minlet=mjets/55*NumberofHolesinMesh;
234         ApproximateInletDensity=AtmosphericPressure/(287.058*UpstreamAverageTemperature);
235         ApproximateInletVelocity=(mjets/55)/(ApproximateInletDensity*pi*(D/2)^2)
236
237         %convert values to meters and into a string
238         %num2str seems to add a lot of extra spaces when it converts the vector to a string
                but fluent doesn't seem to have a problem with it.
239         %need to have 10 digits of precision so the line coordinates don't get rounded to a
                value outside of the meshes actual domain limits
240         LinePlotCoordinatesStart=num2str(LinePlotCoordinatesStart/1000,20);
241         LinePlotCoordinatesStop=num2str(LinePlotCoordinatesStop/1000,20);
242
243         %copy macros out of the raw macros folder
244         eval(['!cp ',PathToRawMacros,'* ',PathToWorkingDirectory,'/input/'])
245
246         %add any other lines or surfaces to the mesh or do any other modifications on the
                mesh
247         %convert from mm to m
248         fprintf(journalfile,['/mesh/scale .001 .001 .001',char(10)]);
249         %add line
250         fprintf(journalfile,['/surface/line-surface radial-line1 ',LinePlotCoordinatesStart,
                ' ',LinePlotCoordinatesStop,char(10)]);
```

```
251
252         %don't think this is necessary since can't export in parallel mode, so might as well
                just do it during post processing.
253         %add a plane through the center for exporting data from
254         %fprintf(journalfile,['/surface/plane CenterPlane 0 0 0 0 1 1 0 -1 1',char(10)]);
255
256         %set some solver settings
257         fprintf(journalfile,['/define/models/viscous/kw-standard yes',char(10)]);
258         fprintf(journalfile,['/define/models/viscous/kw-shear-correction yes',char(10)]);
259         fprintf(journalfile,['/report/reference-values/length .005',char(10)]);
260         %reduce the under relaxation factors so that the solution won't blow up when started
                and
261         %so that the residuals are reduced. reducing under relaxation factors doesn't change
                a steady solution
262         %it just helps to stabilize it while slowing the convergence process. if a solution
                is steady and has some numerical instability then the residuals will go down
263         %if the solution is unsteady, the residuals will go down because the reduced under
                relaxation factors not only stabilize the numerical
264         %instabilities but also dampen out the true unsteadyness, causing the residuals to
                go down.
265         %because just trying to get a good initial condition for the time accurate unsteady
                simulation, it is okay to reduce these
266         %values even though it is actually damping the unsteadiness.
267         %also note that even though the convergence process is slowed, it it stabilizes it,
                it does actually help it to converge faster as long as it is not
268         %overly under relaxed.
269         fprintf(journalfile,['/solve/set/under-relaxation/turb-viscosity .9',char(10)]);
270
271         % set residual settings
272         fprintf(journalfile,['/solve/monitors/residual/plot yes',char(10)]);
273         fprintf(journalfile,['/solve/monitors/residual/print yes',char(10)]);
274         fprintf(journalfile,['/solve/monitors/residual/check-convergence yes yes yes yes yes
                yes',char(10)]);
275         fprintf(journalfile,['/solve/monitors/residual/convergence-criteria 1e-06 1e-06 1e
                -06 1e-06 1e-06 1e-06',char(10)]);
276         fprintf(journalfile,['/solve/monitors/residual/n-save 10000',char(10)]);
277         %note, after the buffer is filled, fluent throws 1/2 of all the points away and then
                 starts recordign again, so this isn't the real x axis range but rather the
                number of points in the x axis that are stored
278         fprintf(journalfile,['/solve/monitors/residual/n-display 5000',char(10)]);      %
                only plot the last 5000 iterations. this doesn't really control the x axis very
                well but at least it limits it somewhat.
279         fprintf(journalfile,['/solve/monitors/residual/scale-by-coefficient yes',char(10)]);
280         %fprintf(journalfile,['/plot/residuals-set/auto-scale yes no -8 0',char(10)]);  %
                keep the scale the same so each image can be looked at in succession
                meaningfully.... fluent lies and doesn't tell you it wants the exponent of 10,
                also, this line was moved to the macro files because it needs to be done
                everytime if not autoscalling and also ploting the radial line plots
281         %note the following commands will give a invalid command yes error until the energy
                equation is turned on
282         fprintf(journalfile,['/solve/execute-commands/add-edit residuals 1000 "iteration" "
                save_residuals_to_png"',char(10)]);
283         fprintf(journalfile,['/solve/execute-commands/add-edit currentresiduals 10 "
                iteration" "save_current_residuals_to_png"',char(10)]);
284
285         %set the gauge pressure
286         fprintf(journalfile,['/define/operating-conditions/operating-pressure ',num2str(
                AtmosphericPressure),char(10)]);
287
288         %set boundary conditions
289         %inlet - mass flow inlet
290         fprintf(journalfile,['/define/boundary-conditions/zone-type ',InletZone,' mass-flow-
                inlet',char(10)]);
291         fprintf(journalfile,['/define/boundary-conditions/mass-flow-inlet ',InletZone,' yes
                yes no ',num2str(minlet),' no 0 no yes no no no yes ',num2str(
                InletTurbulentIntensity),' ',num2str(D),char(10)]);
292
293         %boundary conditions for a pressure inlet (not yet tested thoroughly) and still need
```

```
        to define GageTotalPressure
294     %fprintf(journalfile,['/define/boundary-conditions/zone-type ',InletZone,' pressure-
            inlet',char(10)]);
295     %fprintf(journalfile,['/define/boundary-conditions/pressure-inlet ',InletZone,' yes
            no ',num2str(GageTotalPressure),' no 0 no yes no no no yes ',num2str(
            InletTurbulentIntensity),' ',num2str(D),char(10)]);
296
297     %repeating boundary
298     if 1~=strcmp(FirstRepeatingBoundary,'')
299         fprintf(journalfile,['/mesh/modify-zones/make-periodic ',
                FirstRepeatingBoundary,' ',SecondRepeatingBoundary,' no yes yes',char
                (10)]);
300     end
301     %note, the heat surface is turned on later after the flow develops a little
302
303     %initialize the flow field
304     fprintf(journalfile,['/solve/initialize/initialize-flow',char(10)]);
305     fprintf(journalfile,['/solve/set/reporting-interval 10',char(10)]);
306     %fprintf(journalfile,['/solve/patch ',JetCore,' () z-velocity yes -',num2str(
            ApproximateInletVelocity),char(10)]);      %did a test and it actually converges
            slower with this defined.
307
308     %set image file export settings
309     fprintf(journalfile,['/display/set/picture/driver/png',char(10)]);
310     fprintf(journalfile,['/display/set/picture/x-resolution/1500',char(10)]);
311     fprintf(journalfile,['/display/set/picture/y-resolution/1000',char(10)]);
312     fprintf(journalfile,['/display/set/picture/color-mode/color',char(10)]);
313
314     %first do some iterations without any heating so that the flow can develope.
            otherwise, the energy equation may diverge,
315     %just as the test rig will burn up the foil if you turn the air and the heat on at
            the same time.
316     %also create a status file so know what is going on
317     fprintf(journalfile,['!touch ./output/status/InitalizingFlowWithNoHeat.status',char
            (10)]);
318     fprintf(journalfile,['/solve/iterate ',NoEnergyIterations,' yes yes yes',char(10)]);
            %extra yes gives error if not appending but its okay.
319
320     %turn on the energy equation and change to an ideal gas model
321     fprintf(journalfile,['/define/models/energy yes no no no yes',char(10)]);
322     fprintf(journalfile,['/define/materials/change-create air air yes ideal-gas no no no
            no no no',char(10)]);
323     fprintf(journalfile,['/define/models/viscous/kw-compressibility yes',char(10)]);
324     %change the under relaxation factor for the energy equation
325     fprintf(journalfile,['/solve/set/under-relaxation/temperature .5',char(10)]);    %.9
            seems to work but .3 seems to do better convergence with a first order solution
            because it helps stabilize it. using .5 as a balance. may want to change this on
            unsteady solutions.
326
327     %now define the heater boundary condition
328
329
330
331
332     if strcmp(WallConduction,'yes')
333         %define the stainless steel material. fluent is stupid and it only lets you
                create a new material by first copying an old material and then changing
                everything
334         %all properties are assumed constant and may not actually be the same allow
                as we used in the experiment
335         %properties are taken from MATWEB for T 300 Series Stainless steel. Average
                values were used.
336         %foil is modeled directly rather than using the simpler wall thickness
                option in fluent, because want 3D conduction rather than 1D conduction.
337         fprintf(journalfile,['/define/materials/change-create aluminum stainless-
                steel yes constant 7840 yes constant 498 yes constant 15.2 no',char(10)
                ]);
338         %define the volumetric heat generation. not sure what the final parameter "
```

```
                deactivate thread" is, so leave it as the default of "no"
339             %right now the thickness of the foil is hardcoded in. probably should start
                    loging this in the CaseList.xls spreadsheet.
340             %assumes thickness of the foil is uniform
341             fprintf(journalfile,['/define/boundary-conditions/solid foil yes stainless-
                    steel yes 1 yes ',num2str(FoilHeatFlux/(.0015*.0254)),' no yes 0 0 0 0 0
                    1 no',char(10)]);
342         else
343             fprintf(journalfile,['/define/boundary-conditions/zone-type ',
                    HeatedSurfaceZone1,' wall',char(10)]);
344             fprintf(journalfile,['/define/boundary-conditions/wall ',HeatedSurfaceZone1,
                    ' 0 no 0 no no no ',num2str(FoilHeatFlux),' no no no no 0 no .5',char
                    (10)]);
345             fprintf(journalfile,['/define/boundary-conditions/zone-type ',
                    HeatedSurfaceZone2,' wall',char(10)]);
346             fprintf(journalfile,['/define/boundary-conditions/wall ',HeatedSurfaceZone2,
                    ' 0 no 0 no no no ',num2str(FoilHeatFlux),' no no no no 0 no .5',char
                    (10)]);
347         end
348
349         %now define the inlet temperature
350         %mass flow inlet
351         fprintf(journalfile,['/define/boundary-conditions/mass-flow-inlet ',InletZone,' yes
                yes no ',num2str(minlet),' no ',num2str(UpstreamAverageTemperature),' no 0 no
                yes no no no yes ',num2str(InletTurbulentIntensity),' ',num2str(D),char(10)]);
352
353         %boundary conditions for a pressure inlet (not yet tested thoroughly) and still need
                to define GageTotalPressure
354         %fprintf(journalfile,['/define/boundary-conditions/pressure-inlet ',InletZone,' yes
                no ',num2str(GageTotalPressure),' no ',num2str(UpstreamAverageTemperature),' no
                0 no yes no no no yes ',num2str(InletTurbulentIntensity),' ',num2str(D),char(10)
                ]);
355
356         %now adjust the residual settings for the energy equation
357         fprintf(journalfile,['/solve/monitors/residual/check-convergence yes yes yes yes yes
                yes yes',char(10)]);
358         fprintf(journalfile,['/solve/monitors/residual/convergence-criteria 1e-06 1e-06 1e
                -06 1e-06 1e-06 1e-06 1e-06',char(10)]);
359
360         %now set surface monitor settings
361         %need to make sure you have an extra yes command for each after the iterate commands
                to allow for appending if the case is continued.
362         fprintf(journalfile,['/solve/monitors/surface/set-monitor
                Average_Surface_Temperature "Area-Weighted Average" temperature ',
                HeatedSurfaceZone1,' ',HeatedSurfaceZone2,' () no yes yes "./output/csv/
                AverageSurfaceTemperature.csv" 1',char(10)]);
363         fprintf(journalfile,['/solve/monitors/surface/set-monitor Average_Nusselt_Number "
                Area-Weighted Average" nusselt-number ',HeatedSurfaceZone1,' ',
                HeatedSurfaceZone2,' () no yes yes "./output/csv/AverageNusseltNumber.csv" 1',
                char(10)]);
364         fprintf(journalfile,['/solve/monitors/surface/set-monitor Vertex_Max_Temperature "
                Vertex Maximum" temperature ',HeatedSurfaceZone1,' ',HeatedSurfaceZone2,' () no
                yes yes "./output/csv/VertexMaximumSurfaceTemperature.csv" 1',char(10)]);
365
366         %auto save the figures to a file
367         fprintf(journalfile,['/solve/execute-commands/add-edit savefigures 10 "iteration" "
                save_figures_to_png"',char(10)]);
368
369         %configure auto save of data file
370         fprintf(journalfile,['/file/auto-save/case-frequency if-mesh-is-modified',char(10)])
                ;   %this still seems to save one copy and i am not sure why.
371         fprintf(journalfile,['/file/auto-save/data-frequency 100',char(10)]);
372         fprintf(journalfile,['/file/auto-save/retain-most-recent-files yes',char(10)]);
373         fprintf(journalfile,['/file/auto-save/max-files 2',char(10)]);
374         fprintf(journalfile,['/file/auto-save/root-name ./output/cas-dat/',CFDCase,char(10)
                ]);
375         fprintf(journalfile,['/file/data-file-options nusselt-number heat-transfer-coef
                velocity-magnitude vorticity-mag ()',char(10)]);          %must be after data is
```

```
        initialized?
376
377     %print out a summary
378     fprintf(journalfile,['/report/summary yes ./output/',CFDCase,'-InputSummary.txt',
            char(10)]);
379
380     %save the case that was just configured
381     fprintf(journalfile,['/file/write-case ./output/cas-dat/',CFDCase,'.cas',char(10)]);
382     fprintf(journalfile,['/file/write-data ./output/cas-dat/',CFDCase,'.dat',char(10)]);
383
384     %create a status file so know where to restart the case at
385     fprintf(journalfile,['!touch ./output/status/CaseSetup.status',char(10)]);
386
387     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
388     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
389     %take a break from writing the journal file and
390     %generate macro script for ploting convergence
391     savefiguresmacrofile=fopen([PathToWorkingDirectory,'/input/save_figures.macro'],'w')
            ;
392
393     fprintf(savefiguresmacrofile,['(cx-macro-define',char(10)]);
394     fprintf(savefiguresmacrofile,['''((save_figures_to_png .',char(10)]);
395     fprintf(savefiguresmacrofile,'")');%open parentheses
396
397     %surface contour zoomed out (includes unheated regions)
398     fprintf(savefiguresmacrofile,['/display/set-window 1',char(10)]);
399     fprintf(savefiguresmacrofile,['/display/set/contours/surfaces ',HeatedSurfaceZone1,'
            ',HeatedSurfaceZone2,' ',UnHeatedBottomSurfaceZone,' ()',char(10)]);
400     fprintf(savefiguresmacrofile,['/display/set/contours/node-values yes',char(10)]);
401     fprintf(savefiguresmacrofile,['/display/set/contours/n-contour 100',char(10)]);
402     fprintf(savefiguresmacrofile,['/display/set/contours/filled-contours yes',char(10)])
            ;
403     fprintf(savefiguresmacrofile,['/display/set/colors/skip-label 10',char(10)]);
404     fprintf(savefiguresmacrofile,['/display/set/windows/scale/format \\"%%0.2f\\"',char
            (10)]);
405     %/display/set/colo-map gray
406     fprintf(savefiguresmacrofile,['/display/contour/temperature ',MinPlotTemperature,' '
            ,MaxPlotTemperature,char(10)]);
407     fprintf(savefiguresmacrofile,['/display/views/restore-view front',char(10)]);
408     fprintf(savefiguresmacrofile,['/display/views/auto-scale',char(10)]);
409     fprintf(savefiguresmacrofile,['/display/save-picture \\"./output/images/
            SurfaceTemperatureZoomedOut/SurfaceTemperatureZoomedOut-%%i.png\\" yes',char(10)
            ]);
410     mkdir([PathToWorkingDirectory,'/output/images/SurfaceTemperatureZoomedOut/'])
411
412     %surface contour zoomed in (heated regions only)
413     fprintf(savefiguresmacrofile,['/display/set-window 1',char(10)]);
414     fprintf(savefiguresmacrofile,['/display/set/contours/surfaces ',HeatedSurfaceZone1,'
            ',HeatedSurfaceZone2,' ()',char(10)]);
415     fprintf(savefiguresmacrofile,['/display/set/contours/node-values yes',char(10)]);
416     fprintf(savefiguresmacrofile,['/display/set/contours/n-contour 100',char(10)]);
417     fprintf(savefiguresmacrofile,['/display/set/contours/filled-contours yes',char(10)])
            ;
418     fprintf(savefiguresmacrofile,['/display/set/colors/skip-label 10',char(10)]);
419     fprintf(savefiguresmacrofile,['/display/set/windows/scale/format \\"%%0.2f\\"',char
            (10)]);
420     %/display/set/colo-map gray
421     fprintf(savefiguresmacrofile,['/display/contour/temperature ',MinPlotTemperature,' '
            ,MaxPlotTemperature,char(10)]);
422     fprintf(savefiguresmacrofile,['/display/views/restore-view front',char(10)]);
423     fprintf(savefiguresmacrofile,['/display/views/auto-scale',char(10)]);
424     fprintf(savefiguresmacrofile,['/display/save-picture \\"./output/images/
            SurfaceTemperatureZoomedIn/SurfaceTemperatureZoomedIn-%%i.png\\" yes',char(10)])
            ;
425     mkdir([PathToWorkingDirectory,'/output/images/SurfaceTemperatureZoomedIn/'])
426
427     %radial line
428     fprintf(savefiguresmacrofile,['/display/set-window 1',char(10)]);
```

```
429         fprintf(savefiguresmacrofile,['/plot/solution-set/numbers float 3 float 2',char(10)
                ]);
430         fprintf(savefiguresmacrofile,['/plot/solution-set/auto-scale yes yes',char(10)]);
431         fprintf(savefiguresmacrofile,['/plot/solution-set/auto-scale yes no ',
                MinPlotTemperature,' ',MaxPlotTemperature,char(10)]);
432         fprintf(savefiguresmacrofile,['/plot/solution-set/label \"X Position\" \"\"',
                char(10)]);
433         fprintf(savefiguresmacrofile,['/plot/solution-set/lines 0 \"center\" 1 \"
                foreground\" no',char(10)]);
434         %this doesn't work on multi processor because fluent can do the arclength after
                partitioned
435         %/plot/plot no \"\" no no temperature no yes no radial-line1 ()
436         fprintf(savefiguresmacrofile,['/plot/plot yes \"\" no no temperature yes 1 0 0
                radial-line1 ()',char(10)]);
437         fprintf(savefiguresmacrofile,['/display/save-picture \"./output/images/
                SurfaceTemperatureRadial1/SurfaceTemperatureRadial1-%%i.png\" yes',char(10)]);
438         mkdir([PathToWorkingDirectory,'/output/images/SurfaceTemperatureRadial1/'])
439
440 %skip this for now
441 %       %radial line zoomed in
442 %       fprintf(savefiguresmacrofile,['/display/set-window 1',char(10)]);
443 %       fprintf(savefiguresmacrofile,['/plot/solution-set/numbers float 3 float 2',char(10)
        ]);
444 %       fprintf(savefiguresmacrofile,['/plot/solution-set/auto-scale yes no ',
        MinPlotTemperature,' ',MaxPlotTemperature,char(10)]);
445 %       fprintf(savefiguresmacrofile,['/plot/solution-set/label \"X Position\" \"\"',
        char(10)]);
446 %       fprintf(savefiguresmacrofile,['/plot/solution-set/lines 0 \"center\" 1 \"
        foreground\" no',char(10)]);
447 %       %this doesn't work on multi processor because fluent can do the arclength after
        partitioned
448 %       %/plot/solution-set/auto-scale no 0 .01 yes
449 %       %/plot/plot no \"\" no no temperature no yes no radial-line1 ()
450 %       fprintf(savefiguresmacrofile,['/plot/solution-set/auto-scale no -.095 -.085 yes',
        char(10)]);
451 %       fprintf(savefiguresmacrofile,['/plot/plot yes \"\" no no temperature yes 1 0 0
        radial-line1 ()',char(10)]);
452 %       fprintf(savefiguresmacrofile,['/display/save-picture \"./output/images/
        SurfaceTemperatureRadial1-zoomed/SurfaceTemperatureRadial1-zoomed-%%i.png\"',char(10)])
        ;
453 %       mkdir([PathToWorkingDirectory,'/output/images/SurfaceTemperatureRadial1-zoomed/'])
454
455
456         %close out macro
457         fprintf(savefiguresmacrofile,['"')',char(10)]);
458         fprintf(savefiguresmacrofile,['   ))',char(10)]);
459         fclose(savefiguresmacrofile);
460
461 else    %continue from a previously running solution
462         fprintf(journalfile,['/file/read-data ./input/',DataFileName,char(10)]);
463         fprintf(journalfile,['/file/auto-save/root-name ./output/cas-dat/',CFDCase,char(10)
                ]);
464         fprintf(journalfile,['/file/write-case ./output/cas-dat/',CFDCase,'.cas',char(10)]);
                %this is really a waste of disk space but it makes the continuation of
            previous cases much simpler.
465 end
466
467 if strcmp(CFDCaseToContinue,'N/A')|(exist([PathToWorkingDirectory,'./output/status/
        FirstOrderSteadyShouldBeConverged.status'],'file')~=2)
468         %beging solving
469         fprintf(journalfile,['/solve/iterate ',FirstOrderIterations,' yes yes yes',char(10)
                ]);%extra yes gives error if not appending but its okay.
470
471         %save after converged
472         fprintf(journalfile,['/file/write-data ./output/cas-dat/',CFDCase,'-
                FirstOrderSteadyShouldBeConverged',char(10)]);
473         %these commands don't seem to work in fluent parallel mode
474         %fprintf(journalfile,['/file/export/ascii ./output/csv/',CFDCase,'-HeatedSurface-
```

```
        FirstOrderSteadyShouldBeConverged.csv ',HeatedSurfaceZone1,' ',
        HeatedSurfaceZone2,' () no temperature pressure wall-shear y-plus () no yes',
        char(10)]);
475     %fprintf(journalfile,['/file/export/ascii ./output/csv/',CFDCase,'-CenterPlane-
        FirstOrderSteadyShouldBeConverged.csv CenterPlane () no temperature pressure x-
        velocity y-velocity z-velocity () no yes',char(10)]);
476
477     %create a status file so know where to restart the case at
478     fprintf(journalfile,['!touch ./output/status/FirstOrderSteadyShouldBeConverged.
        status',char(10)]);
479 end
480
481
482 if (strcmp(Unsteady,'yes')~=1)
483     %this code was used to do an steady second order convergence. it doesn't really work
         well though because under relaxation factors need
484     %to be made very low in order to get "reasonable" convergence. it doesn't make a
        whole lot oc sense to do this
485     %second order convergence if the under relaxation factors need to be reduced anyway.
         guessing that the first order steady convergence solution
486     %is probably good enough of an initial condition that once switch over to second
        order unsteady, won't need to spend a hole lot of time getting to a
487     %cyclical steady state condition.
488
489     if strcmp(CFDCaseToContinue,'N/A')|(exist([PathToWorkingDirectory,'./output/status/
        SecondOrderSteadyShouldBeConverged.status'],'file')~=2)      %if you want to
        continue doing second order steady after the previous case did you may need to
        rename this file. otherwise the job will just stop right away.
490         %change over to a second order solver after converged first order
491         fprintf(journalfile,['/solve/set/discretization-scheme/density 1',char(10)]
                );
492         fprintf(journalfile,['/solve/set/discretization-scheme/mom 1',char(10)]);
493         fprintf(journalfile,['/solve/set/discretization-scheme/pressure 12',char(10)
                ]);
494         fprintf(journalfile,['/solve/set/discretization-scheme/k 1',char(10)]);
495         fprintf(journalfile,['/solve/set/discretization-scheme/omega 1',char(10)]);
496         fprintf(journalfile,['/solve/set/discretization-scheme/temperature 1',char
                (10)]);
497
498         %change convergence settings since it isn't going to converge second order
                as well since the flow is unsteady
499         %fprintf(journalfile,['/solve/monitors/residual/convergence-criteria .5e-04
                .5e-04 .5e-04 .5e-04 .5e-04 .5e-04 .5e-04',char(10)]);
500
501
502     else
503         %disp('solution is converged second order steady, nothing to do');
504         %if it is already converged, turn off residual convergenc checking, and
                iterate just to make sure it is really converged.
505         fprintf(journalfile,['/solve/monitors/residual/check-convergence no no no no
                no no no',char(10)]);
506     end
507
508
509     %set new autosave file name
510     fprintf(journalfile,['/file/auto-save/root-name ./output/cas-dat/',CFDCase,'-
        SecondOrderSteady',char(10)]);
511
512     %save the new case file
513     fprintf(journalfile,['/file/write-case ./output/cas-dat/',CFDCase,'-
        SecondOrderSteady',char(10)]);
514
515     %beging solving again
516     fprintf(journalfile,['/solve/iterate ',SecondOrderIterations,' yes yes yes',char(10)
        ]);%extra yes gives error if not appending but its okay.
517
518     %save after converged as good as you can get.
519     fprintf(journalfile,['/file/write-data ./output/cas-dat/',CFDCase,'-
```

```
        SecondOrderSteadyShouldBeConverged',char(10)]);
520     %these commands don't seem to work in fluent parallel mode
521     %fprintf(journalfile,['/file/export/ascii ./output/csv/',CFDCase,'-HeatedSurface-
            SecondOrderSteadyShouldBeConverged.csv ',HeatedSurfaceZone1,' ',
            HeatedSurfaceZone2,' () no temperature pressure wall-shear y-plus () no yes',
            char(10)]);
522     %fprintf(journalfile,['/file/export/ascii ./output/csv/',CFDCase,'-CenterPlane-
            SecondOrderSteadyShouldBeConverged.csv CenterPlane () no temperature pressure x-
            velocity y-velocity z-velocity () no yes',char(10)]);
523
524     %create a status file so know where to restart the case at
525     fprintf(journalfile,['!touch ./output/status/SecondOrderSteadyShouldBeConverged.
            status',char(10)]);
526 else
527     if strcmp(CFDCaseToContinue,'N/A')|(exist([PathToWorkingDirectory,'./output/status/
            SecondOrderUnsteadyBegin.status'],'file')~=2)
528         %change over to a second order solver after converged first order
529         fprintf(journalfile,['/solve/set/discretization-scheme/density 1',char(10)]
                ;
530         fprintf(journalfile,['/solve/set/discretization-scheme/mom 1',char(10)]);
531         fprintf(journalfile,['/solve/set/discretization-scheme/pressure 12',char(10)
                ]);
532         fprintf(journalfile,['/solve/set/discretization-scheme/k 1',char(10)]);
533         fprintf(journalfile,['/solve/set/discretization-scheme/omega 1',char(10)]);
534         fprintf(journalfile,['/solve/set/discretization-scheme/temperature 1',char
                (10)]);
535
536         %switch over to an unsteady simulation
537         fprintf(journalfile,['/define/models/unsteady-2nd-order yes',char(10)]);
538         fprintf(journalfile,['/solve/set/time-step ',UnsteadyTimeStep,char(10)]);
539         fprintf(journalfile,['/solve/set/data-sampling yes 1 yes yes yes',char(10)]
                ;
540         fprintf(journalfile,['/solve/initialize/init-flow-statistics',char(10)]);
541
542         %now set the data file quantities. fluent is dumb and you need to do 1
                iteration in order for it to realize they exist.
543         fprintf(journalfile,['/solve/dual-time-iterate 1 1 yes yes yes yes yes yes',
                char(10)]);
544         fprintf(journalfile,['/file/data-file-options yes nusselt-number heat-
                transfer-coef velocity-magnitude vorticity-mag mean-nusselt-number mean-
                heat-transfer-coef mean-velocity-magnitude mean-temperature ()',char(10)
                ]);
545
546         %set surface monitor settings to do every time step instead of iteration.
547         fprintf(journalfile,['/solve/monitors/surface/set-monitor
                Average_Surface_Temperature_TimeStep "Area-Weighted Average" temperature
                ',HeatedSurfaceZone1,' ',HeatedSurfaceZone2,' () no yes yes "./output/
                csv/AverageSurfaceTemperatureTimeStep.csv" 1 yes flow-time',char(10)]);
548         fprintf(journalfile,['/solve/monitors/surface/set-monitor
                Average_Nusselt_Number_TimeStep "Area-Weighted Average" nusselt-number '
                ,HeatedSurfaceZone1,' ',HeatedSurfaceZone2,' () no yes yes "./output/csv
                /AverageNusseltNumberTimeStep.csv" 1 yes flow-time',char(10)]);
549         fprintf(journalfile,['/solve/monitors/surface/set-monitor
                Vertex_Max_Temperature_TimeStep "Vertex Maximum" temperature ',
                HeatedSurfaceZone1,' ',HeatedSurfaceZone2,' () no yes yes "./output/csv/
                VertexMaximumSurfaceTemperatureTimeStep.csv" 1 yes flow-time',char(10)])
                ;
550         %now setup surface monitors of time averaged quantities
551         fprintf(journalfile,['/solve/monitors/surface/set-monitor
                Mean_Average_Surface_Temperature_TimeStep "Area-Weighted Average" mean-
                temperature ',HeatedSurfaceZone1,' ',HeatedSurfaceZone2,' () no yes yes
                "./output/csv/MeanAverageSurfaceTemperatureTimeStep.csv" 1 yes flow-time
                ',char(10)]);
552         fprintf(journalfile,['/solve/monitors/surface/set-monitor
                Mean_Average_Nusselt_Number_TimeStep "Area-Weighted Average" mean-
                nusselt-number ',HeatedSurfaceZone1,' ',HeatedSurfaceZone2,' () no yes
                yes "./output/csv/MeanAverageNusseltNumberTimeStep.csv" 1 yes flow-time'
                ,char(10)]);
```

```
553                 fprintf(journalfile,['/solve/monitors/surface/set-monitor␣
                        Mean_Vertex_Max_Temperature_TimeStep␣"Vertex␣Maximum"␣mean-temperature␣'
                        ,HeatedSurfaceZone1,'␣',HeatedSurfaceZone2,'␣()␣no␣yes␣yes␣"./output/csv
                        /MeanVertexMaximumSurfaceTemperatureTimeStep.csv"␣1␣yes␣flow-time',char
                        (10)]);
554
555                 %set animation to every time step
556                 fprintf(journalfile,['/solve/execute-commands/add-edit␣savefigures␣1␣"time-
                        step"␣"save_figures_to_png"',char(10)]);
557
558                 %set new autosave settings
559                 if NumberOfFormerTimeStepsToSave==0
560                         fprintf(journalfile,['/file/auto-save/retain-most-recent-files␣no',
                                char(10)]);
561                 else
562                         fprintf(journalfile,['/file/auto-save/retain-most-recent-files␣yes',
                                char(10)]);
563                         fprintf(journalfile,['/file/auto-save/max-files␣',num2str(
                                NumberOfFormerTimeStepsToSave),char(10)]);
564                 end
565                 fprintf(journalfile,['/file/auto-save/data-frequency␣1',char(10)]);
566                 fprintf(journalfile,['/file/auto-save/root-name␣./output/cas-dat/',CFDCase,'
                        -SecondOrderUnSteady',char(10)]);
567
568                 %these commands don't seem to work in fluent parallel mode
569                 %set autosaving of data at each time step
570                 %fprintf(journalfile,['/file/transient-export/ascii ./output/csv/',CFDCase
                        ,'-HeatedSurface-Unsteady ',HeatedSurfaceZone1,' ',HeatedSurfaceZone2,'
                        () temperature pressure wall-shear y-plus () no no HeatedSurface 1 time-
                        step',char(10)]);
571                 %fprintf(journalfile,['/file/transient-export/ascii ./output/csv/',CFDCase
                        ,'-CenterPlane-Unsteady CenterPlane () temperature pressure x-velocity y
                        -velocity z-velocity () no no CenterPlane 1 time-step',char(10)]);
572
573
574                 %save the new case file
575                 fprintf(journalfile,['/file/write-case␣./output/cas-dat/',CFDCase,'-
                        SecondOrderUnSteady',char(10)]);
576
577                 %create a status file so know where to restart the case at
578                 fprintf(journalfile,['!touch␣./output/status/SecondOrderUnsteadyBegin.status
                        ',char(10)]);
579         elseif strcmp(ResetUnsteadyStatistics,'yes')
580                 fprintf(journalfile,['/solve/initialize/init-flow-statistics',char(10)]);
581         end
582
583         %this duplicate command shouldn't be necessary but there seems to be a bug in fluent
                so you have to reissue it every time you load the case file because it won't
                save
584         %set animation to every time step
585         fprintf(journalfile,['/solve/execute-commands/add-edit␣savefigures␣1␣"time-step"␣"
                save_figures_to_png"',char(10)]);
586
587         %start solving
588         fprintf(journalfile,['/solve/dual-time-iterate␣',UnsteadyTimeSteps,'␣',
                SecondOrderIterations,'␣yes␣yes␣yes␣yes␣yes␣yes␣yes␣yes␣yes',char(10)]);
589 end
590
591 fprintf(journalfile,['exit',char(10)]);
592 fclose(journalfile);
593
594
595
596
597
598
599
600
```

```
disp('last chance to modify any files before completion, then press any key to continue');
pause;



%if running locally, then run.
if strcmp(reply,'1')
        setenv('FLUENTLM_LICENSE_FILE','27016@mentis.engineering.uc.edu');
        eval(['cd ',PathToWorkingDirectory])
        if strcmp(CFDCaseToContinue,'N/A')      %if this is a raw mesh then you need to also read in the mesh and resave it because of a multi-cpu bug with fluent or pointwise.
            eval(['!/usr/ansys_inc/v121/fluent/bin/fluent 3ddp -gu -driver null -i ./input/WriteMultiCoreMesh.jou']);
        end
        %if want to use less memory use single precision and single processor
        eval(['!/usr/ansys_inc/v121/fluent/bin/fluent 3ddp -gu -t4 -ssh -cnf=127.0.0.1 -driver null -i ./input/journalfileMATLAB.jou'])
else
        %tar up the file
        eval(['!tar -C ',PathToOutputFolder,' -cf ',PathToUploadFolder,'/',CFDCase,'.input.tar ',CFDCase])

        %make a copy of the batch submission script outside of the tarfile so it can actually be submitted
        eval(['!cp ',PathToWorkingDirectory,'/input/',CFDCase,'.BatchSubmission ',PathToUploadFolder,'/']);
        eval(['!cp ',PathToWorkingDirectory,'/input/',CFDCase,'.DebugBatchSubmission ',PathToUploadFolder,'/']);
end
```

A.4 CFD Post Processing Scripts

A.4.1 ExtractHeatedSurfaceBatchWriter.m

```
clc;
clear all;
clear global;
close all;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0045'
% ExportCaseFileName='CFD0045-SecondOrderUnSteady.cas'
% DataFileNamePrefix='CFD0045-SecondOrderUnSteady'
% FormerHawkJobNumber='';
% StartingTimeStepNumber=788
% EverXXxStepsSaved=1
% EndingTimeStepNumber=819
% SliceNumberofCPUs=8;
% SliceWallTime='21:00:00'
% queue='regular'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0047'
% SliceCaseFileName='CFD0047-SecondOrderUnSteady.cas'
% FormerHawkJobNumber='315072.HAWK';
% StartingTimeStepNumber=270
% EverXXxStepsSaved=1
% EndingTimeStepNumber=1000      %can make this a really high number because it isn't really going to hurt anything if the files aren't all there yet
% SliceNumberofCPUs=8;
% SliceWallTime='24:00:00'
% queue='regular'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0051'
% ExportCaseFileName='CFD0051.cas'                %this variable can't be called CaseFileName because that is repopulated by ReadSpreadsheets
% DataFileNamePrefix='CFD0051'
% FormerHawkJobNumber='319180.HAWK';
% StartingTimeStepNumber=261
% EverXXxStepsSaved=1
% EndingTimeStepNumber=519       %can make this a really high number because it isn't really going to hurt anything if the files aren't all there yet
% SliceNumberofCPUs=12;
% SliceWallTime='48:00:00'       %does about 10 per hour
% queue='regular'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0046'
% ExportCaseFileName='CFD0046-SecondOrderUnSteady.cas'
% DataFileNamePrefix='CFD0046-SecondOrderUnSteady'
% FormerHawkJobNumber='';
% StartingTimeStepNumber=151
% EverXXxStepsSaved=1
% EndingTimeStepNumber=1000
% SliceNumberofCPUs=10;
% SliceWallTime='110:00:00'
% queue='regular'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0052'
% ExportCaseFileName='CFD0052-SecondOrderUnSteady.cas'
% DataFileNamePrefix='CFD0052-SecondOrderUnSteady'
% FormerHawkJobNumber='321153.HAWK';
% StartingTimeStepNumber=323
% EverXXxStepsSaved=1
% EndingTimeStepNumber=1000
% SliceNumberofCPUs=10;
% SliceWallTime='5:00:00'
% queue='regular'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0048'
% ExportCaseFileName='CFD0048-SecondOrderUnSteady.cas'
% DataFileNamePrefix='CFD0048-SecondOrderUnSteady'
% FormerHawkJobNumber='315945.HAWK';
% StartingTimeStepNumber=184
% EverXXxStepsSaved=1
% EndingTimeStepNumber=260
% SliceNumberofCPUs=10;
% SliceWallTime='26:00:00'
% queue='regular'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0053'
% ExportCaseFileName='CFD0053-SecondOrderSteady.cas'
% SliceDataFileName='CFD0053-SecondOrderSteadyShouldBeConverged'               %don't include .dat
% FormerHawkJobNumber='';
% SliceNumberofCPUs=10;
% SliceWallTime='1:00:00'
% queue='debug'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% CFDCase='CFD0020'
% ExportCaseFileName='CFD0020-SecondOrderSteady.cas'
% SliceDataFileName='CFD0020-SecondOrderSteadyShouldBeConverged'                    %don't include .dat
% FormerHawkJobNumber='';
% SliceNumberofCPUs=10;
% SliceWallTime='1:00:00'
% queue='debug'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0025'
% ExportCaseFileName='CFD0025-SecondOrderSteady.cas'
% SliceDataFileName='CFD0025-SecondOrderSteadyShouldBeConverged'                    %don't include .dat
% FormerHawkJobNumber='';
% SliceNumberofCPUs=10;
% SliceWallTime='1:00:00'
% queue='debug'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0029'
% ExportCaseFileName='CFD0029-SecondOrderSteady.cas'
% SliceDataFileName='CFD0029-SecondOrderSteadyShouldBeConverged'                    %don't include .dat
% FormerHawkJobNumber='';
% SliceNumberofCPUs=10;
% SliceWallTime='3:00:00'
% queue='regular'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0031'
% ExportCaseFileName='CFD0031-SecondOrderSteady.cas'
% SliceDataFileName='CFD0031-SecondOrderSteady-1-18700'          %don't include .dat
% FormerHawkJobNumber='';
% SliceNumberofCPUs=10;
% SliceWallTime='1:00:00'
% queue='debug'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0036'
% ExportCaseFileName='CFD0036-SecondOrderSteady.cas'
% SliceDataFileName='CFD0036-SecondOrderSteadyShouldBeConverged'                    %don't include .dat
% FormerHawkJobNumber='';
% SliceNumberofCPUs=10;
% SliceWallTime='1:00:00'
% queue='debug'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0041'
% ExportCaseFileName='CFD0041-SecondOrderSteady.cas'
% SliceDataFileName='CFD0041-SecondOrderSteadyShouldBeConverged'                    %don't include .dat
% FormerHawkJobNumber='';
% SliceNumberofCPUs=10;
% SliceWallTime='1:00:00'
% queue='debug'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
158 % CFDCase='CFD0024'
159 % ExportCaseFileName='CFD0024-SecondOrderSteady.cas'
160 % SliceDataFileName='CFD0024-SecondOrderSteady-1-25000'          %don't include .dat
161 % FormerHawkJobNumber='';
162 % SliceNumberofCPUs=10;
163 % SliceWallTime='1:00:00'
164 % queue='debug'
165 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
166
167 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
168 % CFDCase='CFD0037'
169 % ExportCaseFileName='CFD0037-SecondOrderSteady.cas'
170 % SliceDataFileName='CFD0037-SecondOrderSteadyShouldBeConverged'                %don't
        include .dat
171 % FormerHawkJobNumber='';
172 % SliceNumberofCPUs=10;
173 % SliceWallTime='1:00:00'
174 % queue='debug'
175 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
176
177 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
178 % CFDCase='CFD0043'
179 % ExportCaseFileName='CFD0043-SecondOrderSteady.cas'
180 % SliceDataFileName='CFD0043-SecondOrderSteadyShouldBeConverged'                %don't
        include .dat
181 % FormerHawkJobNumber='';
182 % SliceNumberofCPUs=10;
183 % SliceWallTime='1:00:00'
184 % queue='debug'
185 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
186
187
188 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
189 % CFDCase='CFD0056'
190 % ExportCaseFileName='CFD0056-SecondOrderSteady.cas'
191 % SliceDataFileName='CFD0056-SecondOrderSteadyShouldBeConverged'                %don't
        include .dat
192 % FormerHawkJobNumber='322198.HAWK';
193 % SliceNumberofCPUs=10;
194 % SliceWallTime='1:00:00'
195 % queue='debug'
196 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
197
198 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
199 % CFDCase='CFD0054'
200 % ExportCaseFileName='CFD0054-SecondOrderSteady.cas'
201 % SliceDataFileName='CFD0054-SecondOrderSteadyShouldBeConverged'                %don't
        include .dat
202 % FormerHawkJobNumber='';
203 % SliceNumberofCPUs=10;
204 % SliceWallTime='3:00:00'
205 % queue='regular'
206 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
207
208
209
210 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
211 % CFDCase='CFD0059'
212 % ExportCaseFileName='CFD0059.cas'
213 % DataFileNamePrefix='CFD0059'          %don't include .dat
214 % FormerHawkJobNumber='326193.HAWK';
215 % StartingTimeStepNumber=920
216 % EverXXxStepsSaved=1
217 % EndingTimeStepNumber=2000
218 % SliceNumberofCPUs=10;
219 % SliceWallTime='24:00:00'
220 % queue='regular'
221 % % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CFDCase='CFD0057'
% ExportCaseFileName='CFD0057-SecondOrderSteady.cas'
% SliceDataFileName='CFD0057-SecondOrderSteadyShouldBeConverged'                    %don't include .dat
% FormerHawkJobNumber='';
% SliceNumberofCPUs=10;
% SliceWallTime='3:00:00'
% queue='regular'
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
CFDCase='CFD0055'
ExportCaseFileName='CFD0055-SecondOrderSteady.cas'
SliceDataFileName='CFD0055-SecondOrderSteadyShouldBeConverged'              %don't include .dat
FormerHawkJobNumber='';
SliceNumberofCPUs=10;
SliceWallTime='3:00:00'
queue='regular'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%










% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%begin generic code
CFDCases='CompletedCFDCases';
BasePath='../../';
CaseListDataFile='CaseList.txt';
ExperimentalData='/Experimental/ExperimentalDataCopy/impingement/';
RawExperimentalDataPath=[BasePath,ExperimentalData,'/RawData/'];
PathToCaseListDataFile=[BasePath,ExperimentalData,'/analysis/',CaseListDataFile];
OutputFolder='/local_workspace/OutputDataSlices/';
PathToOutputFolder=[BasePath,OutputFolder,'/'];
PathToWorkingDirectory=[PathToOutputFolder,CFDCase,'/'];

mkdir([PathToWorkingDirectory,'/input/']) %make the directory if it doesn't already exist
delete([PathToWorkingDirectory,'/input/*.*'])    %delete any files that were in the directory, if it already existed, even though they should all be overwritten anyway.


%include some code that reads in data from spreadsheets
addpath /home/andy/Desktop/CFD/CFDScripts/       %this is used because the run command changes the path and messes all other paths defined up
addpath /home/andy/Desktop/CFD/Experimental/ExperimentalDataCopy/impingement/analysis/
addpath /home/andy/Desktop/CFD/Experimental/scripts/

ReadCFDCaseListSpreadsheet        %note: can't put a .m on the end of the sript name because it thinks the . is an operator or something
ReadExperimentalCaseListSpreadsheet



BatchScriptFile=fopen([PathToWorkingDirectory,'/input/',CFDCase,'-ExportSlices.BatchSubmission'],'w');
fprintf(BatchScriptFile,['#!/bin/csh -x',char(10)]);
fprintf(BatchScriptFile,['#PBS -l job_type=''MPI''',char(10)]);
fprintf(BatchScriptFile,['#PBS -o hawk-0:andy/BatchScriptStandardErrorAndOutput/',CFDCase,'-OutputDataSlices.StandardOut',char(10)]);
```

```
282 fprintf(BatchScriptFile,['#PBS -e hawk-0:andy/BatchScriptStandardErrorAndOutput/',CFDCase,'-
        OutputDataSlices.StandardError',char(10)]);
283 fprintf(BatchScriptFile,['#PBS -A WPFAFPRW29132P33',char(10)]);
284 fprintf(BatchScriptFile,['#PBS -m abe',char(10)]);
285 fprintf(BatchScriptFile,['#PBS -M andy@schroder.net',char(10)]);
286 fprintf(BatchScriptFile,['#PBS -N slc',CFDCase,char(10)]);
287 fprintf(BatchScriptFile,['#PBS -l select=ncpus=',num2str(SliceNumberofCPUs),char(10)]);
288 fprintf(BatchScriptFile,['#PBS -l fluent=1',char(10)]);
289 fprintf(BatchScriptFile,['#PBS -l walltime=',SliceWallTime,char(10)]);
290 fprintf(BatchScriptFile,['#PBS -q ',queue,char(10)]);
291 fprintf(BatchScriptFile,['module load CFD/fluent12.1.2',char(10)]);
292 fprintf(BatchScriptFile,['limit stacksize unlimited',char(10)]);
293 fprintf(BatchScriptFile,['mkdir -p $WORK_DIR/',CFDCase,'/output/csv/HeatedSurface',char(10)
        ]);
294 fprintf(BatchScriptFile,['mkdir -p $WORK_DIR/',CFDCase,'/output/csv/CenterPlane',char(10)]);
295 fprintf(BatchScriptFile,['mkdir -p $WORK_DIR/',CFDCase,'/output/cas-dat',char(10)]);
296 fprintf(BatchScriptFile,['cd $WORK_DIR/',CFDCase,char(10)]);
297 fprintf(BatchScriptFile,['/usr/bin/rcp -r ${msas}:inbox/OutputDataSlices/',CFDCase,'/input
        ./',char(10)]);
298 fprintf(BatchScriptFile,['date >>&"./output/',CFDCase,'-OutputDataSlices-$PBS_JOBID.
        FluentStandardOutput"',char(10)]);
299 fprintf(BatchScriptFile,['fluent 3ddp -gu -driver null -i ./input/ExportSlices.jou>>&"./
        output/',CFDCase,'-OutputDataSlices-$PBS_JOBID.FluentStandardOutput"',char(10)]);
300 fprintf(BatchScriptFile,['cat "./output/',CFDCase,'-OutputDataSlices-$PBS_JOBID.
        FluentStandardOutput"| mail -s "complete ',CFDCase,'-OutputDataSlices- $PBS_JOBID"
        andy@schroder.net',char(10)]);
301 fclose(BatchScriptFile);
302
303
304 %generate journal file
305 journalfile=fopen([PathToWorkingDirectory,'/input/ExportSlices.jou'],'w');
306 %print out the date and time for reference
307 fprintf(journalfile,['!date',char(10)]);
308
309 if strcmp(FormerHawkJobNumber,'')
310         fprintf(journalfile,['!/usr/bin/rcp -r ${msas}:CompletedCFDCases/OutputData/',
            CFDCase,'/output/cas-dat/',ExportCaseFileName,' ./output/cas-dat/',char(10)]);
311         fprintf(journalfile,['/file/read-case ./output/cas-dat/',ExportCaseFileName,char(10)
            ]);
312 else
313         fprintf(journalfile,['/file/read-case /workspace/schrodau/',FormerHawkJobNumber,'/',
            CFDCase,'/output/cas-dat/',ExportCaseFileName,char(10)]);
314 end
315
316 fprintf(journalfile,['/surface/plane CenterPlane 0 0 0 1 0 0 -1 0 1',char(10)]);
317 if strcmp(Unsteady,'yes')
318         for x=StartingTimeStepNumber:EverXXxStepsSaved:EndingTimeStepNumber
319                 %print out the date and time for reference
320                 fprintf(journalfile,['!date',char(10)]);
321
322                 if strcmp(FormerHawkJobNumber,'')
323                         fprintf(journalfile,['!/usr/bin/rcp -r ${msas}:CompletedCFDCases/
                            OutputData/',CFDCase,'/output/cas-dat/',DataFileNamePrefix,'-1-'
                            ,num2str(x,'%05.0f'),'.dat ./output/cas-dat/',char(10)]);
324                         fprintf(journalfile,['/file/read-data ./output/cas-dat/',
                            DataFileNamePrefix,'-1-',num2str(x,'%05.0f'),'.dat',char(10)]);
325                 else
326                         fprintf(journalfile,['/file/read-data /workspace/schrodau/',
                            FormerHawkJobNumber,'/',CFDCase,'/output/cas-dat/',
                            DataFileNamePrefix,'-1-',num2str(x,'%05.0f'),'.dat',char(10)]);
327                 end
328                 %assumes it is second order unsteady, which it should be
329                 fprintf(journalfile,['/file/export/ascii ./output/csv/HeatedSurface/',
                        CFDCase,'-HeatedSurface-SecondOrderUnSteady-1-',num2str(x,'%05.0f'),'.
                        csv ',HeatedSurfaceZone1,' ',HeatedSurfaceZone2,' () no temperature
                        pressure wall-shear y-plus absolute-pressure () no yes',char(10)]);
330                 fprintf(journalfile,['/file/export/ascii ./output/csv/CenterPlane/',CFDCase,
                        '-CenterPlane-SecondOrderUnSteady-1-',num2str(x,'%05.0f'),'.csv
```

```
                    CenterPlane␣()␣no␣temperature␣pressure␣x-velocity␣y-velocity␣z-velocity␣
                    absolute-pressure␣()␣no␣yes',char(10)]);
331                 %copy slices to the archives
332                 %this may cause an issue because not in a tar file and a lot of small files
                    but try it for now.
333                 fprintf(journalfile,['!/usr/bin/rcp␣-r␣./output/csv/HeatedSurface/',CFDCase,
                    '-HeatedSurface-SecondOrderUnSteady-1-',num2str(x,'%05.0f'),'.csv␣${msas
                    }:CompletedCFDCases/OutputDataSlices/',CFDCase,'/output/csv/
                    HeatedSurface/',char(10)]);
334                 fprintf(journalfile,['!/usr/bin/rcp␣-r␣./output/csv/CenterPlane/',CFDCase,'-
                    CenterPlane-SecondOrderUnSteady-1-',num2str(x,'%05.0f'),'.csv␣${msas}:
                    CompletedCFDCases/OutputDataSlices/',CFDCase,'/output/csv/CenterPlane/',
                    char(10)]);
335         end
336 else
337         %print out the date and time for reference
338         fprintf(journalfile,['!date',char(10)]);
339         fprintf(journalfile,['!/usr/bin/rcp␣-r␣${msas}:CompletedCFDCases/OutputData/',
                CFDCase,'/output/cas-dat/',SliceDataFileName,'.dat␣./output/cas-dat/',char(10)])
                ;
340         fprintf(journalfile,['/file/read-data␣./output/cas-dat/',SliceDataFileName,'.dat',
                char(10)]);
341         fprintf(journalfile,['/file/export/ascii␣./output/csv/HeatedSurface/',
                SliceDataFileName,'-HeatedSurface.csv␣',HeatedSurfaceZone1,'␣',
                HeatedSurfaceZone2,'␣()␣no␣temperature␣pressure␣wall-shear␣y-plus␣absolute-
                pressure␣()␣no␣yes',char(10)]);
342         fprintf(journalfile,['/file/export/ascii␣./output/csv/CenterPlane/',
                SliceDataFileName,'-CenterPlane.csv␣CenterPlane␣()␣no␣temperature␣pressure␣x-
                velocity␣y-velocity␣z-velocity␣absolute-pressure␣()␣no␣yes',char(10)]);
343         %copy slices to the archives
344         %this may cause an issue because not in a tar file and a lot of small files but try
                it for now.
345         fprintf(journalfile,['!/usr/bin/rcp␣-r␣./output/csv/HeatedSurface/',
                SliceDataFileName,'-HeatedSurface.csv␣${msas}:CompletedCFDCases/OutputDataSlices
                /',CFDCase,'/output/csv/HeatedSurface/',char(10)]);
346         fprintf(journalfile,['!/usr/bin/rcp␣-r␣./output/csv/CenterPlane/',SliceDataFileName,
                '-CenterPlane.csv␣${msas}:CompletedCFDCases/OutputDataSlices/',CFDCase,'/output/
                csv/CenterPlane/',char(10)]);
347 end
348
349
350
351 fprintf(journalfile,['!date',char(10)]);
352 fprintf(journalfile,['exit',char(10)]);
353 fclose(journalfile);
```

A.4.2 GenerateCFDSliceValueAnimation.m

```
1  function GenerateCFDSliceValueAnimation(CFDCase,UnsteadyTimeStep,TimeStepNumbers,SliceValue,
       RegularXGrid,RegularZGrid,D,TargetRe,MinSliceValue,MaxSliceValue,SliceValueStep,
       SliceValueText,SliceFileName,NumberofHolesinMesh,GreyYes)
2
3       %although not reading data, writing data, so need to define where the output files
            go.
4       CFDCases='CompletedCFDCases';
5       BasePath='../../';
6       PathToWorkingDirectory=[BasePath,CFDCases,'/Post/',CFDCase,'/post/'];
7
8       mkdir([PathToWorkingDirectory,'/images/UnSteadySlice',SliceFileName,'/'])
                        %make the directory if it doesn't already exist
9       mkdir([PathToWorkingDirectory,'/videos/'])                    %make the directory
            if it doesn't already exist
10      delete([PathToWorkingDirectory,'/images/UnSteadySlice',SliceFileName,'/*.*'])
                    %delete any files that were in the directory, if it already existed
11      delete([PathToWorkingDirectory,'/videos/',CFDCase,'-Slice',SliceFileName,'-',num2str
            (min(TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.mp4'])
12      delete([PathToWorkingDirectory,'/videos/',CFDCase,'-Slice',SliceFileName,'-',num2str
            (min(TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.mpg'])
```

```
13
14
15          for TimeStep=1:size(TimeStepNumbers,2)
16
17                          [ContourFigure,ColorbarHandle]=PlotCFDSliceValues(squeeze(SliceValue
                                (TimeStep,:,:)),RegularXGrid,RegularZGrid,D,TargetRe,
                                MinSliceValue,MaxSliceValue,SliceValueStep,SliceValueText,
                                NumberofHolesinMesh,GreyYes,(TimeStep-1)*str2num(
                                UnsteadyTimeStep));
18
19                          CurrentImage = getframe(ContourFigure);
20                          %write out file. note that the file has to start with 1 so this
                                numbering scheme isn't the same as the time step number
21                          %note that this doesn't work if the screen is locked. it just
                                outputs a black image
22                          imwrite(CurrentImage.cdata, [PathToWorkingDirectory,'/images/
                                UnSteadySlice',SliceFileName,'/',CFDCase,'-Slice',SliceFileName,
                                '-',num2str(TimeStep,'%05.0f'),'.png']);    %this command doesn't
                                 seem to work right when specifying the output resolution
23
24                          %export as an eps image. for some reason it works okay with this
                                contour plot... not sure why I was having problems before.
25                          %hgexport(ContourFigure,['./images/',CaseNumberSelected,'/',
                                CaseNumberSelected,'-',num2str(TimeStep),'.eps']);          %
                                this command doesn't seem to work right when specifying any
                                other format than eps? also don't remember why it worked better
                                than the print command?
26
27                          close(ContourFigure)
28          end
29
30          %first clear out an environmental variable matlab sets that messes this program up
31          LD_LIBRARY_PATH_original=getenv('LD_LIBRARY_PATH');
32          setenv('LD_LIBRARY_PATH','');
33
34          eval(['!ffmpeg -r ',num2str(1/str2num(UnsteadyTimeStep),3),' -sameq -i ',
                PathToWorkingDirectory,'/images/UnSteadySlice',SliceFileName,'/',CFDCase,'-Slice
                ',SliceFileName,'-%05d.png',' ',PathToWorkingDirectory,'/videos/',CFDCase,'-
                Slice',SliceFileName,'-',num2str(min(TimeStepNumbers)),'-',num2str(max(
                TimeStepNumbers)),'.mp4'])
35          eval(['!ffmpeg -r ',num2str(1/str2num(UnsteadyTimeStep),3),' -sameq -i ',
                PathToWorkingDirectory,'/images/UnSteadySlice',SliceFileName,'/',CFDCase,'-Slice
                ',SliceFileName,'-%05d.png',' ',PathToWorkingDirectory,'/videos/',CFDCase,'-
                Slice',SliceFileName,'-',num2str(min(TimeStepNumbers)),'-',num2str(max(
                TimeStepNumbers)),'.mpg'])
36
37          %now change the environmental variable back
38          setenv('LD_LIBRARY_PATH',LD_LIBRARY_PATH_original)
39  end
```

A.4.3 GenerateSurfaceNuAnimation.m

```
1   function GenerateSurfaceNuAnimation(CFDCase,UnsteadyTimeStep,TimeStepNumbers,Nu,RegularXGrid
        ,RegularYGrid,D,TargetRe,MinNu,MaxNu,NuStep,HoverD,NumberofHolesinMesh)
2
3           %although not reading data, writing data, so need to define where the output files
                go.
4           CFDCases='CompletedCFDCases';
5           BasePath='../../';
6           PathToWorkingDirectory=[BasePath,CFDCases,'/Post/',CFDCase,'/post/'];
7
8           mkdir([PathToWorkingDirectory,'/images/UnSteadySurfaceNu/'])                    %
                make the directory if it doesn't already exist
9           mkdir([PathToWorkingDirectory,'/videos/'])                          %make the directory
                if it doesn't already exist
10          delete([PathToWorkingDirectory,'/images/UnSteadySurfaceNu/*.*'])                %
                delete any files that were in the directory, if it already existed
11          delete([PathToWorkingDirectory,'/videos/',CFDCase,'-SurfaceNu-',num2str(min(
```

```
        TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.mp4'])
12      delete([PathToWorkingDirectory,'/videos/',CFDCase,'-SurfaceNu-',num2str(min(
            TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.mpg'])
13
14      for TimeStep=1:size(TimeStepNumbers,2)
15                  [ContourFigure,ColorbarHandle]=PlotNuSurfaceCFD(squeeze(Nu(TimeStep
                        ,:,:)),RegularXGrid,RegularYGrid,D,TargetRe,MinNu,MaxNu,NuStep,
                        HoverD,NumberofHolesinMesh,'Nu',(TimeStep-1)*str2num(
                        UnsteadyTimeStep));
16
17                  CurrentImage = getframe(ContourFigure);
18                  %write out file. note that the file has to start with 1 so this
                        numbering scheme isn't the same as the time step number
19                  %note that this doesn't work if the screen is locked. it just
                        outputs a black image
20                  imwrite(CurrentImage.cdata, [PathToWorkingDirectory,'/images/
                        UnSteadySurfaceNu/',CFDCase,'-SurfaceNu-',num2str(TimeStep,'
                        %05.0f'),'.png']);      %this command doesn't seem to work right
                        when specifying the output resolution
21
22                  %export as an eps image. for some reason it works okay with this
                        contour plot... not sure why I was having problems before.
23                  %hgexport(ContourFigure,['./images/',CaseNumberSelected,'/',
                        CaseNumberSelected,'-',num2str(TimeStep),'.eps']);          %
                        this command doesn't seem to work right when specifying any
                        other format than eps? also don't remember why it worked better
                        than the print command?
24
25                  close(ContourFigure)
26      end
27
28      %first clear out an environmental variable matlab sets that messes this program up
29      LD_LIBRARY_PATH_original=getenv('LD_LIBRARY_PATH');
30      setenv('LD_LIBRARY_PATH','');
31
32      eval(['!ffmpeg -r ',num2str(1/str2num(UnsteadyTimeStep),3),' -sameq -i ',
            PathToWorkingDirectory,'/images/UnSteadySurfaceNu/',CFDCase,'-SurfaceNu-%05d.png
            ',' ',PathToWorkingDirectory,'/videos/',CFDCase,'-SurfaceNu-',num2str(min(
            TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.mp4'])
33      eval(['!ffmpeg -r ',num2str(1/str2num(UnsteadyTimeStep),3),' -sameq -i ',
            PathToWorkingDirectory,'/images/UnSteadySurfaceNu/',CFDCase,'-SurfaceNu-%05d.png
            ',' ',PathToWorkingDirectory,'/videos/',CFDCase,'-SurfaceNu-',num2str(min(
            TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.mpg'])
34
35      %now change the environmental variable back
36      setenv('LD_LIBRARY_PATH',LD_LIBRARY_PATH_original)
37  end
```

A.4.4 GetRegularCFDSliceValues.m

```
1   function [Regularvx,Regularvy,Regularvz,RegularTemperature,RegularPressure,RegularXGrid,
        RegularZGrid,D,TargetRe,UnsteadyTimeStep,RegularAbsolutePressure,NumberofHolesinMesh]=
        GetRegularCFDSliceValues(CFDCase,DataFileSuffix)
2
3       %setup some paths for reading the input data
4       CFDCases='CompletedCFDCases';
5       BasePath='../../';
6       CaseListDataFile='CaseList.txt';
7       ExperimentalData='/Experimental/ExperimentalDataCopy/impingement/';
8       PathToWorkingDirectory=[BasePath,CFDCases,'/OutputDataSlices/',CFDCase,'/'];
9       RawExperimentalDataPath=[BasePath,ExperimentalData,'/RawData/'];
10      PathToCaseListDataFile=[BasePath,ExperimentalData,'/analysis/',CaseListDataFile];
11
12      %include some code that reads in data from spreadsheets
13      ReadCFDCaseListSpreadsheet      %note: can't put a .m on the end of the sript name
            because it thinks the . is an operator or something
14      ReadExperimentalCaseListSpreadsheet
15      ReadLabViewData         %include some code that reads labview data and calculates a
```

```
        few derived quantities
16
17
18          %read in data
19          Slice=importdata([PathToWorkingDirectory,'output/csv/CenterPlane/',CFDCase,
                DataFileSuffix]);
20
21          X=Slice.data(:,2);
22          Z=-Slice.data(:,4);
23
24          %need to round because all mesh points aren't exactly aligned.
25          X=round(X*10^8)*10^(-8);
26          Z=round(Z*10^8)*10^(-8);
27
28          XMin=min(X);
29          XMax=max(X);
30
31          ZMin=min(Z);
32          ZMax=max(Z);
33
34
35          %lets you do a uniform mesh if you want to for some reason
36          % XRange=XMax-XMin;
37          % XResolution=XRange/100;
38          % RegularX=XMin:XResolution:XMax;
39          % ZRange=ZMax-ZMin;
40          % ZResolution=ZRange/100;
41          % RegularZ=ZMin:ZResolution:ZMax;
42
43
44          RegularX=sort(X(find(Z==ZMin)));          %need to sort because the order they are in
                the data file isn't basically random.
45
46          Center=median(RegularX);          %assumes the is at the median grid point
47          RegularZ=sort(Z(find(X==Center)));
48
49          %make the mesh and then reshape it.
50          [RegularXGrid,RegularZGrid]=meshgrid(RegularX,RegularZ);
51          RegularXGridReshaped=reshape(RegularXGrid,1,[]);
52          RegularZGridReshaped=reshape(RegularZGrid,1,[]);
53
54
55          %determine whether points are inside of the mesh or not
56          %call code that defines the perimeter of the mesh
57          if NumberofHolesinMesh==1
58                  InsideTheLowerGridSingleHole
59                  %since there is no pressure chamber for this case, just set the values to a
                        scalar zero
60                  INPressureChamber=0;
61                  ONPressureChamber=0;
62          elseif NumberofHolesinMesh==11
63                  InsideTheLowerGrid11Hole
64                  InsideThePressureChamberGrid11Hole
65                  %see if in the pressure chamber
66                  [INPressureChamber,ONPressureChamber]=inpolygon(RegularXGridReshaped,
                        RegularZGridReshaped,InsideThePressureChamberGrid(:,1),
                        InsideThePressureChamberGrid(:,2));
67          end
68
69          %see if in the lower grid
70          [INLower,ONLower]=inpolygon(RegularXGridReshaped,RegularZGridReshaped,
                InsideTheLowerGrid(:,1),InsideTheLowerGrid(:,2));
71
72          %now add both together
73          INorON=INLower+ONLower+INPressureChamber+ONPressureChamber;
74
75
76          %now define all values and convert points outside of the mesh to NaN
```

```
77
78          RegularAbsolutePressure=ValuesInGrid(TriScatteredInterp(X,Z,Slice.data(:,5)),
                RegularXGrid,RegularZGrid,INorON);
79          Regularvz=ValuesInGrid(TriScatteredInterp(X,Z,-Slice.data(:,6)),RegularXGrid,
                RegularZGrid,INorON);
80          Regularvy=ValuesInGrid(TriScatteredInterp(X,Z,Slice.data(:,7)),RegularXGrid,
                RegularZGrid,INorON);
81          Regularvx=ValuesInGrid(TriScatteredInterp(X,Z,Slice.data(:,8)),RegularXGrid,
                RegularZGrid,INorON);
82          RegularPressure=ValuesInGrid(TriScatteredInterp(X,Z,Slice.data(:,9)),RegularXGrid,
                RegularZGrid,INorON);
83          RegularTemperature=ValuesInGrid(TriScatteredInterp(X,Z,Slice.data(:,10)),
                RegularXGrid,RegularZGrid,INorON);
84
85  end
```

A.4.5 GetRegularCFDSurfaceNu.m

```
1   function [RegularNu,RegularXGrid,RegularYGrid,D,TargetRe,ExperimentalCaseNumber,
        RawExperimentalDataPath,CaseListDataFile,UnsteadyTimeStep,HoverD,Regularyplus,
        NumberofHolesinMesh]=GetRegularCFDSurfaceNu(CFDCase,RegularlySpacedPointsX,
        RegularlySpacedPointsY,DataFileSuffix)
2
3           %setup some paths for reading the input data
4           CFDCases='CompletedCFDCases';
5           BasePath='../../';
6           CaseListDataFile='CaseList.txt';
7           ExperimentalData='/Experimental/ExperimentalDataCopy/impingement/';
8           PathToWorkingDirectory=[BasePath,CFDCases,'/OutputDataSlices/',CFDCase,'/'];
9           RawExperimentalDataPath=[BasePath,ExperimentalData,'/RawData/'];
10          PathToCaseListDataFile=[BasePath,ExperimentalData,'/analysis/',CaseListDataFile];
11
12          %include some code that reads in data from spreadsheets
13          ReadCFDCaseListSpreadsheet      %note: can't put a .m on the end of the sript name
                because it thinks the . is an operator or something
14          ReadExperimentalCaseListSpreadsheet
15          ReadLabViewData          %include some code that reads labview data and calculates a
                few derived quantities
16
17          SurfaceTemperatureData=importdata([PathToWorkingDirectory,'output/csv/HeatedSurface/
                ',CFDCase,DataFileSuffix]);
18
19          SurfaceTemperature=SurfaceTemperatureData.data(:,9);
20          X=SurfaceTemperatureData.data(:,2);
21          Y=SurfaceTemperatureData.data(:,3);
22          yplus=SurfaceTemperatureData.data(:,6);
23
24          Tfilm=(SurfaceTemperature+UpstreamAverageTemperature)/2;          %film temperature on
                the forced convection side
25          h=(FoilHeatFlux)./(SurfaceTemperature-UpstreamAverageTemperature);
26
27          %define thermal conductivity of air as a function of temperature at atmospheric
28          %pressure (Incropera and DeWitt 5th edition table A.5
29          %K, W/(m*K)
30          k_T=[
31                  100,.00934
32                  150,.0138
33                  200,.0181
34                  250,.0223
35                  300,.0263
36                  350,.0300
37                  ];
38          %calculate thermal conductivity based upon the film temperature
39          %do a manual, linear interpolation between 350K and 300K
40          k=((k_T(6,2)-k_T(5,2))/(k_T(6,1)-k_T(5,1)))*(Tfilm-k_T(6,1))+k_T(6,2);
41
42          Nu=h*D./k;
43
```

```
        if size(RegularlySpacedPointsX,2)==1
                XMin=min(X);
                XMax=max(X);

                XRange=XMax-XMin;
                XResolution=XRange/RegularlySpacedPointsX;
                RegularX=XMin+XResolution:XResolution:XMax-XResolution;
        else
                RegularX=RegularlySpacedPointsX;
        end

        if size(RegularlySpacedPointsY,2)==1
                YMin=min(Y);
                YMax=max(Y);

                YRange=YMax-YMin;
                YResolution=YRange/RegularlySpacedPointsY;
                RegularY=YMin+YResolution:YResolution:YMax-YResolution;
        else
                RegularY=RegularlySpacedPointsY;
        end


        RegularNu=TriScatteredInterp(X,Y,Nu);
        Regularyplus=TriScatteredInterp(X,Y,yplus);
        [RegularXGrid,RegularYGrid]=meshgrid(RegularX,RegularY);
end
```

A.4.6 GetUnsteadyRegularCFDSliceValues.m

```
function [Regularvx,Regularvy,Regularvz,RegularTemperature,RegularPressure,RegularXGrid,RegularZGrid,D,TargetRe,UnsteadyTimeStep,RegularAbsolutePressure,NumberofHolesinMesh]=GetUnsteadyRegularCFDSliceValues(CFDCase1,CFDCase1TimeSteps,CFDCase2,CFDCase2TimeSteps,CFDCase3,CFDCase3TimeSteps)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
        %process the CFD data and make it a regularly spaced grid. count down in for loops so everything goes faster because memory is already preallocated.
        for x=size(CFDCase1TimeSteps,2):-1:1
                disp(['Current Time Step Number = ',num2str(CFDCase1TimeSteps(x))]);
                [Regularvx(x,:,:),Regularvy(x,:,:),Regularvz(x,:,:),RegularTemperature(x,:,:),RegularPressure(x,:,:),RegularXGrid,RegularZGrid,D,TargetRe,UnsteadyTimeStep,RegularAbsolutePressure(x,:,:),NumberofHolesinMesh]=GetRegularCFDSliceValues(CFDCase1,['-CenterPlane-SecondOrderUnSteady-1-',num2str(CFDCase1TimeSteps(x),'%05.0f'),'.csv']);
        end
        if ~strcmp(CFDCase2,'')
                for x=size(CFDCase2TimeSteps,2):-1:1
                        disp(['Current Time Step Number = ',num2str(CFDCase2TimeSteps(x))]);
                        [Regularvx(x+size(CFDCase1TimeSteps,2),:,:),Regularvy(x+size(CFDCase1TimeSteps,2),:,:),Regularvz(x+size(CFDCase1TimeSteps,2),:,:),RegularTemperature(x+size(CFDCase1TimeSteps,2),:,:),RegularPressure(x+size(CFDCase1TimeSteps,2),:,:),RegularXGrid,RegularZGrid,D,TargetRe,UnsteadyTimeStep,RegularAbsolutePressure(x+size(CFDCase1TimeSteps,2),:,:),NumberofHolesinMesh]=GetRegularCFDSliceValues(CFDCase2,['-CenterPlane-SecondOrderUnSteady-1-',num2str(CFDCase2TimeSteps(x),'%05.0f'),'.csv']);
                end
        end
end
```

A.4.7 PlotConvergence.m

```
clc;
clear global;
clear all;
close all;

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%hard coded values
CFDCase='CFD0036'
LastIterationsToPlot=1500                       %if steady

UnSteady='no';                                  %don't feel like reading in the case spreadsheet so just define this manually
FirstIterationToPlot=126*.022                   %if unsteady
LastIterationToPlot=226*.022                    %if unsteady

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
addpath /home/andy/Desktop/CFD/Experimental/scripts/
CFDCases='CompletedCFDCases/SmallFiles';
BasePath='../../';
PathToWorkingDirectory=[BasePath,CFDCases,'/',CFDCase,'/'];

if strcmp(UnSteady,'yes')
        Vertex_Max_Temperature_file='VertexMaximumSurfaceTemperatureTimeStep.csv';
        AverageSurfaceTemperature_file='AverageSurfaceTemperatureTimeStep.csv';
else
        Vertex_Max_Temperature_file='VertexMaximumSurfaceTemperature.csv';
        AverageSurfaceTemperature_file='AverageSurfaceTemperature.csv';
end


Vertex_Max_Temperature=importdata([PathToWorkingDirectory,'output/csv/',Vertex_Max_Temperature_file]);
AverageSurfaceTemperature=importdata([PathToWorkingDirectory,'output/csv/',AverageSurfaceTemperature_file]);
numberofdatapoints=size(Vertex_Max_Temperature.data,1);

if strcmp(UnSteady,'yes')
        FirstDataPointPlotted=round((FirstIterationToPlot/(Vertex_Max_Temperature.data(2,1)-Vertex_Max_Temperature.data(1,1))));;
        LastDataPointPlotted=round((LastIterationToPlot/(Vertex_Max_Temperature.data(2,1)-Vertex_Max_Temperature.data(1,1))));
else
        FirstDataPointPlotted=numberofdatapoints-(LastIterationsToPlot/(Vertex_Max_Temperature.data(2,1)-Vertex_Max_Temperature.data(1,1)));
        LastDataPointPlotted=numberofdatapoints;
end



Vertex_Max_Temperature_Variation=range(Vertex_Max_Temperature.data(FirstDataPointPlotted:LastDataPointPlotted,2))
AverageSurfaceTemperature_Variation=range(AverageSurfaceTemperature.data(FirstDataPointPlotted:LastDataPointPlotted,2))



Vertex_Max_Temperature_handle=plot(Vertex_Max_Temperature.data(:,1),Vertex_Max_Temperature.data(:,2));
xlim([Vertex_Max_Temperature.data(FirstDataPointPlotted,1) Vertex_Max_Temperature.data(LastDataPointPlotted,1)])
Xlabels=get(gca,'XTick');
for x=1:size(Xlabels,2)
        XlabelsFormatted{x}=thousands(Xlabels(x));
end
set(gca,'XTickLabel',XlabelsFormatted)
xlabel('Iteration')
ylabel('Vertex Maximum Temperature [K]')

pause
delete(Vertex_Max_Temperature_handle)
```

```
64 AverageSurfaceTemperature_handle=plot(AverageSurfaceTemperature.data(:,1),
       AverageSurfaceTemperature.data(:,2));
65 xlim([Vertex_Max_Temperature.data(FirstDataPointPlotted,1) Vertex_Max_Temperature.data(
       LastDataPointPlotted,1)])
66 set(gca,'XTickLabel',XlabelsFormatted)
67 xlabel('Iteration')
68 ylabel('Average Surface Temperature [K]')
```

A.4.8 PlotCFDSliceValues.m

```
1  function [ContourFigure,ColorbarHandle]=PlotCFDSliceValues(SliceValue,RegularXGrid,
       RegularZGrid,D,TargetRe,MinSliceValue,MaxSliceValue,SliceValueStep,SliceValueText,
       NumberofHolesinMesh,GreyYesNo,Time)
2
3          MinXD=round(min(min(RegularXGrid))/D*10)/10;     %need to round because round off
               error makes the grid points not exactly and also GetRegularCFDSurfaceNu throws
               out the end points
4          MaxXD=round(max(max(RegularXGrid))/D*10)/10;
5          MinZD=round(min(min(RegularZGrid))/D*10)/10;
6          MaxZD=round(max(max(RegularZGrid))/D*10)/10;
7
8          %make the aspect ratio of the figure match that of the graph so minimal whitespace
               exists in the exported images, but still not too large for the screen
9          MaxFigureHeight=900;
10         MaxFigureWidth=1700;
11
12         FigureHeight=MaxFigureHeight;
13         FigureWidth=floor(FigureHeight*(max(max(RegularXGrid))-min(min(RegularXGrid)))/(max(
               max(RegularZGrid))-min(min(RegularZGrid))));
14         if FigureWidth>MaxFigureWidth
15                 FigureWidth=MaxFigureWidth;
16                 FigureHeight=floor(FigureWidth*(max(max(RegularZGrid))-min(min(RegularZGrid)
                       ))/(max(max(RegularXGrid))-min(min(RegularXGrid))));
17         end
18
19         XZero=60;
20         ZZero=XZero;
21         ContourFigure=figure('OuterPosition',[XZero ZZero FigureWidth+XZero+120 FigureHeight
               +ZZero],'Name',['Re=',num2str(TargetRe)]);  %big figure
22
23         %FontSize=40;
24         FontSize=25;
25         LineWidth=3;
26
27         %saturate the SliceValue, mainly important for vorticity since there are local
               points with extreme vorticity, which require a huge number of levels to get a
               good contour plot
28         %note this will cause the data ticker to give the wrong value for saturated points.
29         SliceValue(find(SliceValue>MaxSliceValue))=MaxSliceValue;
30         SliceValue(find(SliceValue<MinSliceValue))=MinSliceValue;
31
32         contourf(RegularXGrid/D,RegularZGrid/D,SliceValue,100,'LineStyle','none');
33         ColorbarHandle=colorbar;
34         set(get(ColorbarHandle,'ylabel'),'String', SliceValueText, 'Rotation', 90, '
               VerticalAlignment', 'Bottom','FontSize',FontSize)
35
36         hold all
37         if ~exist('GreyYesNo','var')||strcmp(GreyYesNo,'yes')
38                 colormap('gray')
39         end
40
41         caxis([MinSliceValue, MaxSliceValue])
42         set(gca,'FontSize',FontSize);
43         set(get(gca,'XLabel'),'FontSize',FontSize);
44         set(get(gca,'YLabel'),'FontSize',FontSize);
45         set(gca,'LineWidth',LineWidth)
46         set(ColorbarHandle,'LineWidth',LineWidth);
47         set(ColorbarHandle,'ytick', [MinSliceValue:SliceValueStep:MaxSliceValue]);
```

```
        %adjust the location of the colorbar
        LocateColorbarLabel = get(ColorbarHandle,'ylabel');
        pos = get(LocateColorbarLabel,'position');
        pos(1,1) = pos(1,1)+10;
        set(LocateColorbarLabel,'Position',pos);

        %plot(RegularXGridReshaped(find(INorON~=0))/D,RegularZGridReshaped(find(INorON~=0))/D,'Marker','.','LineStyle','none','MarkerEdgeColor','r')
        %pause
        %plot(X,Z,'Marker','.','LineStyle','none','MarkerEdgeColor','black')
        %plot(InsideTheGrid(:,1)/D,InsideTheGrid(:,2)/D)

        xlabel('x/D');
        ylabel('z/D');
        daspect([1 1 1])        %constrain the axes

        if exist('Time','var')
                title(['Re=',thousands(TargetRe),' - t=',num2str(Time,'%07.4f'),'s'])
        else
                title(['Re=',thousands(TargetRe)])
        end

        %xlim([MinXD MaxXD])
        %xlim([0 .05]/D)
        %ylim([-MeasurementDomainHeight/2 MeasurementDomainHeight/2]/D)
        %set(get(ContourFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick','on','XTick',[-15:3:15],'YTick',[-6:3:6])

        set(gcf,'Color','w')

        if NumberofHolesinMesh==1
                set(get(ContourFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick','on','XTick',[-30:.5:30],'YTick',[-5,-4,-3,-2,-1,0])
                %make the figure a little wider since the label is cut off
                CurrentDimensions=get(ContourFigure,'OuterPosition');
                CurrentDimensions(3)=CurrentDimensions(3)+150;
                set(ContourFigure,'OuterPosition',CurrentDimensions);
        elseif NumberofHolesinMesh==11
                set(get(ContourFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick','on','XTick',[-30:3:30],'YTick',[-5,-2,0,3,6,9,12,15])
        end
end
```

A.4.9 PlotNuCFDSingleHoleGridDependencyCases.m

```
clc;
clear global;
clear all;
close all;

%add path for some other scripts used
addpath /home/andy/Desktop/CFD/CFDScripts/      %this is used because the run command changes the path and messes all other paths defined up
addpath /home/andy/Desktop/CFD/Experimental/ExperimentalDataCopy/impingement/analysis/
addpath /home/andy/Desktop/CFD/Experimental/scripts/

%need to setup path so know where to save files to
CFDCases='CompletedCFDCases';
BasePath='../../';
PathToWorkingDirectory=[BasePath,CFDCases,'/Post/GridDependencyStudy/'];
mkdir([PathToWorkingDirectory])


NumberofRegularlySpacedPointsX=100;
NumberofRegularlySpacedPointsY=100;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%hard coded values, in order of increasing grid points
%Re=4,000 cases
CaseNumbersSelected={
        %'casenumber','CFD marker', 'Experimental Marker','CFD Label Text','experimental yes or nothing','Experimental Label Text',GridPoints,'NuFilenamesuffix','SliceFilenamesuffix'
        'CFD0036','+','o','Single Jet CFD -   1.2 million points - (1/1.3^{-1})x','no','Center Jet Experimental',1.2,'-SecondOrderSteadyShouldBeConverged-HeatedSurface.csv','-SecondOrderSteadyShouldBeConverged-CenterPlane.csv'
        'CFD0053','x','o','Single Jet CFD -   2.4 million points - (1/1.3^0)x (baseline)','no','Center Jet Experimental',2.4,'-SecondOrderSteadyShouldBeConverged-HeatedSurface.csv','-SecondOrderSteadyShouldBeConverged-CenterPlane.csv'
        'CFD0054','.','o','Single Jet CFD -   5.6 million points - (1/1.3^1)x','no','Center Jet Experimental',5.6,'-SecondOrderSteadyShouldBeConverged-HeatedSurface.csv','-SecondOrderSteadyShouldBeConverged-CenterPlane.csv'
        'CFD0055','s','o','Single Jet CFD - 12.6 million points - (1/1.3^2)x','no','Center Jet Experimental',12.6,'-SecondOrderSteadyShouldBeConverged-HeatedSurface.csv','-SecondOrderSteadyShouldBeConverged-CenterPlane.csv'
        };
NuContourPlotMin=0;
NuContourPlotMax=75;
NuContourPlotStep=25;
VorticityContourPlotMin=-2500;
VorticityContourPlotMax=2500;
VorticityContourPlotStep=500;
VectorScaleFactor=1/100;





% % %Re=15,000 cases
% CaseNumbersSelected={
%         %'casenumber','CFD marker', 'Experimental Marker','CFD Label Text','experimental yes or nothing','Experimental Label Text',GridPoints,'NuFilenamesuffix','SliceFilenamesuffix'
%         'CFD0041','+','o','Single Jet CFD -   1.2 million points - (1/1.3^{-1})x','no','Center Jet Experimental',1.2,'-SecondOrderSteadyShouldBeConverged-HeatedSurface.csv','-SecondOrderSteadyShouldBeConverged-CenterPlane.csv'
%         'CFD0024','x','o','Single Jet CFD -   2.6 million points - (1/1.3^0)x (baseline)','no','Center Jet Experimental',2.6,'-SecondOrderSteady-1-25000-HeatedSurface.csv','-SecondOrderSteady-1-25000-CenterPlane.csv'
%         'CFD0056','.','o','Single Jet CFD -   5.9 million points - (1/1.3^1)x','no','Center Jet Experimental',5.9,'-SecondOrderSteadyShouldBeConverged-HeatedSurface.csv','-SecondOrderSteadyShouldBeConverged-CenterPlane.csv'
%         'CFD0057','s','o','Single Jet CFD - 13.2 million points - (1/1.3^2)x','no','Center Jet Experimental',13.2,'-SecondOrderSteadyShouldBeConverged-HeatedSurface.csv','-SecondOrderSteadyShouldBeConverged-CenterPlane.csv'
%         };
% NuContourPlotMin=0;
% NuContourPlotMax=150;
% NuContourPlotStep=25;
% VorticityContourPlotMin=-5000;
% VorticityContourPlotMax=5000;
% VorticityContourPlotStep=1000;
% VectorScaleFactor=1/200;
% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
% % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



for CurrentCase=1:size(CaseNumbersSelected,1)
        [RegularNu(CurrentCase,:,:),RegularXGrid,RegularYGrid,TargetRe,D,HoverD,Regularyplus(CurrentCase,:,:),NumberofHolesinMesh]=PlotNuSpanwiseAverageExperimentalAndCFD(CaseNumbersSelected{CurrentCase,1},CaseNumbersSelected{CurrentCase,2},CaseNumbersSelected{CurrentCase,3},NumberofRegularlySpacedPointsX,NumberofRegularlySpacedPointsY,CaseNumbersSelected{CurrentCase,8},
```

```
            CaseNumbersSelected{CurrentCase,4},CaseNumbersSelected{CurrentCase,5},
            CaseNumbersSelected{CurrentCase,6});
67          MeanNu(CurrentCase)=mean(mean(RegularNu(CurrentCase,:,:)));
68
69          %calculate percent differences.
70          if CurrentCase~=1
71                  PercentChange(CurrentCase,:,:)=(RegularNu(CurrentCase,:,:)-RegularNu(
                        CurrentCase-1,:,:))./(RegularNu(CurrentCase-1,:,:))*100;
72                  AbsPercentChangeofAverage(CurrentCase,:,:)=abs((MeanNu(CurrentCase)-MeanNu(
                        CurrentCase-1))./(MeanNu(CurrentCase-1))*100);
73                  MaxPercentChange(CurrentCase)=max(max(abs(PercentChange(CurrentCase,:,:))));
74                  AveragePercentChange(CurrentCase)=mean(mean(abs(MaxPercentChange(CurrentCase
                        ))));
75          end
76
77  end
78  %override a few plot parameters for the single hole case
79  title(['Span Averaged Nusselt Number']);
80  ylim([min(get(gca,'YTick')) (max(get(gca,'YTick'))+15)])          %add a little space so the
        legend isn't on top of things
81  hgexport(gcf,[PathToWorkingDirectory,'Re',num2str(TargetRe),'SpanwiseAverages.eps'])
82  %pause
83  close all
84
85
86  %plot percent differences
87  figure('OuterPosition',[30 30 500 400]);
88  plot([CaseNumbersSelected{2:size(CaseNumbersSelected,1),7}],MaxPercentChange(2:size(
        CaseNumbersSelected,1)),'-s')
89  title(['Maximum Absolute Value of Percent Change in Local Nusselt Number',char(10),'Single
        Jet - Re=',thousands(TargetRe),char(10),'Data Interpolated to a Uniform ',num2str(
        NumberofRegularlySpacedPointsX),'x',num2str(NumberofRegularlySpacedPointsY),' Grid'])
90  xlabel('Million Grid Points');
91  ylabel('% Change');
92  set(get(gcf,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick','on','
        XTick',[0:1:15])
93  hgexport(gcf,[PathToWorkingDirectory,'Re',num2str(TargetRe),'MaxAbsPercentChangeLocalNu.eps'
        ])
94  %pause
95  close all
96
97
98  %this plot isn't very useful if the others are presented
99  close all
100 figure('OuterPosition',[30 30 500 400]);
101 plot([CaseNumbersSelected{2:size(CaseNumbersSelected,1),7}],AveragePercentChange(2:size(
        CaseNumbersSelected,1)),'-s')
102 title(['Area Weighted Average Absolute Value of Percent Change in Local Nusselt Number',char
        (10),'Single Jet - Re=',thousands(TargetRe),char(10),'Data Interpolated to a Uniform ',
        num2str(NumberofRegularlySpacedPointsX),'x',num2str(NumberofRegularlySpacedPointsY),'
        Grid'])
103 xlabel('Million Grid Points');
104 ylabel('% Change');
105 set(get(gcf,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick','on','
        XTick',[0:1:15])
106 hgexport(gcf,[PathToWorkingDirectory,'Re',num2str(TargetRe),'AvgAbsPercentChangeLocalNu.eps'
        ])
107 %pause
108 close all
109
110
111 close all
112 figure('OuterPosition',[30 30 500 400]);
113 plot([CaseNumbersSelected{2:size(CaseNumbersSelected,1),7}],AbsPercentChangeofAverage(2:size
        (CaseNumbersSelected,1)),'-s')
114 title(['Absolute Value of Percent Change in Area Weighted Average Nusselt Number',char(10),'
        Single Jet - Re=',thousands(TargetRe),char(10),'Data Interpolated to a Uniform ',num2str
        (NumberofRegularlySpacedPointsX),'x',num2str(NumberofRegularlySpacedPointsY),' Grid'])
```

```
115 xlabel('Million Grid Points');
116 ylabel('% Change');
117 set(get(gcf,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick','on','
    XTick',[0:1:15])
118 hgexport(gcf,[PathToWorkingDirectory,'Re',num2str(TargetRe),'AbsPercentChangeAvgNu.eps'])
119 %pause
120 close all
121
122
123 close all
124 figure('OuterPosition',[30 30 500 400]);
125 plot([CaseNumbersSelected{:,7}],MeanNu,'-s')
126 title(['Area Weighted Average Nusselt Number',char(10),'Single Jet - Re=',thousands(TargetRe
    ),char(10),'Data Interpolated to a Uniform ',num2str(NumberofRegularlySpacedPointsX),'x'
    ,num2str(NumberofRegularlySpacedPointsY),' Grid'])
127 xlabel('Million Grid Points');
128 ylabel('Nu');
129 set(get(gcf,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick','on','
    XTick',[0:1:15])
130 hgexport(gcf,[PathToWorkingDirectory,'Re',num2str(TargetRe),'AvgNu.eps'])
131 %pause
132 close all
133
134
135
136 %do contour plot of y+ for the coarsest case
137 CFDCase=CaseNumbersSelected{1,1}
138 PathToWorkingDirectory=[BasePath,CFDCases,'/Post/',CFDCase,'/post/images/'];
139 mkdir([PathToWorkingDirectory])
140 [ContourFigure,ColorbarHandle]=PlotNuSurfaceCFD(squeeze(Regularyplus(1,:,:)),RegularXGrid,
    RegularYGrid,D,TargetRe,0,ceil(max(max(Regularyplus(1,:,:)))*10)/10,ceil(max(max(
    Regularyplus(1,:,:)))*10)/10/2,HoverD,NumberofHolesinMesh,'y+');
141 hgexport(gcf,[PathToWorkingDirectory,CFDCase,'-yplus.eps'])
142 %pause
143 close all
144
145
146
147
148 %Now do contour plots for the finest grid point case
149 CFDCase=CaseNumbersSelected{size(CaseNumbersSelected,1),1}
150 PathToWorkingDirectory=[BasePath,CFDCases,'/Post/',CFDCase,'/post/images/'];
151 mkdir([PathToWorkingDirectory])
152
153
154 %now do countour plot of Nu for the finest grid, assuming it is last in the case list
155 [ContourFigure,ColorbarHandle]=PlotNuSurfaceCFD(squeeze(RegularNu(size(CaseNumbersSelected
    ,1),:,:)),RegularXGrid,RegularYGrid,D,TargetRe,NuContourPlotMin,NuContourPlotMax,
    NuContourPlotStep,HoverD,NumberofHolesinMesh);
156 hgexport(gcf,[PathToWorkingDirectory,CFDCase,'-Nu.eps'])
157 %pause
158 close all
159
160 %now do countour plot of y+ for the finest grid, assuming it is last in the case list
161 [ContourFigure,ColorbarHandle]=PlotNuSurfaceCFD(squeeze(Regularyplus(size(
    CaseNumbersSelected,1),:,:)),RegularXGrid,RegularYGrid,D,TargetRe,0,ceil(max(max(
    Regularyplus(size(CaseNumbersSelected,1),:,:)))*10)/10,ceil(max(max(Regularyplus(size(
    CaseNumbersSelected,1),:,:)))*10)/10/2,HoverD,NumberofHolesinMesh,'y+');
162 hgexport(gcf,[PathToWorkingDirectory,CFDCase,'-yplus.eps'])
163 %pause
164 close all
165
166
167 %get the slice data
168 [Regularvx,Regularvy,Regularvz,RegularTemperature,RegularPressure,RegularXGrid,RegularZGrid,
    D,TargetRe,UnsteadyTimeStep,RegularAbsolutePressure,NumberofHolesinMesh]=
    GetRegularCFDSliceValues(CFDCase,CaseNumbersSelected{size(CaseNumbersSelected,1),9});
169
```

```
%plot the vorticity
[ContourFigure]=PlotCFDSliceValues(curl(RegularXGrid,RegularZGrid,Regularvx,Regularvz),
    RegularXGrid,RegularZGrid,D,TargetRe,VorticityContourPlotMin,VorticityContourPlotMax,
    VorticityContourPlotStep,'Normal Component of Vorticity [1/s]',NumberofHolesinMesh,'no')
    ;
hgexport(gcf,[PathToWorkingDirectory,CFDCase,'-Vorticity.eps'])
%pause
close all

%plot the temperature
[ContourFigure]=PlotCFDSliceValues(RegularTemperature,RegularXGrid,RegularZGrid,D,TargetRe,
    floor(min(min(RegularTemperature))/10)*10,ceil(max(max(RegularTemperature))/10)*10,10,'
    Static Temperature [K]',NumberofHolesinMesh,'no');
hgexport(gcf,[PathToWorkingDirectory,CFDCase,'-Temperature.eps'])
%pause
close all

%plot the velocity magnitude
[ContourFigure]=PlotCFDSliceValues((Regularvx.^2+Regularvz.^2).^.5,RegularXGrid,RegularZGrid
    ,D,TargetRe,0,max(max((Regularvx.^2+Regularvz.^2).^.5)),5,'Velocity Magnitude [m/s]',
    NumberofHolesinMesh);
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,min(min(RegularXGrid)),max(max(
    RegularXGrid)),50,min(min(RegularZGrid)),max(max(RegularZGrid)),50,D,VectorScaleFactor,
    NumberofHolesinMesh)
hgexport(gcf,[PathToWorkingDirectory,CFDCase,'-Velocity.eps'])
%pause

xlim([-1.75 -.75])
ylim([-5 -4])
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,-D,-D,50,min(min(RegularZGrid))
    ,-4.5*D,100,D,1/100,NumberofHolesinMesh)
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,-1.5*D,-1.5*D,50,min(min(
    RegularZGrid)),-4.5*D,100,D,VectorScaleFactor,NumberofHolesinMesh)
hgexport(gcf,[PathToWorkingDirectory,CFDCase,'-VelocityZoomed1.eps'])
%pause
close all

[ContourFigure]=PlotCFDSliceValues((Regularvx.^2+Regularvz.^2).^.5,RegularXGrid,RegularZGrid
    ,D,TargetRe,0,max(max((Regularvx.^2+Regularvz.^2).^.5)),5,'Velocity Magnitude [m/s]',
    NumberofHolesinMesh);
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,min(min(RegularXGrid)),max(max(
    RegularXGrid)),50,min(min(RegularZGrid)),max(max(RegularZGrid)),50,D,VectorScaleFactor,
    NumberofHolesinMesh)
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,-D,-D,50,min(min(RegularZGrid))
    ,-4.5*D,50,D,VectorScaleFactor,NumberofHolesinMesh)
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,-1.5*D,-1.5*D,50,min(min(
    RegularZGrid)),-4.5*D,50,D,VectorScaleFactor,NumberofHolesinMesh)
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,-2*D,-2*D,50,min(min(
    RegularZGrid)),-4.5*D,50,D,VectorScaleFactor,NumberofHolesinMesh)
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,-2.5*D,-2.5*D,50,min(min(
    RegularZGrid)),-4.5*D,50,D,VectorScaleFactor,NumberofHolesinMesh)

xlim([-3 0])
ylim([-5 -2])
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,-2.5*D,0,100,-4*D,-4*D,100,D,
    VectorScaleFactor,NumberofHolesinMesh)
VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,-2.5*D,0,100,-3*D,-3*D,100,D,
    VectorScaleFactor,NumberofHolesinMesh)
hgexport(gcf,[PathToWorkingDirectory,CFDCase,'-VelocityZoomed2.eps'])
%pause
close all
```

A.4.10 PlotNuMultipleCFDCases.m

```
clc;
clear global;
clear all;
close all;
```

```
5
6  %add path for some other scripts used
7  addpath /home/andy/Desktop/CFD/CFDScripts/       %this is used because the run command
       changes the path and messes all other paths defined up
8  addpath /home/andy/Desktop/CFD/Experimental/ExperimentalDataCopy/impingement/analysis/
9  addpath /home/andy/Desktop/CFD/Experimental/scripts/
10
11
12
13
14
15
16
17
18
19 %define the cases
20 %this is setup so it can be a copy and paste from PlotMultipleNuSpanwiseAverages.m, so some
       fields are not used
21 %keep all lines commented except the case to process
22 UnsteadyCaseNumbersSelected={
23         %'casenumber1',timestepnumbers1,'casenumber2',timestepnumbres2,'CFD marker','
               Experimental Marker','CFD label text','experimental yes or no', 'experimental
               label text',NumberofRegularlySpacedPointsX,NumberofRegularlySpacedPointsY
24 %       'CFD0048',227:260,'CFD0051',261:727,'+','o','11 Hole - y/D=4.03 - Numerical','no','N
       /A',650,100
25 %       'CFD0052',227:329,'CFD0059',330:727,'x','o','11 Hole - y/D=3 - Numerical','yes','
       Center Row Experimental',650,100
26 %       'CFD0046',227:727,'',0,'.','o','Single Hole Repeating Case I - Numerical','no','N/A
       ',65,100
27         'CFD0045',227:727,'',0,'d','o','Single Hole Repeating Case II - Numerical','no','
               Center Row Experimental',65,100
28         };
29
30 %now, assign the values from the selected case so the rest of the code doesn't need to be
       modified, and also so it stays more compact/understandable.
31 CFDCase1=UnsteadyCaseNumbersSelected{1};
32 TimeStepNumbers1=UnsteadyCaseNumbersSelected{2};
33 %note files are exported to CFDCase2's path, also, both cases should have had the same time
       step
34 CFDCase2=UnsteadyCaseNumbersSelected{3};
35 TimeStepNumbers2=UnsteadyCaseNumbersSelected{4};
36 NumberofRegularlySpacedPointsX=UnsteadyCaseNumbersSelected{10};
37 NumberofRegularlySpacedPointsY=UnsteadyCaseNumbersSelected{11};
38
39
40 %check to see if doing multiple cases
41 if ~strcmp(CFDCase2,'')
42         TimeStepNumbers=[TimeStepNumbers1,TimeStepNumbers2];     %add the 2 together
43         CFDCase=CFDCase2;
44 else
45         TimeStepNumbers=TimeStepNumbers1;
46         CFDCase=CFDCase1;
47 end
48
49 %need to setup path so know where to save files to
50 CFDCases='CompletedCFDCases';
51 BasePath='../../';
52 PathToWorkingDirectory=[BasePath,CFDCases,'/Post/',CFDCase,'/post/'];
53 mkdir([PathToWorkingDirectory,'/images/'])
54
55
56 if ~strcmp(CFDCase2,'')
57         TimeStepNumbers=[TimeStepNumbers1,TimeStepNumbers2];     %add the 2 together
58         CFDCase=CFDCase2;
59 else
60         TimeStepNumbers=TimeStepNumbers1;
61         CFDCase=CFDCase1;
62 end
```

```
63
64 [RegularNu,RegularXGrid,RegularYGrid,D,TargetRe,UnsteadyTimeStep,HoverD,Regularyplus,
       NumberofHolesinMesh]=PlotNuSpanwiseAverageExperimentalAndCFDUnsteady(CFDCase1,
       TimeStepNumbers1,CFDCase2,TimeStepNumbers2,'',0,'x','o',NumberofRegularlySpacedPointsX,
       NumberofRegularlySpacedPointsY,'11 Hole CFD','yes','Center Row Experimental');
65 %override a few plot parameters
66 title(['Span and Time Averaged Nusselt Number - Entire Numerical y/D']);
67 ylim([min(get(gca,'YTick')) (max(get(gca,'YTick'))+5)]) %add a little space so the legend
       isn't on top of things
68 hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'SpanwiseTimeAverageNu-',num2str(min
       (TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
69 %pause
70 close all
71
72
73
74
75
76
77
78 %now do countour plot of Nu
79 close all
80 [ContourFigure,ColorbarHandle]=PlotNuSurfaceCFD(squeeze(mean(RegularNu)),RegularXGrid,
       RegularYGrid,D,TargetRe,0,100,50,HoverD,NumberofHolesinMesh);
81 %override a few values to make the single hole come out better.
82 hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeAverageNu-',num2str(min(
       TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
83 %pause
84 close all
85
86
87 %now do countour plot of y+
88 close all
89 [ContourFigure,ColorbarHandle]=PlotNuSurfaceCFD(squeeze(max(Regularyplus)),RegularXGrid,
       RegularYGrid,D,TargetRe,0,ceil(max(max(max(Regularyplus)))*10)/10,ceil(max(max(max(
       Regularyplus)))*10)/10/2,HoverD,NumberofHolesinMesh,'y+');
90 hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeMaxyplus-',num2str(min(
       TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
91 %pause
92 close all
93
94
95 GenerateSurfaceNuAnimation(CFDCase,UnsteadyTimeStep,TimeStepNumbers,RegularNu,RegularXGrid,
       RegularYGrid,D,TargetRe,0,100,50,HoverD,NumberofHolesinMesh)
```

A.4.11 PlotMultipleCFDSliceValues.m

```
1 clc;
2 clear global;
3 clear all;
4 close all;
5
6 %add path for some other scripts used
7 addpath /home/andy/Desktop/CFD/CFDScripts/       %this is used because the run command
      changes the path and messes all other paths defined up
8 addpath /home/andy/Desktop/CFD/Experimental/ExperimentalDataCopy/impingement/analysis/
9 addpath /home/andy/Desktop/CFD/Experimental/scripts/
10
11
12
13
14
15
16
17
18
19 %define the cases
20 %this is setup so it can be a copy and paste from PlotMultipleNuSpanwiseAverages.m, so some
```

```
    fields are not used
%keep all lines commented except the case to process
UnsteadyCaseNumbersSelected={
        %'casenumber1',timestepnumbers1,'casenumber2',timestepnumbres2,'CFD marker','
            Experimental Marker','CFD label text','experimental yes or no', 'experimental
            label text',NumberofRegularlySpacedPointsX,NumberofRegularlySpacedPointsY
%       'CFD0048',227:260,'CFD0051',261:727,'+','o','11 Hole - y/D=4.03 - Numerical','no','N
    /A',650,100
%       'CFD0052',227:329,'CFD0059',330:727,'x','o','11 Hole - y/D=3 - Numerical','yes','
    Center Row Experimental',650,100
%       'CFD0046',227:727,'',0,'.','o','Single Hole Repeating Case I - Numerical','no','N/A
    ',65,100
        'CFD0045',227:727,'',0,'d','o','Single Hole Repeating Case II - Numerical','no','
            Center Row Experimental',65,100
        };

%now, assign the values from the selected case so the rest of the code doesn't need to be
    modified, and also so it stays more compact/understandable.
CFDCase1=UnsteadyCaseNumbersSelected{1};
TimeStepNumbers1=UnsteadyCaseNumbersSelected{2};
%note files are exported to CFDCase2's path, also, both cases should have had the same time
    step
CFDCase2=UnsteadyCaseNumbersSelected{3};
TimeStepNumbers2=UnsteadyCaseNumbersSelected{4};



%check to see if doing multiple cases
if ~strcmp(CFDCase2,'')
        TimeStepNumbers=[TimeStepNumbers1,TimeStepNumbers2];     %add the 2 together
        CFDCase=CFDCase2;
else
        TimeStepNumbers=TimeStepNumbers1;
        CFDCase=CFDCase1;
end

%need to setup path so know where to save files to
CFDCases='CompletedCFDCases';
BasePath='../../';
PathToWorkingDirectory=[BasePath,CFDCases,'/Post/',CFDCase,'/post/'];
mkdir([PathToWorkingDirectory,'/images/'])


%get the data
[Regularvx,Regularvy,Regularvz,RegularTemperature,RegularPressure,RegularXGrid,RegularZGrid,
    D,TargetRe,UnsteadyTimeStep,RegularAbsolutePressure,NumberofHolesinMesh]=
    GetUnsteadyRegularCFDSliceValues(CFDCase1,TimeStepNumbers1,CFDCase2,TimeStepNumbers2,''
    ,0);



%plot the static pressure
PlotCFDSliceValues(squeeze(mean(RegularPressure)),RegularXGrid,RegularZGrid,D,TargetRe,floor
    (min(min(min(RegularPressure)))),ceil(max(max(max(RegularPressure)))),50,'Static Gauge
    Pressure [Pa]',NumberofHolesinMesh,'no');
hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeAverageStaticPressure-',num2str
    (min(TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
%pause
close all


%compute the vorticity
for CurrentTimeStep=1:size(Regularvx,1)
        RegularVorticity(CurrentTimeStep,:,:)=curl(RegularXGrid,RegularZGrid,squeeze(
            Regularvx(CurrentTimeStep,:,:)),squeeze(Regularvz(CurrentTimeStep,:,:)));
end
%plot the vorticity
```

```
73  PlotCFDSliceValues(squeeze(mean(RegularVorticity)),RegularXGrid,RegularZGrid,D,TargetRe
        ,-2500,2500,500,'Normal Component of Vorticity [1/s]',NumberofHolesinMesh,'no');
74  hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeAverageVorticity-',num2str(min(
        TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
75  %pause
76  close all
77  %also generate an animation of vorticity
78  GenerateCFDSliceValueAnimation(CFDCase,UnsteadyTimeStep,TimeStepNumbers,RegularVorticity,
        RegularXGrid,RegularZGrid,D,TargetRe,-2500,2500,500,'Normal Component of Vorticity [1/s]
        ','Vorticity',NumberofHolesinMesh,'no')
79  close all
80
81
82  %plot the velocity magnitude
83  PlotCFDSliceValues(squeeze(mean((Regularvx.^2+Regularvz.^2).^.5)),RegularXGrid,RegularZGrid,
        D,TargetRe,0,20,5,'In Plane Velocity Magnitude [m/s]',NumberofHolesinMesh);
84  hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeAverageVelocityMagnitude-',
        num2str(min(TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
85  %pause
86  close all
87  %also generate an animation of velocity
88  GenerateCFDSliceValueAnimation(CFDCase,UnsteadyTimeStep,TimeStepNumbers,(Regularvx.^2+
        Regularvz.^2).^.5,RegularXGrid,RegularZGrid,D,TargetRe,0,20,5,'In Plane Velocity
        Magnitude [m/s]','VelocityMagnitude',NumberofHolesinMesh,'yes')
89  close all
90
91
92  %plot the temperature
93  [ContourFigure]=PlotCFDSliceValues(squeeze(mean(RegularTemperature)),RegularXGrid,
        RegularZGrid,D,TargetRe,floor(min(min(min(RegularTemperature)))/10)*10,ceil(max(max(max(
        RegularTemperature)))/10)*10,10,'Static Temperature [K]',NumberofHolesinMesh,'no');
94  hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeAverageTemperature-',num2str(
        min(TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
95  %now plot a close up
96  xlim([-.5 2.5])
97  ylim([-5 -4])
98  set(get(ContourFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick
        ','on','XTick',[-30:.5:30],'YTick',[-5:.25:20])
99  hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeAverageTemperatureZoomed-',
        num2str(min(TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
100 %pause
101 close all
102
103 %plot the density
104 %specific ideal gas constant for air
105 R=287;  %J/(kg*K), anderson, modern compressible flow, page 21
106 RegularDensity=(RegularAbsolutePressure)./(R*RegularTemperature);
107 PlotCFDSliceValues(squeeze(mean(RegularDensity)),RegularXGrid,RegularZGrid,D,TargetRe,floor
        (10*min(min(min(RegularDensity))))/10,ceil(10*max(max(max(RegularDensity))))/10,.1,'
        Static Density [kg/m^3]',NumberofHolesinMesh,'no');
108 hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeAverageDensity-',num2str(min(
        TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
109 (1-min(min(min(RegularDensity)))/max(max(max(RegularDensity))))*100
110 %now plot a close up
111 xlim([-.5 2.5])
112 ylim([-5 -4])
113 set(get(ContourFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick
        ','on','XTick',[-30:.5:30],'YTick',[-5:.25:20])
114 hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeAverageDensityZoomed-',num2str(
        min(TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
115 %pause
116 close all
117
118
119 %plot the total pressure
120 gamma=1.4;
121 RegularSpeedofSound=(gamma*R*RegularTemperature).^.5;
122 RegularMach=((Regularvx.^2+Regularvy.^2+Regularvz.^2).^.5)./RegularSpeedofSound;
```

```
123 RegularTotalPressure=(RegularAbsolutePressure).*(1+((gamma-1)/2)*RegularMach.^2).^(gamma/(
    gamma-1));
124 PlotCFDSliceValues(squeeze(mean(RegularTotalPressure/1000)),RegularXGrid,RegularZGrid,D,
    TargetRe,floor(min(min(min(RegularTotalPressure/1000)))*10)/10,ceil(max(max(max(
    RegularTotalPressure/1000)))*10)/10,.1,'Total Pressure [kPa]',NumberofHolesinMesh);
125 hgexport(gcf,[PathToWorkingDirectory,'/images/',CFDCase,'TimeAverageTotalPressure-',num2str(
    min(TimeStepNumbers)),'-',num2str(max(TimeStepNumbers)),'.eps'])
126 %pause
127 close all
```

A.4.12 PlotMultipleNuSpanwiseAverages.m

```
1  clc;
2  clear global;
3  clear all;
4  close all;
5  tic;
6
7  %add path for some other scripts used
8  addpath /home/andy/Desktop/CFD/CFDScripts/      %this is used because the run command
       changes the path and messes all other paths defined up
9  addpath /home/andy/Desktop/CFD/Experimental/ExperimentalDataCopy/impingement/analysis/
10 addpath /home/andy/Desktop/CFD/Experimental/scripts/
11
12 %need to setup path so know where to save files to
13 CFDCases='CompletedCFDCases';
14 BasePath='../../';
15 PathToWorkingDirectory=[BasePath,CFDCases,'/Post/SpanwiseAverages/'];
16 mkdir([PathToWorkingDirectory])
17
18
19
20 %plot the unsteady cases
21 UnsteadyCaseNumbersSelected={
22         %'casenumber1',timestepnumbers1,'casenumber2',timestepnumbres2,'CFD marker','
               Experimental Marker','CFD label text','experimental yes or no', 'experimental
               label text',NumberofRegularlySpacedPointsX,NumberofRegularlySpacedPointsY
23         %make the case with the lowest x/D range last, also make the last case plot
               experimental
24         'CFD0048',227:260,'CFD0051',261:727,'+','o','11 Hole - y/D=4.03 - Numerical','no','N
               /A',650,100
25         'CFD0052',227:329,'CFD0059',330:727,'x','o','11 Hole - y/D=3 - Numerical','yes','
               Center Row Experimental',650,100
26         'CFD0046',227:727,'',0,'.','o','Single Hole Repeating Case I - Numerical','no','N/A'
               ,65,100
27         'CFD0045',227:727,'',0,'d','o','Single Hole Repeating Case II - Numerical','no','
               Center Row Experimental',65,100
28         };
29
30 for CurrentCase=1:size(UnsteadyCaseNumbersSelected,1)
31         clear RegularNu;          %clear out some data from memory. variable will just be
               overwritten anyway on the next line, but this saves matlab from putting data
               into virtual memory (if ram is exausted) before it is overwritten
32         [RegularNu,RegularXGrid,RegularYGrid,D,TargetRe,UnsteadyTimeStep,HoverD,~,
               NumberofHolesinMesh]=PlotNuSpanwiseAverageExperimentalAndCFDUnsteady(
               UnsteadyCaseNumbersSelected{CurrentCase,1},UnsteadyCaseNumbersSelected{
               CurrentCase,2},UnsteadyCaseNumbersSelected{CurrentCase,3},
               UnsteadyCaseNumbersSelected{CurrentCase,4},'',0,UnsteadyCaseNumbersSelected{
               CurrentCase,5},UnsteadyCaseNumbersSelected{CurrentCase,6},
               UnsteadyCaseNumbersSelected{CurrentCase,10},[0.0047625*-1.5+0.0047625*1.5*2/
               UnsteadyCaseNumbersSelected{CurrentCase,11}:0.0047625*1.5*2/
               UnsteadyCaseNumbersSelected{CurrentCase,11}:0.0047625*1.5-0.0047625*1.5*2/
               UnsteadyCaseNumbersSelected{CurrentCase,11}],UnsteadyCaseNumbersSelected{
               CurrentCase,7},UnsteadyCaseNumbersSelected{CurrentCase,8},
               UnsteadyCaseNumbersSelected{CurrentCase,9});
33         ylim([0 65])     %add a little space so the legend isn't on top of things
34         hgexport(gcf,[PathToWorkingDirectory,'SpanwiseAverages-',num2str(CurrentCase),'-',
               num2str(min(UnsteadyCaseNumbersSelected{CurrentCase,2})),'-',num2str(max(
```

```
            UnsteadyCaseNumbersSelected{CurrentCase,2})),'-',num2str(min(
            UnsteadyCaseNumbersSelected{CurrentCase,4})),'-',num2str(max(
            UnsteadyCaseNumbersSelected{CurrentCase,4})),'.eps'])
end
%ylim([min(get(gca,'YTick')) (max(get(gca,'YTick'))+15)])        %add a little space so the legend isn't on top of things


%plot the steady cases
%although there are not timesteps associated, the file name is based on the last unsteady case plotted's time step.
CaseNumbersSelected={
        %'casenumber','CFD marker', 'Experimental Marker','CFD Label Text','experimental yes or nothing','Experimental Label Text',GridPoints,'NuFilenamesuffix','SliceFilenamesuffix',NumberofRegularlySpacedPointsX,NumberofRegularlySpacedPointsY
        'CFD0055','p','o','Single Hole CFD - 12.6 million points - 1.3^2x','no','Center Hole Experimental',12.6,'-SecondOrderSteadyShouldBeConverged-HeatedSurface.csv','-SecondOrderSteadyShouldBeConverged-CenterPlane.csv',65,100
        };

for CurrentCase=1:size(CaseNumbersSelected,1)
        [RegularNu(CurrentCase,:,:),RegularXGrid,RegularYGrid,TargetRe,D,HoverD,~,NumberofHolesinMesh]=PlotNuSpanwiseAverageExperimentalAndCFD(CaseNumbersSelected{CurrentCase,1},CaseNumbersSelected{CurrentCase,2},CaseNumbersSelected{CurrentCase,3},CaseNumbersSelected{CurrentCase,10},CaseNumbersSelected{CurrentCase,11},CaseNumbersSelected{CurrentCase,8},CaseNumbersSelected{CurrentCase,4},CaseNumbersSelected{CurrentCase,5},CaseNumbersSelected{CurrentCase,6});
        ylim([0 75])    %add a little space so the legend isn't on top of things
        hgexport(gcf,[PathToWorkingDirectory,'SpanwiseAverages-',num2str(CurrentCase+size(UnsteadyCaseNumbersSelected,1)),'-',num2str(min(UnsteadyCaseNumbersSelected{size(UnsteadyCaseNumbersSelected,1),2})),'-',num2str(max(UnsteadyCaseNumbersSelected{size(UnsteadyCaseNumbersSelected,1),2})),num2str(min(UnsteadyCaseNumbersSelected{size(UnsteadyCaseNumbersSelected,1),4})),'-',num2str(max(UnsteadyCaseNumbersSelected{size(UnsteadyCaseNumbersSelected,1),4})),'.eps'])
end

hours=toc/3600
```

A.4.13 PlotNuSpanwiseAverageExperimentalAndCFD.m

```
function [RegularNu,RegularXGrid,RegularYGrid,TargetRe,D,HoverD,Regularyplus,NumberofHolesinMesh]=PlotNuSpanwiseAverageExperimentalAndCFD(CFDCase,CFDMarker,ExperimentalMarker,NumberofRegularlySpacedPointsX,NumberofRegularlySpacedPointsY,DataFileSuffix,LabelTextCFD,ExperimentalYes,LabelTextExperimental)
    %main reason this function is used is because it automatically clears all variables after plotting, and it truncates the experimental data to the range of the CFD data if not doing all holes.
    %it also automatically finds the correct experimental case to compare to

    %if want this script to compare as a precentage, need to add another input which is the raw data to compare to

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
    %process the CFD data and make it a regularly spaced grid.
    [RegularNuInstant,RegularXGrid,RegularYGrid,D,TargetRe,ExperimentalCaseNumber,RawExperimentalDataPath,CaseListDataFile,~,HoverD,RegularyplusInstant,NumberofHolesinMesh]=GetRegularCFDSurfaceNu(CFDCase,NumberofRegularlySpacedPointsX,NumberofRegularlySpacedPointsY,DataFileSuffix);
    RegularNu=RegularNuInstant(RegularXGrid,RegularYGrid);
    Regularyplus=RegularyplusInstant(RegularXGrid,RegularYGrid);
    SpanwiseAverageNu=mean(RegularNu);
    MinXDSingleHole=round(min(min(RegularXGrid))/D*10)/10;  %need to round because round off error makes the grid points not exactly and also GetRegularCFDSurfaceNu throws out the end points
```

```
        MaxXDSingleHole=round(max(max(RegularXGrid))/D*10)/10;
        PlotNuSpanwiseAverage(SpanwiseAverageNu,RegularXGrid(1,:),D,TargetRe,MinXDSingleHole,MaxXDSingleHole,CFDMarker,LabelTextCFD,['␣-␣Re=',thousands(TargetRe)])

        if strcmp(ExperimentalYes,'yes')
                %process the raw experimental data, truncate it, and plot, and add an extra label.
                [OverallAverageNu,TargetRe,XExperimental,YExperimental,D,MinXD,MaxXD,MeasurementDomainHeight,TimeAverageNuExperimental,Nu]=ProcessDatausingPlotFunction(ExperimentalCaseNumber,RawExperimentalDataPath,CaseListDataFile,'+');
                TimeAverageNuExperimentalInterp=interp2(XExperimental,YExperimental,TimeAverageNuExperimental,XExperimental,RegularYGrid(:,1)); %use an interpolating function to upsample the experimental data to the numerical data and then truncate to the region of interest.
                SpanAndTimeAverageNuExperimentalInterp=mean(TimeAverageNuExperimentalInterp);
                PlotNuSpanwiseAverage(SpanAndTimeAverageNuExperimentalInterp,XExperimental,D,TargetRe,MinXDSingleHole,MaxXDSingleHole,ExperimentalMarker,LabelTextExperimental,['␣-␣Re=',thousands(TargetRe)])
        end

        %override a few plot parameters for the single hole case
        global SpanwiseAverageFigure
        if NumberofHolesinMesh==1
                set(get(SpanwiseAverageFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick','on','YMinorTick','on','XTick',[-1.5:.5:1.5])
                set(SpanwiseAverageFigure,'OuterPosition',[30 30 622 650])
        end

end
```

A.4.14 PlotNuSpanwiseAverageExperimentalAndCFDUnsteady.m

```
function [RegularNu,RegularXGrid,RegularYGrid,D,TargetRe,UnsteadyTimeStep,HoverD,Regularyplus,NumberofHolesinMesh]=PlotNuSpanwiseAverageExperimentalAndCFDUnsteady(CFDCase1,CFDCase1TimeSteps,CFDCase2,CFDCase2TimeSteps,CFDCase3,CFDCase3TimeSteps,CFDMarker,ExperimentalMarker,RegularlySpacedPointsX,RegularlySpacedPointsY,LabelTextCFD,ExperimentalYes,LabelTextExperimental)
        %main reason this function is used is because it automatically clears all variables after plotting, and it truncates the experimental data to the range of the CFD data if not doing all holes.
        %it also automatically finds the correct experimental case to compare to

        %if want this script to compare as a precentage, need to add another input which is the raw data to compare to

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
        %process the CFD data and make it a regularly spaced grid. count down in for loops so everything goes faster because memory is already preallocated.

        %do case 1
        for x=size(CFDCase1TimeSteps,2):-1:1
                disp(['Current␣Time␣Step␣Number␣=␣',num2str(CFDCase1TimeSteps(x))]);
                [RegularNuInstant,RegularXGrid,RegularYGrid,D,TargetRe,ExperimentalCaseNumber,RawExperimentalDataPath,CaseListDataFile,UnsteadyTimeStep,HoverD,RegularyplusInstant,NumberofHolesinMesh]=GetRegularCFDSurfaceNu(CFDCase1,RegularlySpacedPointsX,RegularlySpacedPointsY,['-HeatedSurface-SecondOrderUnSteady-1-',num2str(CFDCase1TimeSteps(x),'%05.0f'),'.csv']);
                RegularNu(x,:,:)=RegularNuInstant(RegularXGrid,RegularYGrid);
                Regularyplus(x,:,:)=RegularyplusInstant(RegularXGrid,RegularYGrid);
        end

        %do case 2
        if ~strcmp(CFDCase2,'')
                for x=size(CFDCase2TimeSteps,2):-1:1
                        disp(['Current␣Time␣Step␣Number␣=␣',num2str(CFDCase2TimeSteps(x))]);
```

```
22                          [RegularNuInstant,RegularXGrid,RegularYGrid,D,TargetRe,
                                ExperimentalCaseNumber,RawExperimentalDataPath,CaseListDataFile,
                                UnsteadyTimeStep,HoverD,RegularyplusInstant,NumberofHolesinMesh
                                ]=GetRegularCFDSurfaceNu(CFDCase2,RegularlySpacedPointsX,
                                RegularlySpacedPointsY,['-HeatedSurface-SecondOrderUnSteady-1-',
                                num2str(CFDCase2TimeSteps(x),'%05.0f'),'.csv']);
23                          RegularNu(x+size(CFDCase1TimeSteps,2),:,:)=RegularNuInstant(
                                RegularXGrid,RegularYGrid);
24                          Regularyplus(x+size(CFDCase1TimeSteps,2),:,:)=RegularyplusInstant(
                                RegularXGrid,RegularYGrid);
25                  end
26          end
27
28          %never added code for case 3
29
30          TimeAverageNu=squeeze(mean(RegularNu));
31          SpanwiseTimeAverageNu=mean(TimeAverageNu);
32
33
34          %assume all time steps and cases should have the same grid
35          MinXDSingleHole=round(min(min(RegularXGrid))/D*10)/10;  %need to round because round
                off error makes the grid points not exactly and also GetRegularCFDSurfaceNu
                throws out the end points
36          MaxXDSingleHole=round(max(max(RegularXGrid))/D*10)/10;
37
38          PlotNuSpanwiseAverage(SpanwiseTimeAverageNu,RegularXGrid(1,:),D,TargetRe,
                MinXDSingleHole,MaxXDSingleHole,CFDMarker,LabelTextCFD,[' - Re=',thousands(
                TargetRe)])
39
40          if strcmp(ExperimentalYes,'yes')
41                  %process the raw experimental data, truncate it, and plot, and add an extra
                        label.
42                  addpath ../../ExperimentalDataCopy/impingement/analysis/        %needed to
                        use the process data script to get experimental data.
43                  [OverallAverageNu,TargetRe,XExperimental,YExperimental,D,MinXD,MaxXD,
                        MeasurementDomainHeight,TimeAverageNuExperimental,Nu]=
                        ProcessDatausingPlotFunction(ExperimentalCaseNumber,
                        RawExperimentalDataPath,CaseListDataFile,'+');
44                  TimeAverageNuExperimentalInterp=interp2(XExperimental,YExperimental,
                        TimeAverageNuExperimental,XExperimental,RegularYGrid(:,1)); %use an
                        interpolating function to upsample the experimental data to the
                        numerical data and then truncate to the region of interest.
45                  SpanAndTimeAverageNuExperimentalInterp=mean(TimeAverageNuExperimentalInterp)
                        ;
46                  PlotNuSpanwiseAverage(SpanAndTimeAverageNuExperimentalInterp,XExperimental,D
                        ,TargetRe,MinXDSingleHole,MaxXDSingleHole,ExperimentalMarker,
                        LabelTextExperimental,[' - Re=',thousands(TargetRe)])
47          end
48
49          %override a few plot parameters for the single hole case
50          global SpanwiseAverageFigure
51          if NumberofHolesinMesh==1
52                  set(get(SpanwiseAverageFigure,'CurrentAxes'),'XGrid','on','YGrid','on','
                        XMinorTick','on','YMinorTick','on','XTick',[-1.5:.5:1.5])
53                  set(SpanwiseAverageFigure,'OuterPosition',[30 30 622 650])
54          end
55
56  end
```

A.4.15 PlotNuSurfaceCFD.m

```
1   function [ContourFigure,ColorbarHandle]=PlotNuSurfaceCFD(Nu,RegularXGrid,RegularYGrid,D,
        TargetRe,MinNu,MaxNu,NuStep,HoverD,NumberofHolesinMesh,ContourScaleLabel,Time)
2
3           MinXD=round(min(min(RegularXGrid))/D*10)/10;     %need to round because round off
                error makes the grid points not exactly and also GetRegularCFDSurfaceNu throws
                out the end points
4           MaxXD=round(max(max(RegularXGrid))/D*10)/10;
```

```
5           MinYD=round(min(min(RegularYGrid))/D*10)/10;
6           MaxYD=round(max(max(RegularYGrid))/D*10)/10;
7
8           if ~exist('ContourScaleLabel','var')
9                   ContourScaleLabel='Nu';
10          end
11
12          if exist('Time','var')
13                  [ContourFigure,ColorbarHandle]=PlotNuContour(Nu,RegularXGrid(1,:),
                        RegularYGrid(:,1),D,TargetRe,MinXD,MaxXD,MinYD,MaxYD,MinNu,MaxNu,NuStep,
                        HoverD,ContourScaleLabel,Time);
14          else
15                  [ContourFigure,ColorbarHandle]=PlotNuContour(Nu,RegularXGrid(1,:),
                        RegularYGrid(:,1),D,TargetRe,MinXD,MaxXD,MinYD,MaxYD,MinNu,MaxNu,NuStep,
                        HoverD,ContourScaleLabel);
16          end
17
18          if NumberofHolesinMesh==1
19                  set(get(ContourFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick',
                        'on','YMinorTick','on','XTick',[-1.5:.5:1.5],'YTick',[-1.5:.5:1.5])
20                  %make the figure a little wider since the label is cut off
21                  CurrentDimensions=get(ContourFigure,'OuterPosition');
22                  CurrentDimensions(3)=CurrentDimensions(3)+200;
23                  set(ContourFigure,'OuterPosition',CurrentDimensions);
24          elseif NumberofHolesinMesh==11
25                  set(get(ContourFigure,'CurrentAxes'),'XGrid','on','YGrid','on','XMinorTick',
                        'on','YMinorTick','on','XTick',[-15:3:15],'YTick',[-1.5:1.5:1.5])
26          end
27  end
```

A.4.16 VectorProfile.m

```
1   function VectorProfile(Regularvx,Regularvz,RegularXGrid,RegularZGrid,XMinVector,XMaxVector,
        XCount,ZMinVector,ZMaxVector,ZCount,D,ScaleFactor,NumberofHolesinMesh)
2           %display velocity vectors inside a rectangle with a given spacing.
3
4
5
6           XResolution=(XMaxVector-XMinVector)/XCount;
7           if XResolution==0
8                   RegularXVector=XMinVector;
9           else
10                  RegularXVector=XMinVector:XResolution:XMaxVector;
11          end
12          ZResolution=(ZMaxVector-ZMinVector)/ZCount;
13          if ZResolution==0
14                  RegularZVector=ZMinVector;
15          else
16                  RegularZVector=ZMinVector:ZResolution:ZMaxVector;
17          end
18          [RegularXGridVector,RegularZGridVector]=meshgrid(RegularXVector,RegularZVector);
19
20          RegularXGridVectorReshaped=reshape(RegularXGridVector,1,[]);
21          RegularZGridVectorReshaped=reshape(RegularZGridVector,1,[]);
22          RegularvxReshaped=reshape(interp2(RegularXGrid,RegularZGrid,Regularvx,
                RegularXGridVector,RegularZGridVector),1,[]);
23          RegularvzReshaped=reshape(interp2(RegularXGrid,RegularZGrid,Regularvz,
                RegularXGridVector,RegularZGridVector),1,[]);
24
25
26          %undo meshgrid and do some other stupid stuff to setup some other values so can use
                with InsideTheLowerGridSingleHole and InsideThePressureChamberGrid11Hole and
                InsideTheLowerGrid11Hole
27          X=reshape(RegularXGrid,1,[]);
28          Z=reshape(RegularZGrid,1,[]);
29          X(find(isnan(reshape(Regularvx,1,[])))=0;
30          Z(find(isnan(reshape(Regularvx,1,[])))=0;
31          XMin=min(X);
```

```
32          XMax=max(X);
33          ZMin=min(Z);
34          ZMax=max(Z);
35
36
37          %determine whether points are inside of the mesh or not
38          %call code that defines the perimeter of the mesh
39          if NumberofHolesinMesh==1
40                  InsideTheLowerGridSingleHole
41                  %since there is no pressure chamber for this case, just set the values to a
                        scalar zero
42                  INPressureChamber=0;
43                  ONPressureChamber=0;
44          elseif NumberofHolesinMesh==11
45                  InsideTheLowerGrid11Hole
46                  InsideThePressureChamberGrid11Hole
47                  %see if in the pressure chamber
48                  [INPressureChamber,ONPressureChamber]=inpolygon(RegularXGridReshapedVector,
                        RegularZGridReshapedVector,InsideThePressureChamberGrid(:,1),
                        InsideThePressureChamberGrid(:,2));
49          end
50
51          %see if in the lower grid
52          [INLower,ONLower]=inpolygon(RegularXGridVectorReshaped,RegularZGridVectorReshaped,
                InsideTheLowerGrid(:,1),InsideTheLowerGrid(:,2));
53
54          %now add both together
55          INorON=INLower+ONLower+INPressureChamber+ONPressureChamber;
56
57
58
59          RegularvxReshaped(find(INorON==0))=NaN;
60          RegularvzReshaped(find(INorON==0))=NaN;
61
62          %now remove all NaN because quiver will put a dot for NaN
63          OriginalISNANTest=(isnan(RegularvxReshaped)==0)|(isnan(RegularvzReshaped)==0);
64          RegularXGridVectorReshaped=RegularXGridVectorReshaped(OriginalISNANTest);
65          RegularZGridVectorReshaped=RegularZGridVectorReshaped(OriginalISNANTest);
66          RegularvxReshaped=RegularvxReshaped(OriginalISNANTest);
67          RegularvzReshaped=RegularvzReshaped(OriginalISNANTest);
68
69          %now draw all arrows.
70          quiver(RegularXGridVectorReshaped/D,RegularZGridVectorReshaped/D,RegularvxReshaped*
                ScaleFactor,RegularvzReshaped*ScaleFactor,0,'b')
71  end
```

A.4.17 ValuesInGrid.m

```
1   function [RegularParameterMatrix]=ValuesInGrid(RegularParameter,RegularXGrid,RegularZGrid,
        INorON)
2           %need to convert all values outside of the mesh to NaN so the countour plot doesn't
                show them
3           RegularParameterReshaped=reshape(RegularParameter(RegularXGrid,RegularZGrid),1,[]);
4           RegularParameterReshaped(find(INorON==0))=NaN;
5           RegularParameterMatrix=reshape(RegularParameterReshaped,size(RegularParameter(
                RegularXGrid,RegularZGrid)));
6           %done converting all values to NaN outside of th mesh
7   end
```

A.4.18 InsideTheLowerGridSingleHole.m

```
1   hole=1;
2   InsideTheLowerGrid=[
3           XMin,max(Z(find(X==XMin)))
4           (min(X(find(Z==ZMax)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
5           (min(X(find(Z==ZMax)))+D*((hole-2)*3+3)),ZMax
6           (min(X(find(Z==ZMax)))+D*((hole-2)*3+4)),ZMax
7           (min(X(find(Z==ZMax)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
```

```
8           XMax,max(Z(find(X==XMin)))
9           XMax,ZMin
10          XMin,ZMin
11          ];
```

A.4.19 InsideTheLowerGrid11Hole.m

```
1   %parameters that define the border of the grid. (11 hole, no pressure chamber)
2   InsideTheLowerGrid=[
3           XMin,max(Z(find(X==XMin)))
4
5           ];
6
7   hole=1;
8   InsideTheLowerGrid=[InsideTheLowerGrid,
9           (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
10          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
11          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
12          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
13          ];
14
15  hole=2;
16  InsideTheLowerGrid=[InsideTheLowerGrid,
17          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
18          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
19          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
20          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
21          ];
22
23  hole=3;
24  InsideTheLowerGrid=[InsideTheLowerGrid,
25          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
26          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
27          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
28          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
29          ];
30
31  hole=4;
32  InsideTheLowerGrid=[InsideTheLowerGrid,
33          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
34          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
35          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
36          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
37          ];
38
39  hole=5;
40  InsideTheLowerGrid=[InsideTheLowerGrid,
41          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
42          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
43          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
44          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
45          ];
46
47  hole=6;
48  InsideTheLowerGrid=[InsideTheLowerGrid,
49          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
50          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
51          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
52          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
53          ];
54
55  hole=7;
56  InsideTheLowerGrid=[InsideTheLowerGrid,
57          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
58          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
59          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
60          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
61          ];
```

```
62
63  hole=8;
64  InsideTheLowerGrid=[InsideTheLowerGrid,
65          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
66          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
67          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
68          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
69          ];
70
71  hole=9;
72  InsideTheLowerGrid=[InsideTheLowerGrid,
73          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
74          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
75          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
76          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
77          ];
78
79  hole=10;
80  InsideTheLowerGrid=[InsideTheLowerGrid,
81          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
82          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
83          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
84          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
85          ];
86
87  hole=11;
88  InsideTheLowerGrid=[InsideTheLowerGrid,
89          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),max(Z(find(X==XMin)))
90          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+3)),0
91          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),0
92          (min(X(find(Z<-.5*D&Z>-1.5*D)))+D*((hole-2)*3+4)),max(Z(find(X==XMin)))
93
94          XMax,max(Z(find(X==XMin)))
95          XMax,ZMin
96          XMin,ZMin
97          ];
```

A.4.20 InsideThePressureChamberGrid11Hole.m

```
1  %parameters that define the border of the grid. (11 hole, no pressure chamber)
2  InsideThePressureChamberGrid=[
3          min(X(find(Z>.5*D&Z<1.5*D))),0
4          min(X(find(Z>.5*D&Z<1.5*D))),ZMax
5          max(X(find(Z>.5*D&Z<1.5*D))),ZMax
6          max(X(find(Z>.5*D&Z<1.5*D))),0
7          ];
```

A.4.21 ReadCFDCaseListSpreadsheet.m

```
1  PathToCFDCaseListDataFilePrefix=[BasePath,'CFDCaseList'];
2  PathToCFDCases=[BasePath,CFDCases,'/'];
3
4
5  %convert ods file to tab delimited text file
6  %first clear out an environmental variable matlab sets that messes this program up
7  LD_LIBRARY_PATH_original=getenv('LD_LIBRARY_PATH');
8  setenv('LD_LIBRARY_PATH','');
9  !unoconv --listener&
10 pause(5)
11 %now change the environmental variable back
12 setenv('LD_LIBRARY_PATH',LD_LIBRARY_PATH_original)
13 eval(['!unoconv -f csv --stdout ',PathToCFDCaseListDataFilePrefix,'.ods|sed ''s/,/\t/g''|sed
       ''s/"//g''>',PathToCFDCaseListDataFilePrefix,'.txt'])
14
15
16
17
18
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%read additional descriptive data for this run
filehandle=fopen([PathToCFDCaseListDataFilePrefix,'.txt']);
%note: if the number of delimiters is less than the total number of fields,
%matlab just breaks the current row at the last defined delimiter
%such that the next field is on a new row.
%because of this, extra delmiters are put in so that if a new field is
%added, it will still work okay
CFDCaseListData = textscan(filehandle, '%s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s %s','CollectOutput',1,'delimiter','\t');
fclose(filehandle);     %close file
CFDCaseListData=CFDCaseListData{1,1};  %simplify variable a little

%define variable and column mappings
CFDCaseListDataVariables={
        'Field Name','Column Number','Units','String?'
        'CFDCaseNumber',1,'','no'
        'ExperimentalCaseNumber',2,'','yes'
        'CFDCaseToContinue',3,'','yes'
        'CaseFileName',4,'','yes'
        'DataFileName',5,'','yes'
%       'FilesStillOnHawkWorkspace',6,'','yes'
        'NumberofHolesinMesh',6,'','no'
        'Unsteady',7,'','yes'
        'NumberofCPUs',8,'','no'
        'WallTime',9,'','yes'
        'NoEnergyIterations',10,'','yes'
        'FirstOrderIterations',11,'','yes'
        'SecondOrderIterations',12,'','yes'
        'NumberOfFormerTimeStepsToSave',13,'','no'
        'UnsteadyTimeSteps',14,'','yes'
        'UnsteadyTimeStep',15,'','yes'
        'ResetUnsteadyStatistics',16,'','yes'
        'WallConduction',17,'','yes'
        'HeatedSurfaceZone1',18,'','yes'
        'HeatedSurfaceZone2',19,'','yes'
        'UnHeatedBottomSurfaceZone',20,'','yes'
        'InletZone',21,'','yes'
        'OutletZone',22,'','yes'
        'InteriorZone',23,'','yes'
        'JetCore',24,'','yes'
        'TopWallsZone',25,'','yes'
        'FirstRepeatingBoundary',26,'','yes'
        'SecondRepeatingBoundary',27,'','yes'
        'InletTurbulentIntensity',28,'','yes'
        'LinePlotCoordinatesStart',29,'mm','no'
        'LinePlotCoordinatesStop',30,'mm','no'
        'MaxPlotTemperature',31,'K','yes'
        'MinPlotTemperature',32,'K','yes'
        };



%double check values are mapped properly
disp(' ')
disp('Check and make sure each field is properly mapped to the correct variable')
for CFDCaseListDataVariableCount=2:size(CFDCaseListDataVariables,1)
        %double check values are mapped properly
        disp([CFDCaseListDataVariables{CFDCaseListDataVariableCount,1},'[',
            CFDCaseListDataVariables{CFDCaseListDataVariableCount,3}, '] <-----------> ',
            CFDCaseListData{1,CFDCaseListDataVariables{CFDCaseListDataVariableCount,2}}]);
end
disp('please wait')
for CFDCaseListDataCount=2:size(CFDCaseListData,1)
        if strcmp(CFDCaseListData(CFDCaseListDataCount,1),CFDCase)
                for CFDCaseListDataVariableCount=2:size(CFDCaseListDataVariables,1)
```

```
                        if strcmp(CFDCaseListDataVariables{CFDCaseListDataVariableCount,4},'no')
                            eval([CFDCaseListDataVariables{CFDCaseListDataVariableCount,1},'=str2num(CFDCaseListData{CFDCaseListDataCount,CFDCaseListDataVariables{CFDCaseListDataVariableCount,2}});']);
                        else
                            eval([CFDCaseListDataVariables{CFDCaseListDataVariableCount,1},'=CFDCaseListData{CFDCaseListDataCount,CFDCaseListDataVariables{CFDCaseListDataVariableCount,2}};']);
                        end
                end
        end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

CFDCaseToContinuePathSmallFiles=[PathToCFDCases,'/SmallFiles/',CFDCaseToContinue,'/'];
CFDCaseToContinuePathOutputData=[PathToCFDCases,'/OutputData/',CFDCaseToContinue,'/'];
```