Fall 2014

Non-intrusive two-phase flow regime identification and transport characterization in microchannels subject to uniform and non-uniform heat input

Susan N. Ritchey
Purdue University

Graduate School Form 30
(Updated 11/20/2014)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Susan N. Ritchey

Entitled
Non-Intrusive Two-Phase Flow Regime Identification and Transport Characterization in Microchannels Subject to Uniform and Non-Uniform Heat Input

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Suresh V. Garimella

Jayathi Y. Murthy

Jong H. Choi

Shripad T. Revankar

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Suresh V. Garimella

Approved by: Ganesh Subbarayan 12/03/2014

Head of the Department Graduate Program Date

NON-INTRUSIVE TWO-PHASE FLOW REGIME IDENTIFICATION AND TRANSPORT CHARACTERIZATION IN MICROCHANNELS SUBJECT TO UNIFORM AND NON-UNIFORM HEAT INPUT

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Susan N. Ritchey

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2014

Purdue University

West Lafayette, Indiana

To my husband, who has always supported me in everything I do.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Suresh Garimella, for his support and advice throughout my studies at Purdue University. I would also like to thank my committee, Professors Jayathi Murthy, Jong Hyun Choi, and Shripad Revankar for their guidance in this work.

I would like to extend a special thank you to my labmates for their help and feedback on my research. I would especially like to thank Professor Justin Weibel. His help was greatly appreciated.

Finally, I would like to thank my husband, Philip Ritchey, my parents, Frank and Susan Williams, and my sister, Christine Williams, for their continued support. I could not have done this without them.

Finanacial support for this work was provided by the Cooling Technologies Research Center, a National Science Foundation Industry/University Cooperative Research Center at Purdue University and by the Office of Naval Research (Grant No. N000141010921).

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Appendix Figure Page

Appendix Figure Page

NOMENCLATURE

$a_{i,j,k}$	coefficient of the ith, jth, kth point
A	area, m^2
A_b	base area of the heat sink, m^2
A_f	wetted area of a fin, m^2
A_w	total wetted area of the microchannels, m^2
$b_{i,j,k}$	coefficient of the i+1, jth, kth point
$c_{i,j,k}$	coefficient of the i-1, jth, kth point
c_p	specific heat, J kg^{-1} K^{-1}
C	electrical capacitance, F; shape factor
C_0	distribution parameter from the drift flux model
Co	Confinement number
d	depth of the microchannel, m
$d_{i,j,k}$	coefficient of the ith, j+1, kth point
D	charge flux, C m^{-2}
D_h	hydraulic diameter, m
$e_{i,j,k}$	coefficient of the ith, j-1, kth point
E_{RMS}	mean square deviation
$f_{i,j,k}$	coefficient of the ith, jth, k+1 point
$g_{i,j,k}$	coefficient of the ith, jth, k-1 point
G	mass flux, kg m^{-2} s^{-1}; admittance, ohm^{-1}
h	heat transfer coefficient, W m^2 K^{-1}
$h_{i,j,k}$	source term for the ith, jth, kth point
h_{fg}	latent heat of vaporization, J kg^{-1}

j	volumetric flux or superficial velocity, m s^{-1}
J	current flux, C m^{-2} s^{-1}
k	relative permittivity or dielectric constant; thermal conductivity, W m^{-1} K^{-1}
L	total length and width of the microchannel heat sink, m
m	variable in fin efficiency calculation, m^{-1}
$\dot{m}$	mass flow rate, kg s^{-1}
N	number of elements; number of microchannels
Pr	Prandtl number
$\dot{q}$	heat transfer rate, W
q''	heat flux, W m^{-2}
Q	power input, W
R	electrical resistance, ohm
$\Re$	resistive impedance, the real part of the electrical impedance, ohm
Re	Reynolds number
t	total thickness of the microchannel heat sink, m
T	temperature, °C
v_{gj}	drift velocity, m s^{-1}
V	voltage, V
w	width of the microchannel, m
w_f	width of a fin, m
x	vapor quality
x^*	non-dimensional weighting factor
X	reactive impedance, the imaginary part of the electrical impedance, ohm
Z	electrical impedance, ohm
z_{th}^*	non-dimensional thermal entrance length

GREEK SYMBOLS

Δx	x-dimension of the cell
Δy	y-dimension of the cell
Δz	z-dimension of the cell

Φ	degree of nonuniformity
α	void fraction; aspect ratio
β	ratio of gas volumetric flux to total volumetric flux, homogeneous equilibrium model void fraction
ε_0	permittivity of free space, F m^{-1}
η_f	fin efficiency
η_0	overall surface efficiency
ρ	electrical resistivity, ohm m; density, kg m^{-3}
σ	surface tension, N m^{-1}
ϕ	electric potential, V
ω	angular frequency, rad s^{-1}

SUBSCRIPTS

$*$	normalized
∞	fully developed
0	liquid
1	gas
b	base; cell to the back
cal	calibration
$cond$	conduction
$conv$	convective
d	diode
e	effective
$exit$	exit
f	liquid; the cell to the front
FB	flow boiling
g	gas
gen	generated
$high$	peak heater element region
i	ith point; heater element in the flow direction

im	from image analysis
imp	impedance meter
in	inlet
j	jth point; heater element in the transverse direction
k	kth point
l	cell to the left; liquid
$loss$	heat loss
low	background heater element region
m	mixture; cell to the bottom
n	n-phase
NB	nucleate boiling
net	net or total
ONB	onset of nucleate boiling
p	current cell
r	cell to the right
s	surface
sp	single-phase
sat	saturation
Si	silicon
t	time averaged; cell to the top
tp	two-phase
w	wall
v	vapor
V	volume averaged

ABSTRACT

Ritchey, Susan N., Ph.D., Purdue University, December 2014. Non-Intrusive Two-Phase Flow Regime Identification and Transport Characterization in Microchannels Subject to Uniform and Non-Uniform Heat Input. Major Professor: Suresh V. Garimella, School of Mechanical Engineering.

Direct integration of compact microchannel heat sinks is an attractive thermal management solution for the dissipation of high heat fluxes, specifically under boiling conditions that provide high rates of heat transfer at a uniform heat sink temperature. Under two-phase flow conditions, the heat transfer and pressure drop are a function of the local flow regime. Development of sensors that detect local void fraction and flow regimes may enable better understanding of the fundamental flow phenomena.

The void fraction in air-water two-phase adiabatic flow in a microchannel is measured in this work using a custom-designed impedance-based sensor with electrodes on opposing walls of a single microchannel, a ‘crosswise’ geometry. The impedance response of the sensor is calibrated against the time-averaged void fraction determined via high-speed flow visualizations. The temporal signal is depicted as a probability density function that is used for quantitative determination of two-phase flow regimes using a Kohonen Self-Organizing Map.

To characterize the sensor impedance response, numerical simulations are implemented in two- and three-dimensions. Electrical simulations of the crosswise electrode geometry are performed to acquire both instantaneous and time-averaged responses. For arbitrarily defined voids, the shape and distribution has no effect on the simulated impedance; the relationship between the void fraction and impedance is found to be non-linear. Time-averaged three-dimensional impedance simulations are in good agreement with the experimental data.

A second set of experiments are performed using multiple electrodes placed along the flow direction of a single microchannel wall, a ‘streamwise’ geometry. Multiple water electrical conductivities are tested, and an optimal range between 100 and 175 μS/cm is found to provide maximum instrument sensitivity. The dependency of the impedance output on water conductivity is characterized to fit all of the data to a single calibration curve, independent of water conductivity.

One application where the determination of the local void fraction is important is in the case of non-uniform heating in microchannels. An experimental investigation is performed to explore flow boiling phenomena in a microchannel heat sink with hotspots, as well as non-uniform streamwise and transverse heating conditions across the entire heat sink. Local heat transfer coefficients and wall temperatures are measured while the location of boiling incipience is observed via high-speed visualizations of the flow. It is found that even though the substrate thickness beneath the microchannels is very small (200 um), significant lateral conduction occurs and must be accounted for in the calculation of the local heat flux imposed. For non-uniform heat input profiles, with peak heat fluxes along the central streamwise and transverse directions, it is found that the local flow regimes, heat transfer coefficients, and wall temperatures deviate significantly from a uniformly heated case.

A simple computational model is developed to predict the thermal performance of a microchannel heat sink with an imposed non-uniform heating profile. While the model underpredicts the base temperatures and overpredicts the heat transfer coefficients, the trends agree with experimental data. For the cases investigated with the model, flow non-uniformities between the channels are estimated using image analysis of high-speed videos taken during the experiments. It is observed that flow maldistribution must be taken into account in the model for heating profiles that are prone to flow maldistribution in order to improve the match to experimental data.

Another experimental investigation is performed to measure the critical heat flux (CHF) in a microchannel heat sink with uniform heating and various hotspot heating locations. It is found that a hotspot spanning the entire length of the heat sink in the flow direction produces the lowest CHF of all the cases investigated due to the flow maldistribution induced by boiling. A single hotspot spanning the heat sink perpendicular to the flow direction produces different CHF values based on its streamwise location. The visualizations reveal that CHF occurs when there is a sudden and unalleviated upstream expansion of vapor in one or more channels above the hotspot, causing the local wall temperature to rapidly increase. The proximity of the hotspot to the inlet manifold, which communicates between all channels and can relieve upstream vapor expansion, appears to determine the resiliency of the heat sink to CHF.

Non-uniform heating profiles often found in actual applications greatly affect the thermal performance of microchannel heat sinks. Measuring the void fraction and understanding how the location of hotspots affects local heat transfer allows for the creation of a computational model to aid future heat sink designs.

CHAPTER 1. INTRODUCTION

1.1 Background

The development of high-power electronics systems for use in commercial, automotive, and military applications has led to an increasing demand for more effective and compact electronics cooling methods. The functionality of microelectronics is increasing while the system packaging volume is decreasing, yielding higher densities of heat generation. Novel methods are required for removing excess heat from these systems. Microchannel heat sinks are an attractive solution due to their compact size and effectiveness at removing high heat fluxes. Microchannel heat sinks can also be integrated directly into semiconductor heat generation sources thus decreasing the overall thermal packaging volume. Additionally, operating under flow boiling conditions allows for higher heat transfer rates and a more uniform temperature profile.

In order to determine heat transfer rates and pressure drops under boiling conditions in microchannels, void fraction and flow regime-dependent correlations are required. Quantitative determination of the flow regime, and direct measurement of the void fraction, will aid in understanding fundamental flow characteristics, and enable the design of future heat sinks. Previously, Serizawa *et al*.[1] performed flow visualizations to determine the void fraction and flow regimes in microchannels. A small electrical impedance-based sensor has the ability to measure the temporal variations of the void fraction and characterize the two-phase flow based on the inherent electrical property differences between gas and liquid phases. Yang *et al.* [2] used electrical impedance void meters to measure the void fraction in a rod bundle. Numerical simulations were performed by Rosa *et al.* [3] to optimize the geometry of an impedance-based sensor in pipes.

The effects of microchannel size, heat flux, and mass flux on the boiling regimes in microchannels has been recently studied; however, correlations and performance models are developed only for uniform heating conditions. Liu *et al.* [4] developed an analytical model to predict the onset of nucleate boiling of water in copper microchannels. Lee and Garimella [5] measured the pressure drop and heat transfer coefficient of water boiling in silicon microchannels, while Bertsch *et al*. [6] measured the heat transfer coefficient for refrigerants in copper microchannels. Both Revellin and Thome [7] and Kosar [8] developed models to predict the critical heat flux in microchannels. Harirchian and Garimella [9,10,11] performed an extensive experimental investigation of the effects of heat flux, mass flux, and channel dimension on boiling heat transfer as well as developed comprehensive flow regime maps. All of this work has been conducted using uniform heating profiles. Thinner and more compact systems prevent the use of thick heat spreading layers to mitigate heat generation non-uniformities at the die level. Thus, non-uniform heat flux profiles are imposed directly on the heat sink base, and impact the two-phase flow characteristics and thermal performance limits.

1.2 Objectives and Major Contributions

The main goals of this work are: (1) to develop a non-intrusive impedance-based void fraction sensor to quantitatively determine the local flow regime in two-phase microchannel flows, (2) to study the effects of non-uniform global substrate heating profiles on two-phase flow through microchannels to better understand their operation under realistic boundary conditions, and (3) to measure the change in the location and quantitative value of the critical heat flux under non-uniform heating conditions.

Experiments are performed for air-water adiabatic two-phase flow in a single microchannel using an impedance-based void fraction sensor to measure the electrical characteristics of multiple flow regimes. High-speed videos are recorded and analyzed to determine the actual void fraction for sensor calibration; a calibration equation is developed between the electrical impedance of the flow and the void fraction. The temporal impedance response of the sensor is processed via a neural network to quantitatively determine the flow regime. Numerical simulations are also performed to predict the response of the sensor using different electrode orientations, void fractions, and void shapes.

Experiments are performed using FC-77 for hotspot as well as non-uniform peak heating conditions imposed on a silicon microchannel heat sink to explore flow boiling phenomena. High-speed videos are also recorded to observe instabilities not present in uniform heating situations, while local wall temperatures and heat transfer coefficients are measured. It is found that these parameters as well as local flow regimes deviate significantly from uniform heating conditions, and the trends are assessed as a function of the increase in the relative magnitude of the nonuniformity between peak and background heat fluxes.

Experiments are performed using HFE-7100 for hotspot heating conditions imposed on a silicon microchannel heat sink to explore the location and quantitative values of the critical heat flux (CHF). The fluid was changed from FC-77 to HFE-7100 in order to reach CHF without exceeding the operational temperature limit of the test chip. High-speed videos are recorded to observe the locations of the critical heat flux simultaneously with measurement of local wall temperatures and heat transfer coefficients. It is found that both the configuration and location of the hotspot significantly affects both the location and magnitude of the critical heat flux.

A simple computational model was developed to predict the behavior of a microchannel heat sink under any non-uniform heating profile. The model contains a three-dimensional conduction analysis in the base of the heat sink, a fin analysis, and employs correlations for the heat transfer coefficient in the microchannels based on the fluid phase. The flow maldistribution was estimated from high-speed videos and incorporated into the model to enable comparison against experimental data.

1.3 Organization of the Document

Chapter 1 described the background information on two-phase flow in microchannels and presented the objectives and major contributions of the current work. Chapter 2 provides a comprehensive literature review. Reviewed topics include flow regime identification and void fraction measurements in microchannel heat sinks, void fraction measurement methods and numerical simulations of impedance-based sensors, and experimental investigations of non-uniform heating conditions in macroscale and microscale flow boiling. Chapter 3 describes the experimental setup for measuring void fraction using crosswise electrodes and presents the results of the experiments. Chapter 4 describes a numerical simulation performed to predict the response of an impedance-based void fraction meter using crosswise electrodes. The results are presented and compared to the previous experimental results. Chapter 5 describes a second experimental setup for measuring void fraction using streamwise electrodes. It presents the results and discusses the sensitivity dependence of the instrument to the electrical conductivity of the liquid phase. Chapter 6 describes the experimental setup and results for hotspot and non-uniform peak heating conditions in a microchannel heat sink. Chapter 7 describes a computational model developed to predict the performance of a microchannel heat sink exposed to non-uniform heating conditions. Chapter 8 describes the results for hotspot heating conditions on the critical heat flux in a microchannel heat sink. Chapter 9 contains a summary of the thesis and suggestions for future work.

CHAPTER 2. LITERATURE REVIEW

2.1 Void Fraction Measurement

Microchannel heat sinks based on boiling and two-phase flow can meet the increasing cooling needs for high-end electronics systems in applications ranging from high-performance computers to avionics and spacecraft to electric vehicles. In order to design and build such heat sinks, a unified model accounting for the prevalent flow regimes is needed to predict the boiling heat transfer rates and pressure drops in microchannels. Flow regime-based correlations are desired in two-phase flow analyses since a single heat transfer correlation does not apply in all flow regimes [12]. A number of studies in recent years have attempted to better understand the flow patterns during boiling in microchannels using various working fluids as reviewed in [13,14,15]. A systematic investigation into the effects of channel size, mass flux, and heat flux on the boiling flow patterns and heat transfer in microchannels was recently performed by Harirchian and Garimella [9,10]. A generalized flow regime map for boiling in microchannels covering a wide range of channel geometries, heat fluxes, and mass fluxes was developed in terms of three nondimensional parameters – Boiling number, Reynolds number, and Bond number – by Harirchian and Garimella [11]. In order to further develop predictive models for flow regime transitions, it is necessary to measure the void fraction in two-phase flow, since the void fraction and its temporal variation is a characteristic of the flow regime. Several studies in the past have relied on flow visualization for the identification of flow regimes as well as for the measurement of void fraction [1,16,17,18]. Even though flow regimes can be determined by observing high-speed movie camera recordings, the method is subjective and cannot be used for conditions in which intermittent phenomena occur, as well as when the aspect ratios of the observed field is such that the mechanisms are obscured from visual observation. In order to overcome these shortcomings, a non-intrusive void fraction measurement technique, which is based on the measurement of electrical impedance, is explored.

Electrical impedance-based void fraction measurements have been successfully performed in the past several decades in macroscale two-phase flows. For cross-sectional area-averaged or volume-averaged measurements, impedance void fraction meters with electrodes flush mounted to the channel walls were used by Asali *et al*. [19], Andreussi *et al*. [20], Tsochatzidis *et al*. [21], Fossa [22], and Mi *et al*. [23]. A theoretical basis for this design is given by Coney [24]. A single conductive ring was used as an electrical impedance tomographic sensor to perform image reconstruction on air-water two-phase flow in a tube [25]. The conductive ring was used in lieu of multiple electrodes to achieve a more homogenous sensitivity distribution throughout the sensing domain. Another experimental study used flush mounted stainless steel ring electrodes in a pipe to determine the liquid hold-up in two-phase flow [20]. Annular, stratified, and bubbly flows were created in a pipe to calibrate the probe and it was found that the impedance method has a large sensitivity to different flow patterns; however a distance between the electrodes in the range of 1.5 to 2.5 pipe diameters provides a good compromise between obtaining a localized measurement and a reading independent of the flow regime.

A similar impedance void fraction meter configuration is adapted to the microscale channels considered in this thesis. The practical implementation of the electronic circuit measures the net electrical admittance, or the inverse of the electrical impedance, of the two-phase mixture. The admittance is a function of the material properties (the specific conductance and electrical permittivity of the two phases), the void fraction, and the flow regime. The specific conductance determines the conductive reactance, while the permittivity determines the capacitive reactance. For a given geometry of the electrodes, an appropriately normalized admittance is a function of the void fraction and the flow regime.

In addition to electrical impedance-based sensors, capacitance-based sensors have been widely used. A wire mesh sensor has been used to measure transient phase fraction distributions in a thin rectangular channel via permittivity (capacitance) measurements [26]. Two planes were embedded with 16 wires each and assembled perpendicular to each other. Measurements were taken at the points of wire intersection, giving 256 spatial points at a rate of 625 frames per second. Images of air bubbles in silicone oil were reconstructed using the measured data. A set of capacitance probes was used to measure the void fraction of HFC refrigerants in a horizontal tube at the macroscale [27]. The concave probes were placed opposite each other around the tube and three flow regimes were observed: slug, intermittent, and annular. A set of twelve capacitance probes was used to measure the void fraction of an oil-gas mixture in a 50 mm diameter pipe, as well as polyethylene particles in air for a 100 mm diameter pipe [28]. The electrodes were arranged in a circle around the pipe providing 66 independent capacitance measurements.

A mathematical model was developed to reconstruct an image of the distribution of the voids.

Two-phase flow regimes are typically described using qualitative categorization of flow visualizations. This involves subjectivity in their identification. In order to overcome this difficulty, Jones and Zuber [29] first employed quantitative means for flow regime determination. Using an X-ray source, they measured the temporal variation of the area-averaged void fraction in a rectangular channel with a cross-section of 10 cm × 1 cm and plotted a probability density function (PDF) of the void fraction. The significant differences in the PDF between various flow regimes suggested their use for flow regime determination. Later studies by Tutu [30] and Matsui [31] used void fraction distributions obtained with differential pressure transducers, while non-intrusive impedance void fraction meters were used by Mi *et al*. [23] as flow regime indicators. A comprehensive study by Costigan and Whalley [32] on flow regimes in vertical upflow used segmental impedance electrodes to determine the void fraction, combined with the PDF technique. Recently, bubble chord-length distributions obtained from conductivity probes were used as flow regime indicators by Julia *et al*. [33]. Caniere *et al.* [27] used the fuzzy c-means clustering algorithm for flow regime classification based on the signal, variance, and frequency measured from capacitance probes. The quantitative flow regime classification proposed by Mi *et al*. [23] is adapted in this study to identify the flow regimes.

While many experimental studies have been performed, few numerical simulations have been implemented to determine the response to electrical impedance-based sensors. An experimental and numerical study investigated two-phase flow in macroscale pipes using an impedance meter [3]. The electrodes consisted of a stainless steel electrode electrically insulated from the pipe, and the pipe itself, assembled in a streamwise configuration. Using the electro-quasi-static framework, the output voltage was solved via the Laplace equation and the impedance was evaluated for pipe diameters ranging from 25 mm to 100 mm. Experiments were performed under adiabatic conditions using co-current air and water in an upward tube with a diameter of 26 mm. Bubbly, spherical cap, stable and unstable slug, and semi-annular flows were observed. A stepwise correlation using separate linear fits corresponding to annular, intermittent, and bubble flows was developed to predict the relationship between the time-averaged void fraction and the normalized sensor output.

In summary, there are many electrical sensor-based methods for measuring void fraction [34]. These methods rely on the fluid permittivity (capacitance), resistance, or both (impedance). While many previous studies have included primarily capacitance measurements, the use of impedance based measurements removes limitations on feasible fluids for use in applications by taking advantage of the difference in the electrical conductivities of the fluids. Previous studies have been conducted at the macroscale; however, microscale studies are needed to determine the optimal sensor design for use in microchannel heat sinks. Numerical simulations that accurately capture the experimentally measured sensor output can be leveraged in the design process.

2.2 Effects of Non-Uniform Base Heating on Microchannel Flow Boiling

Many studies in the literature have investigated uniform base heating profiles applied to microchannel heat sinks, as reviewed, for example, by Tullius [35], Kandlikar [36], and Garimella and Harirchian [37]. These studies experimentally measured the onset of nucleate boiling [4], pressure drop [5,9], and heat transfer coefficients [5,6,38], and also developed models to predict the critical heat flux (CHF) [7,8]. In addition, flow regime maps have been developed under a variety of operating conditions [11,39]. While these studies have provided a thorough understanding of microchannel flow boiling under ideal heating conditions, realistic applications may impose highly non-uniform heat fluxes due to chip- and system-level variations [40]. In order to reliably predict the performance in actual applications, a better understanding of two-phase microchannel cooling under non-uniform heating conditions is needed, especially in terms of deviation in heat transfer performance and flow behavior compared to uniform heating conditions.

A discretized theoretical model for assessment of non-uniform heating in microchannels was developed by Koo *et al*. [41] using correlations for flow boiling heat transfer and pressure drop. The model was used to explore optimal geometric designs, but was limited in its ability to assess lateral flow instabilities across channels and for CHF prediction. A numerical simulation of the effect of header shape on flow maldistribution was performed by Cho *et al*. [42] for a microchannel heat sink. While an optimally designed header produced a low deviation in the mass flow distribution under uniform heating conditions, poor flow distribution was present in the case of a large locally applied thermal load. A numerical model developed by Sarangi *et al*. [43] predicted the pressure drop and thermal resistance of a uniformly heated microchannel, and location of boiling incipience. The models were also extended to include non-uniform heating conditions, which showed a large impact on the overall fluid dynamics and heat transfer of the system. Revellin and Thome [7] developed a one-dimensional theoretical model to predict CHF in microchannels under uniform heating conditions, which was further modified by Revellin *et al*. [44] to incorporate non-uniform axial heat fluxes.

Past experimental efforts have studied the effects of non-uniform microchannel heating on flow boiling instabilities [45], pressure drop, and maximum wall temperatures [46,47,48]. It was found that hotspots near the inlet created a large transverse temperature variation across the heat sink due to non-uniform fluid distribution. Maldistribution was caused by a local increase in two-phase pressure drop due to boiling, which diverted single-phase liquid to other locations; this effect was most pronounced for a hotspot at the inlet. Transient non-uniform heating situations have also been investigated [46,49]. Despite the numerous experimental studies conducted on microchannel heat sinks for uniform heating conditions, far fewer experiments utilizing non-uniform heating profiles have been conducted.

Prior experimental studies with non-uniform heating conditions have typically focused on single point hotspots. The effect of location and configuration of the hotspot as well as that of multiple hotspots on thermal performance has not been fully explored. In addition, a rigorous study of other heating profiles, especially superposed on a uniform background heat flux as would be realized in application, has not been reported. This thesis studies both local hotspots and increasingly non-uniform peak heating profiles across the heat sink, both in the flow direction and perpendicular to it, with respect to thermal performance and flow boiling phenomena. This work considers the effects of non-uniform heating on the local heat transfer coefficients, wall temperatures, heat fluxes, and boiling characteristics of a microchannel heat sink. Concentration of the heat input typically results in higher local heat transfer coefficients due to transition into the more efficient boiling regime at the expense of increased local wall temperatures. This work enables better assessment of existing heat transfer models for prediction of non-uniform heating profiles.

2.3 Effects of Non-Uniform Base Heating on the Critical Heat Flux

There have been many studies in the literature that have investigated the critical heat flux (CHF) in microchannels and have been summarized at length in [35,36,50,51]. Studies typically conduct experiments to investigate critical heat flux using either a single channel [52,53]or a parallel array of channels [54,55,56,57]. Although these studies have provided a better understanding of the parameters affecting the critical heat flux under uniform heating conditions, realistic applications typically impose highly non-uniform heat fluxes due to variation at both the chip and system levels [40]. A better understanding of how the location of hotspots affect the critical heat flux in two-phase microchannel cooling is needed to reliably predict the performance of heat sinks in actual applications.

Experiments using a single, circular channel were performed by Del Col and Bortolin [53] using three refrigerants. A non-uniform heat flux was imposed on the channel by using a hot water jacket to heat the test section, however, the entire flow length was supplied heat. The dryout quality and average critical heat flux were measured during annular flow. The data was compared to several models that were developed for uniformly heated microchannels [7,8,36,58,59] and were found to overpredict the critical heat flux.

Experimental investigations by Qu and Mudawar [54] and Chen and Garimella [55] were performed using a parallel array of microchannels. At near-CHF conditions, both studies reported that significant amounts of vapor in some channels were seen to reverse flow into the inlet manifold thus altering the inlet bulk fluid temperature. This vapor backflow negated any advantage of utilizing inlet subcooling and proved to have no effect on the critical heat flux. An abrupt decrease in the pressure drop was measured by Chen and Garimella [55] during CHF, and found that CHF is strongly dependent on the fluid properties, flow rate, and area of heat flux. Additionally, a CHF correlation developed for a single channel was used as a comparison to the measured data from Qu and Mudawar [54] but was shown to be a poor predictor for multiple-channel heat sinks.

A few studies have been performed to compare data from multiple CHF experiments to correlations found in the literature to determine which best predict CHF [59,60,61]. Zhang *et al*. [59] determined that the Hall-Mudawar correlation [62] best predicted CHF for subcooled water while the Shah correlation [63] best predicted CHF for saturated water. Although they developed their own correlation, Zhang *et al*. [59] clearly state it is limited to use for uniform heating situations. Revellin *et al*. [60] denoted that the most accuracte correlations for predicting CHF should be categorized based on the fluid used. For non-aqueous fluids, the theoretical model by Revellin and Thome [7] best matches extant data, while for water, the correlation by Zhang *et al*. [59] best matches. Additionally, other studies have proposed CHF models based on experimental data found in the literature [7,8,64,65]. Of these models, only those that were developed by Revellin and Thome [7] can be used with non-uniform heat fluxes; the model is never compared to non-uniform heating experimental data.

Although most studies involving microchannel heat sinks only look at uniform heating profiles, there are some that have looked at hotspot heating effects on flow boiling [66,67]. These investigations have shown that local hotspots cause a significant deviation of the local wall temperatures, local heat fluxes, and the total power dissipated as compared to a uniformly heated case. However, these studies lack experimental CHF data and cannot predict the effects of hotspot heating on the critical heat flux.

Critical heat flux values in non-uniformly heated macroscale tubes have been reported in the literature [68,69,70]. According to Yang *et al*. [68] the critical heat flux in an axial non-uniform heat flux distribution test section of inner diameter 5.46 mm could occur at single or multiple locations simultaneously and shift up or downstream depending on the inlet temperature andm ass flux. Olekhnovitch *et al*. [70] studied the effect of circumferentially non-uniform heating in 22 mm diameter round tubes. Significant bowing of the tubes was noted after running the experiments. Even though critical heat flux values for non-uniform heating cases at the macroscale are found in the literature, this phenomenon has not previously been tested in a microchannel heat sink.

This thesis studies several canonical hotspot heating cases to determine their effect on the critical heat flux in a microchannel heat sink. Local wall temperatures and heat fluxes are reported, and the location of the hotspot is determined to have a significant effect on CHF. This work gives a better understanding of how non-uniform heating profiles change the critical heat flux as compared to a uniform heating case.

CHAPTER 3. VOID FRACTION MEASUREMENT USING CROSSWISE ELECTRODES

The electrical impedance of a two-phase mixture is a function of the void fraction and phase distribution. The difference in the specific electrical conductance and permittivity of the two phases can be exploited to measure the electrical impedance to obtain the void fraction and flow regime characteristics of a mixture. An experimental investigation of the void fraction using an electrically impedance-based sensor is studied in this chapter for a variety of adiabatic air-water two-phase flow conditions in a microchannel. Flow regimes are identified quantitatively using the statistics of the signals acquired by the impedance void fraction sensor. The material in this chapter was presented at the *ASME Pacific Rim Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Systems* in 2011 and published in the proceedings [71]. It was later refined and published in the *International Journal of Multiphase Flow* [72]. The author would like to thank Sidharth Paranjape for designing and building the experimental facility, designing and building the impedance void fraction meter, collecting experimental data, collecting high-speed flow visualizations, and performing the calibration and flow regime identification analysis.

3.1 Experimental Method

3.1.1 Test Section

An experimental test cell to measure the void fraction of two-phase air-water flow was fabricated using clear transparent acrylic. The experimental facility was designed by Sidharth Paranjape. A photograph and drawing of the test cell is shown in Figure 3.1. A single flow channel with a 780 μm × 780 μm square cross-section and length of 50.8 mm is cut into the base plate. Two stainless steel 304 electrodes are embedded in the base plate so that the faces of the electrodes are flush-mounted to the side walls of the channel. The electrodes are located 25.4 mm (32.6 hydraulic diameters) from the inlet of the microchannel. The electrodes were designed to be identical to the width and height to the flow channel, *i.e.*, 780 μm. Inlet and outlet plenums are machined into the top cover plate to provide manifolds for water flow into the flow channel. The top cover plate is equipped with tube fittings to connect the test cell to the flow loop. Liquid water enters the flow channel from the inlet manifold and air is directly injected into the flow channel through a 0.3 mm diameter orifice at the bottom of the channel. The air inlet orifice is located 10 mm downstream from the inlet of the flow channel. The electrodes are connected to an electronic circuit via 14 gauge copper cables. Silver epoxy is used to minimize the contact resistance between the electrodes and copper cables.

A flow loop is constructed to provide air and water flow through the test cell and is shown in Figure 3.2. Deionized water is used as the liquid and a small amount of morpholine and ammonium-hydroxide (1 mg of each per liter of deionized water) is added in order to increase its electrical conductivity while maintaining a pH value of 7. The addition of these chemicals has a negligible impact on the flow regime through a change in surface tension as suggested by Mi *et al*. [23]. The specific conductivity of the water is maintained at 100 μS/cm. The water flow loop is equipped with a frequency-controlled water pump and a needle valve to control the water flow rate. The water flow rate is measured with a micro-turbine flow meter (McMillan Flo-106) with a range of 0 to 200 mL/min. Air flow is provided by a compressed air cylinder equipped with a pressure regulator and is controlled by a needle valve. The air flow rate through the test cell is measured via an air mass flow sensor (Omega FMA6704) with a range of 0 to 100 mL/min. The flow sensor also measures the temperature and pressure of the gas at the flow meter. The measured temperature and pressure are used to correct the mass flow rate from standard conditions since the flow sensor is factory-calibrated at standard temperature and pressure. Pressure is measured at the inlet and outlet of the channel. The local pressure at the measurement point in the channel is interpolated based on these two measurements. The actual volumetric flux of air is corrected for the interpolated pressure at the measurement location. The water storage tank is open to the atmosphere and serves as an air-water flow separator. Special care is taken to avoid flow instabilities from occurring due to the accumulation of air in various tube fittings in the exit section of the flow loop. In order to do this, flexible tubing (Saint-Gobain Tygon) is used to connect the exit of the test cell to the storage tank, which is located at a higher elevation than the test cell.

3.1.2 Impedance Void Fraction Meter

An auto-balancing bridge method is implemented in a custom-built unit for measurement of the electrical impedance of the two-phase mixture in the test cell. The instrument was designed by Sidharth Paranjape. The details of auto-balancing bridge methods can be found in Tumanski [73]. The signal processing scheme is shown in Figure 3.3. The test cell is excited with an alternating sine wave voltage signal with a peak-to-peak voltage difference of 3 V. The exciter signal is set to a frequency of 20 kHz. A current-to-voltage amplifier is used to measure the resulting current. The voltage measured across the reference resistor of the amplifier circuit serves as a measure of the current flowing through the test cell. This signal is referred to as the modulated signal, while the exciter signal is taken as the carrier wave. Both of these voltage signals are logged to a high-speed data acquisition system (National Instruments NI 6259-USB) and are sampled at a rate of 500 kHz. The data acquisition system has a 16-bit quantization for analog to digital conversion in the voltage range of -5 V to +5 V. The signals are then processed numerically using a MATLAB program developed in-house. The acquired signal is synchronously demodulated using the excitation signal and a 90° phase-shifted excitation signal in order to calculate the real and imaginary parts of the impedance across the channel. A low-pass Butterworth filter with a cut-off frequency of 10 kHz is used to filter out the excitation signal. The filtered signal is proportional to the electrical impedance of the two-phase mixture between the electrodes.

3.2 <u>Image Analysis and Data Reduction</u>

3.2.1 Flow Visualization and Image Processing

A Photron Fastcam-Ultima APX high-speed digital video camera combined with a Keyence VH-Z50L lens at 100X magnification is used for flow visualization. The videos are acquired at a frame rate of 24,000 frames per second with a shutter speed of

120,000 Hz. An illumination source (Henke-Sass Wolf) is used to illuminate the microchannels from below for visualization. This combination provides a spatial resolution of 8 μm per pixel. The digital videos are acquired for 4 s for each flow condition. The stored images are further processed in order to calculate the void fraction. The image processing is performed in the sequence described below. The complete MATLAB script can be found in Appendix A.

The video frames are taken through a number of image processing steps to determine the air and water regions. First, each frame is rotated and cropped to the area of interest. This is a square-shaped interrogation window that has the same width as that of the flow channel. Second, the background is subtracted and the gray-scale image is passed through a threshold to obtain the negative of the image and increase the contrast. Third, the edges of the air regions are detected using the Canny algorithm implemented in MATLAB [74]. Fourth, the interior boundaries due to light reflection are then removed via algorithms implemented in MATLAB. The details of the algorithm are explained in Soille [75]. Fifth, the objects are passed through a convex hull algorithm to remove cusps and concave boundaries. Sixth, the volumes and cross-sectional areas of the bubbles are calculated and hence the volumetric void fraction and cross-sectional area-averaged void fractions, respectively. In order to calculate the bubble volume from a two-dimensional image obtained via flow visualization, the bubbles are assumed to be axisymmetric about their major axes. An approximate uncertainty analysis was performed for the calculation of bubble volumes for the simple geometries of spherical and cylindrical bubbles. The maximum error was found to be 8% of the measured value. Lastly, steps 1 through 6 are repeated for each frame in the video to obtain a time series of the void fraction. In addition, the time-averaged volume- and area-averaged void fractions are calculated for each flow condition.

The morphological operations performed to extract the boundaries of the bubbles from the original image are shown in Figure 3.4. The time-averaged void fractions calculated from processing the images in the videos are used as reference measurements to calibrate the impedance void fraction meter.

3.2.2 Uncertainty Analysis

The uncertainties in the measurements of the steady-state values of the gas and liquid flow rates stem from a combination of uncertainty in the measurement by the flow meters and the inherent physical fluctuations in the flow conditions. The measurement uncertainties for the instruments used in the experiment are shown in Table 3.1. The last column denotes the maximum standard deviation as a percentage of the measured value that was observed in the dataset that consists of 71 flow conditions.

For each flow condition, the quantities were acquired at 500 Hz for 10 seconds to obtain time-averaged values after reaching steady state. The resulting maximum uncertainties in the measurement of gas and liquid flow rates are found to be 2.5% and 1% of measured values, respectively.

3.3 Results and Discussion

Void fraction measurements were performed under 71 different flow conditions. Data was collected by Sidharth Paranjape. Each condition was characterized by the velocity inlet boundary conditions: volumetric flux of the gas, $\langle j_g \rangle$, and volumetric flux of the liquid, $\langle j_f \rangle$. The range of flow conditions covered are 0.13 m/s < $\langle j_g \rangle$ < 2.65 m/s and 0.8 m/s < $\langle j_f \rangle$ < 5.1 m/s. The test matrix is shown in Figure 3.5 using coordinates of volumetric flux of the gas and liquid flow.

3.3.1 Calibration of the Impedance Void Fraction Meter

High-speed visualization revealed the flow regimes observed under the set of test conditions. The images obtained using the high-speed video camera in various flow regimes are shown in Figure 3.6. The images are presented as acquired by the camera without any morphological transformations. The void fraction reported for each image is the time-averaged value of the volume-averaged void fraction calculated using the image processing algorithm. The time series of the volume-averaged void fractions corresponding to each flow conditions is also shown in Figure 3.6.

The measurement method utilized determines the current passing through the test cell for a given potential difference at a known excitation frequency. The measured current is proportional to the admittance, or the inverse of the impedance of the two-phase mixture in the test cell. In order to make the measurement independent of the material properties, the measured admittance is normalized as

$$G^* = \frac{G_m - G_1}{G_0 - G_1}, \tag{3.1}$$

where G_m is the instantaneous two-phase mixture admittance, G_0 is the admittance for a void fraction of zero (for single-phase liquid) and G_1 is the admittance for a void fraction of unity (for single-phase gas). The liquid fraction of the mixture is a monotonically increasing function of normalized admittance, G^*, and the void fraction is proportional to

$\alpha_{imp} = 1 - G^*$. For finely dispersed bubbly flow (a void fraction less than 10%), the functional relationship can be obtained by the effective conductivity of a medium impregnated with uniformly distributed non-conducting spheres. The expression for the effective conductivity given by Maxwell [76] to a first-order approximation is

$$G^* = 1 - \frac{3\alpha}{2+\alpha}, \tag{3.2}$$

where α is the void fraction of the dispersed phase. This model is applicable to the bubbly flow regime for void fractions less than 0.2. For void fractions above this limit, the sensor must be calibrated due to the statistical nature of the distribution of voids, where no closed-form analytical solution is available. The impedance void fraction meter is calibrated in a time-averaged sense. The time-averaged value of the impedance void fraction meter reading, $\langle\alpha_{imp}\rangle_t$, is compared with the time-averaged void fraction, $\langle\alpha\rangle_{V,t,im}$, obtained by flow visualization.

The calibration curve of the impedance void fraction meter against the void fraction obtained by image processing for various flow regimes is shown in Figure 3.7. The data show that the instrument has a nearly linear response. The data are also compared with Equation (3.2) for bubbly flow conditions. It can be observed that the data match the predicted values from this equation closely for void fractions less than 0.15. A third-order polynomial curve is fit to the data to obtain a calibration curve. The calibration curve is given by

$$\alpha_{cal} = -1.18\alpha_{imp}^3 + 1.57\alpha_{imp}^2 + 0.61\alpha_{imp}. \tag{3.3}$$

In order to assess the accuracy of the measurement, the mean square deviation is calculated as

$$E_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\alpha_{cal} - \langle\alpha\rangle_{V,t,im}\right)^2}, \tag{3.4}$$

where α_{cal} and $\langle\alpha\rangle_{V,t,im}$ are void fractions obtained from the calibration curve and by image processing, respectively. The mean square deviation is 0.023.

In order to validate the measurement of void fraction by the impedance void fraction meter, the void fraction measured by the sensor is plotted against the ratio of gas volumetric flux to total volumetric flux, β, which is defined as

$$\beta = \frac{\langle j_g\rangle}{\langle j_g\rangle + \langle j_f\rangle}. \tag{3.5}$$

In the case of the homogenous equilibrium model, or under the assumptions of uniform distribution phases in flow cross-section and equal velocities, the void fraction is given by

$$\alpha = \beta. \tag{3.6}$$

In view of the drift flux model, the relation between void fraction and volumetric fluxes is given by Zuber and Findlay [77],

$$\langle\alpha\rangle = \frac{\langle j_g\rangle}{C_0\left(\langle j_g\rangle + \langle j_f\rangle\right) + \langle\langle v_{gj}\rangle\rangle}. \tag{3.7}$$

In Equation (3.7), C_0 is the distribution parameter, while $\langle\langle v_{gj}\rangle\rangle$ is the void-weighted drift velocity. These two parameters are specified by empirical correlations. The recommended value for the distribution parameter is 1.2 as suggested by Armand [78], Ali *et al*. [79], and Mishima and Hibiki [80]. For horizontal flow, the drift velocity is close to zero. A comparison of the measured void fraction against the homogenous flow and drift-flux models is shown in Figure 3.8. The agreement between the predictions from both models and the data is remarkable considering the lack of established values for parameters in the drift flux model for the case of microchannel flow. Since the void fraction is a function of flow boundary conditions, volumetric fluxes of the gas and liquid, the measured void fraction contours are plotted on gas and liquid volumetric flux coordinates as shown in Figure 3.9. Contour maps are helpful in developing void fraction correlations for microchannel two-phase flows.

3.3.2 Flow Regime Identification

The approach originally developed by Jones and Zuber [29], which utilizes the probability density function (PDF) of the void fraction fluctuations as flow regime indicators, is employed for flow regime identification. Physically, the PDF denotes the contribution of different kinds of bubbles to the time-averaged void fraction for a given flow condition. The normalized time series signal obtained by the impedance void fraction meter, $G^*(t)$, is used for this purpose. The PDF of $G^*(t)$ denoted by $f_{G^*}(G^*)$ is calculated using the kernel smoothing density estimation method described by Bowman and Azzalini [81]. A normal kernel is used as the smoothing function. The PDF $f_{G^*}(G^*)$ is evaluated at 200 discrete points in the domain of $G^* \in [0,1]$. Thus, each flow condition is represented by a 200-dimensional vector. The problem of identifying flow regimes is equivalent to identifying clusters of vectors in 200-dimensional vector space. The clusters of vectors are found by minimizing the distance between the vectors representing flow conditions and the weight vectors corresponding to a flow regime. After minimization, the weight vector positions align with the centroid of the clusters. Thus, the weight vectors that denote the positions of the cluster centroids are characteristic of the flow regime. This optimization problem is solved by the *Kohonen Self-Organizing Map* algorithm for pattern recognition implemented in the Neural Network Toolbox of MATLAB based on the method developed by Kohonen [82]. Physical interpretation of the recognized patterns is accomplished by comparing them with the flow regimes observed using the high-speed camera.

Examples of the impedance void fraction meter signals and corresponding PDFs obtained for various air-water flow regimes are shown in Figure 3.10 through Figure 3.14. The flow regimes, their qualitative description, and characteristics of the corresponding impedance void fraction meter signals are described below. It is noted that stable annular flow could not be achieved with the current experimental setup.

- *Bubbly Flow*: Bubbly flow is characterized by spherical or ellipsoidal bubbles dispersed in the continuous phase. The major diameters of these bubbles are smaller than the width of the channel. The PDF $f_{G^*}(G^*)$ shows a relatively small width with a peak at a higher admittance (see Figure 3.10).
- *Cap-Bubbly Flow*: As the bubble size increases, it is confined by the channel walls. It is distorted and forms a cap-shaped bubble with a round nose at its downstream end. The PDF is characterized by two distinct peaks located close to each other (see Figure 3.11). The peak corresponding to higher G^* represents the liquid regions between the bubbles, while the peak corresponding to lower G^* represents the cap bubbles.
- *Slug Flow*: Long bullet-shaped bubbles are separated by liquid or small spherical bubbles. The PDF shows two distinct peaks, with one located at low G^* corresponding to slug bubbles and the other located at high G^*

corresponding to the continuous liquid phase between the slug bubbles (see Figure 3.12).

- *Churn-Turbulent Flow*: Due to turbulent agitation at higher flow rates, churn-turbulent flow exhibits interacting slug bubbles with a distorted shape. This leads to a wider spread in the PDF, where peaks corresponding to slug bubbles and liquid gaps between them are merged. It should be noted here that the existence of a churn-turbulent regime does not imply a higher void fraction than that in the slug flow regime in a time-averaged sense (see Figure 3.13).
- *Long Slug Flow*: This regime is characterized by the occurrence of long stable slugs such that it appears to have a structure similar to annular flow in a local or short-time-averaged sense. The PDF shows a high peak at a low G^* (see Figure 3.14). Annular flow was not observed in the current dataset.

Using the quantitative method of flow regime classification described above, the dataset was categorized into five regimes. The result of this classification is shown in Figure 3.15. The flow conditions are presented on coordinates of volumetric flux of the gas and liquid phases. The contours of the time-averaged void fraction are superimposed on the flow regime map. This shows the relationship between the flow regime boundaries and the void fraction. This map could be used for the development of theoretical flow regime transition criteria in microchannel two-phase flow.

3.4 Conclusions

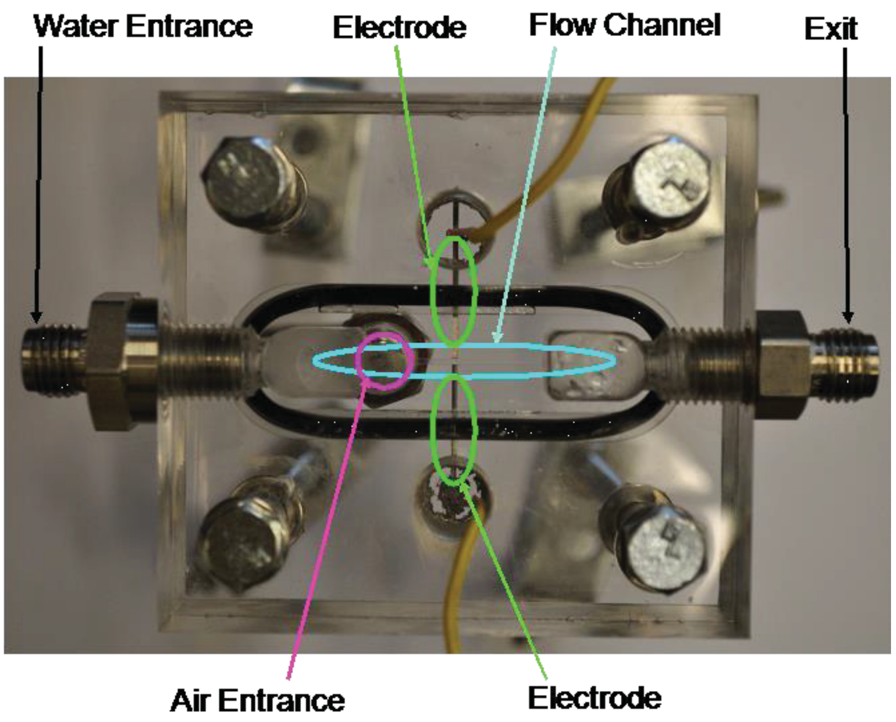
The void fraction of air-water two-phase flow is measured in a microchannel with a square cross-section of 780 μm × 780 μm using a custom-designed impedance void fraction meter. The impedance void fraction meter is calibrated against the time-averaged void fraction determined from flow visualizations using a high-speed video camera. The calculated time-averaged void fraction shows reasonable agreement with those predicted by the homogeneous equilibrium and drift-flux models. However, a conclusive statement in favor of a particular model cannot be made since the model parameters (the distribution parameter and the drift velocity for the drift-flux model) are not available for microchannel flows.

The probability density function (PDF) of the time series signal obtained by the impedance meter is utilized for quantitative characterization of two-phase flow regimes. The flow regimes are identified using a Kohonen Self-Organizing Map. This study shows that the impedance void fraction meter that was designed can be used for microchannel two-phase flows for the measurement of void fraction and the identification of flow regimes. The void fraction and flow regime data obtained by the impedance void fraction meter may be used for developing and benchmarking theoretical flow regime transition criteria for microchannel two-phase flows.

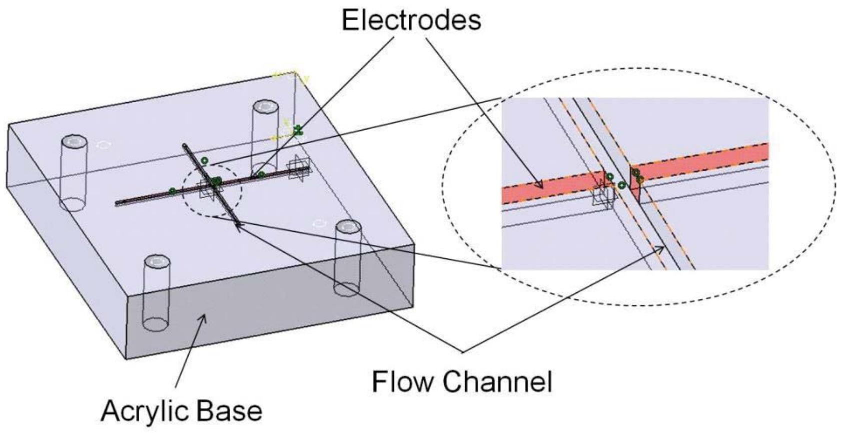
Further studies are discussed in later chapters over developing flow regime maps for additional flow channel geometries. The measurement technique developed here can be used to study non-adiabatic and boiling flows with a similar geometry of the electrodes along with the same electronic circuit, as long as the changes in electrical properties of the fluid with temperature are taken into account.

Table 3.1. Measurement uncertainties as a percentage of measured value.

Instrument	Reported measurement accuracy (%)	Maximum standard deviation (%)
Liquid flow (mL/min)	0.2	0.74
Gas flow (mL/min)	0.1	3.2
Pressure (kPa)	0.2	2
Temperature (K)	0.1	0.05



(a)



(b)

Figure 3.1. Impedance meter test cell. (a) Top view of the test cell. (b) Base plate with flow channel and electrodes.

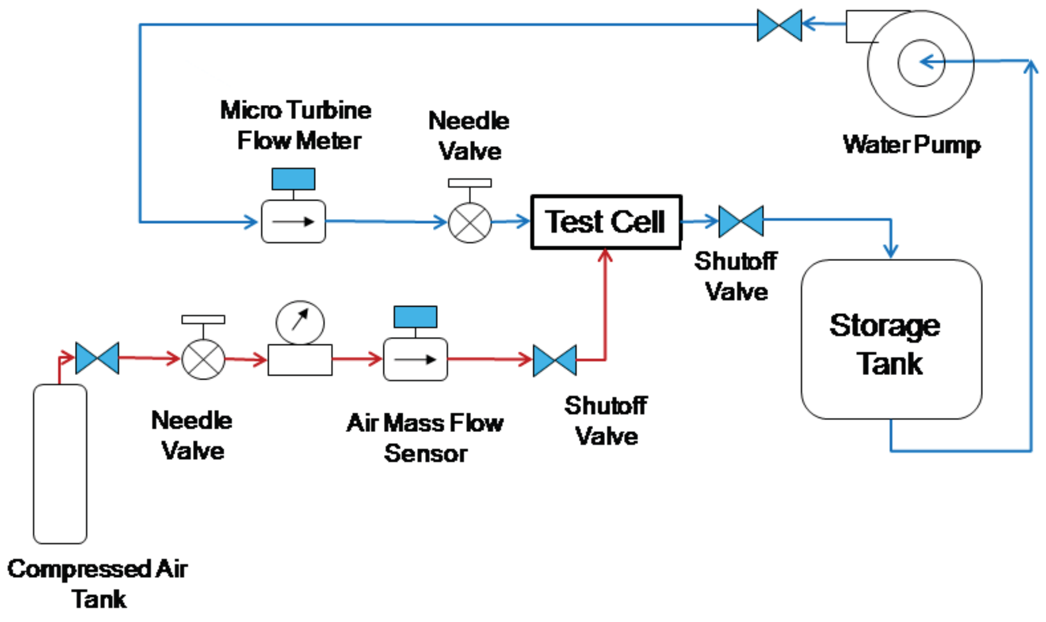
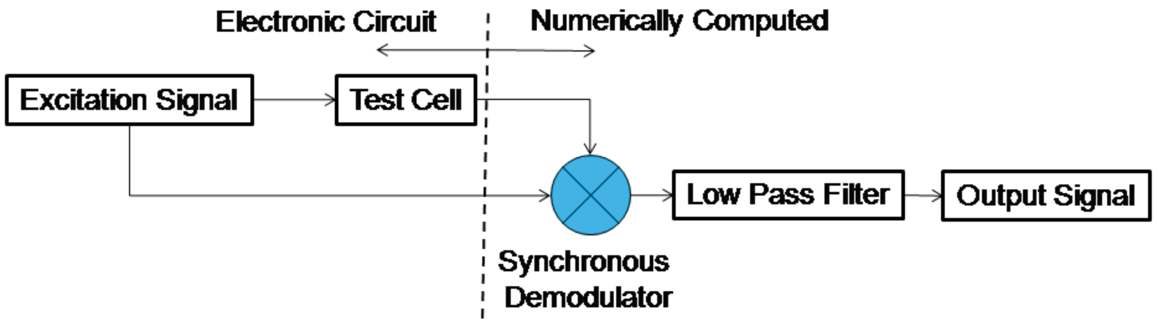
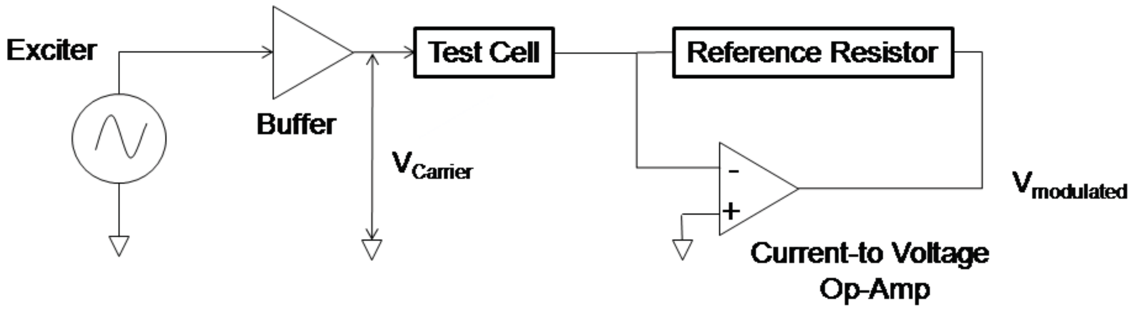


Figure 3.2. Air-water two-phase flow loop.

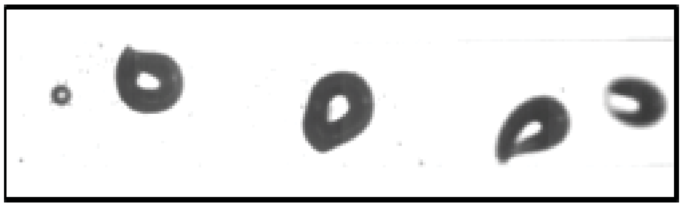


(a)

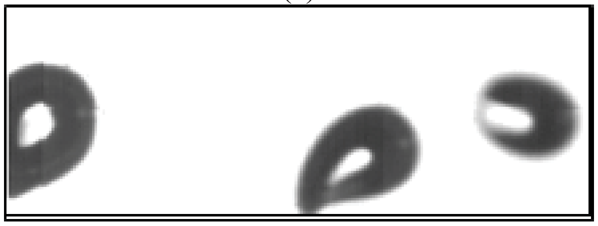


(b)

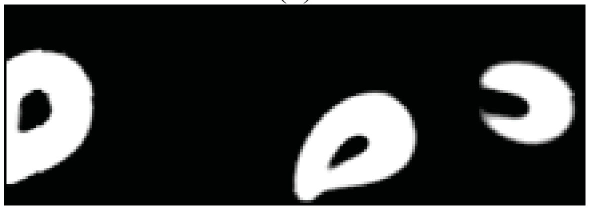
Figure 3.3. Impedance meter circuit. (a) Signal processing scheme. (b) Basic electronic circuit.



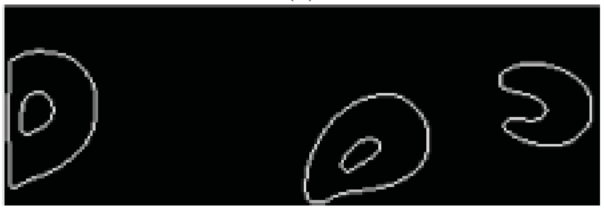
(a)



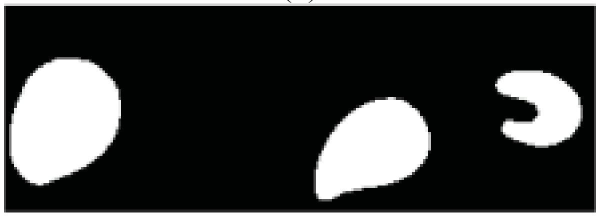
(b)



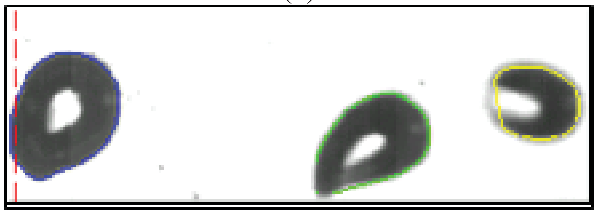
(c)



(d)



(e)



(f)

Figure 3.4. Image processing steps. (a) Original image, top view. (b) Rotated and cropped image for interrogation window. (c) Background subtracted and threshold adjusted image. (d) Edge detection. (e) Interior boundaries removed. (f) Edges after finding the convex hull, superimposed on the original image.

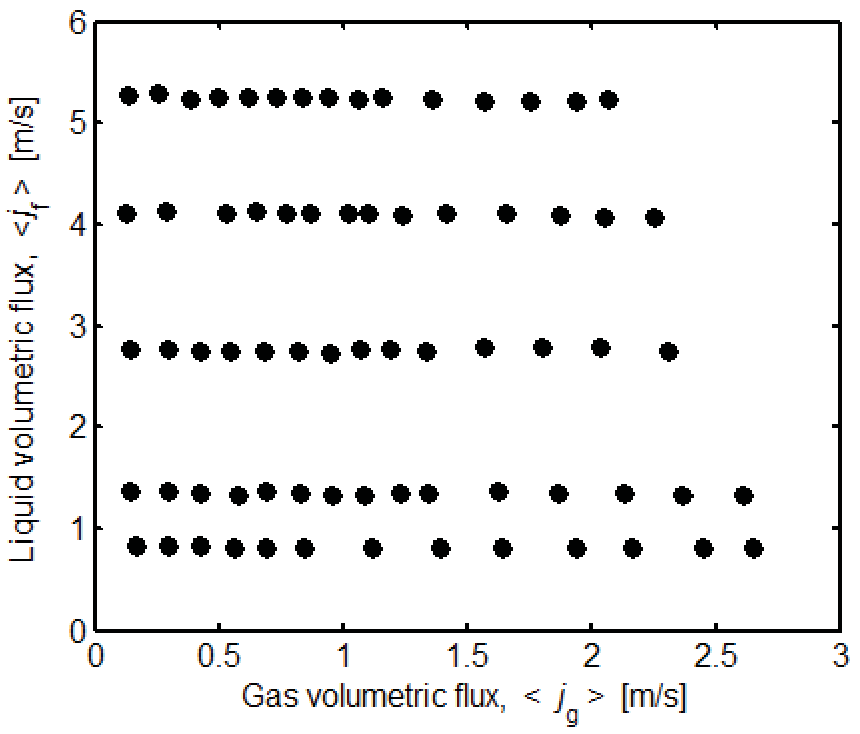
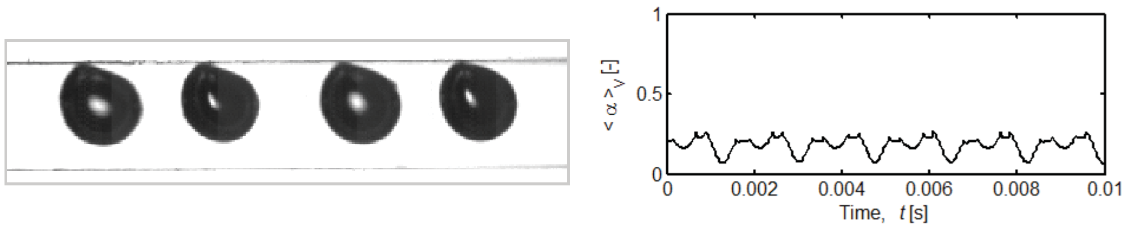
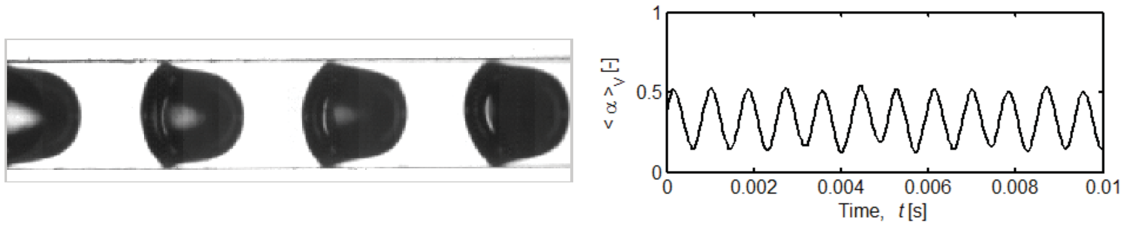


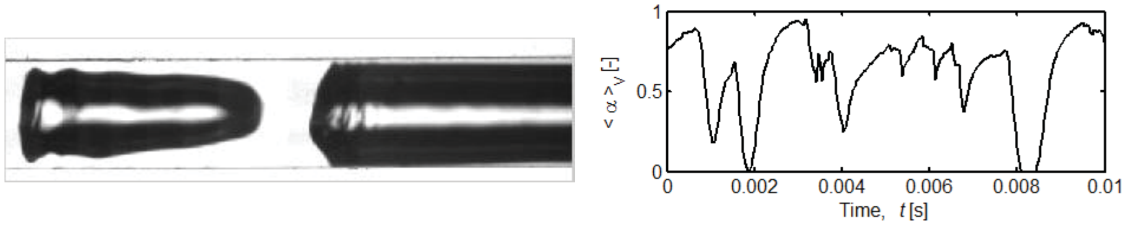
Figure 3.5. Test matrix.



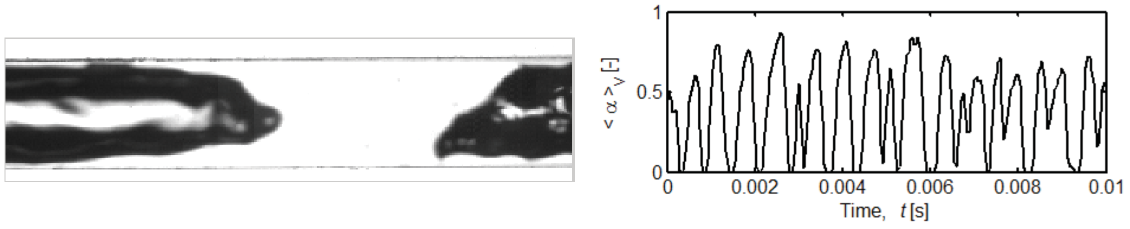
(a)



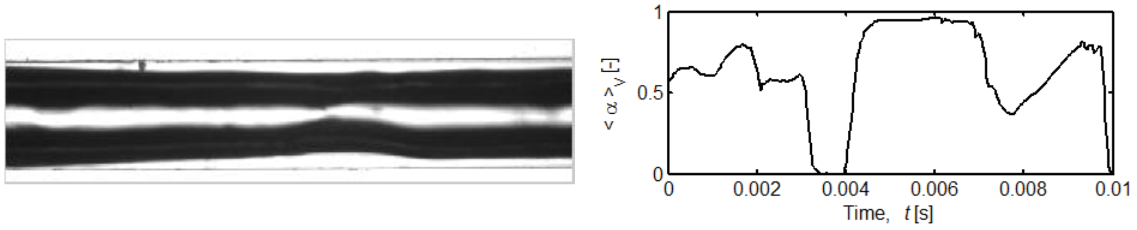
(b)



(c)



(d)



(e)

Figure 3.6. Flow visualization and void fraction measured by image processing. Flow direction is from left to right. (a) Bubbly, $\langle j_g \rangle$ = 0.29 m/s, $\langle j_f \rangle$ = 0.83 m/s, $\langle \alpha \rangle_{V,t}$ = 0.20. (b) Cap bubbly, $\langle j_g \rangle$ = 0.56 m/s, $\langle j_f \rangle$ = 0.83 m/s, $\langle \alpha \rangle_{V,t}$ = 0.37. (c) Slug, $\langle j_g \rangle$ = 1.39 m/s, $\langle j_f \rangle$ = 0.83 m/s, $\langle \alpha \rangle_{V,t}$ = 0.58. (d) Churn-turbulent, $\langle j_g \rangle$ = 2.26 m/s, $\langle j_f \rangle$ = 4.08 m/s, $\langle \alpha \rangle_{V,t}$ = 0.36. (e) Long slug, $\langle j_g \rangle$ = 2.65 m/s, $\langle j_f \rangle$ = 0.82 m/s, $\langle \alpha \rangle_{V,t}$ = 0.65.

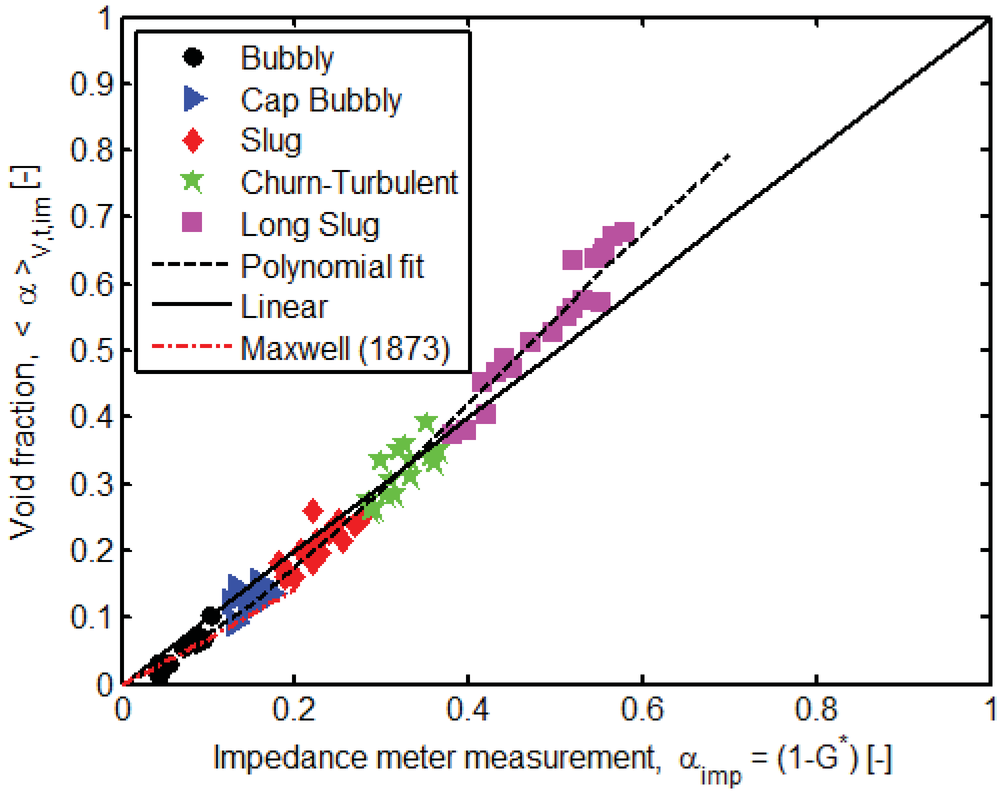


Figure 3.7. Impedance meter calibration.

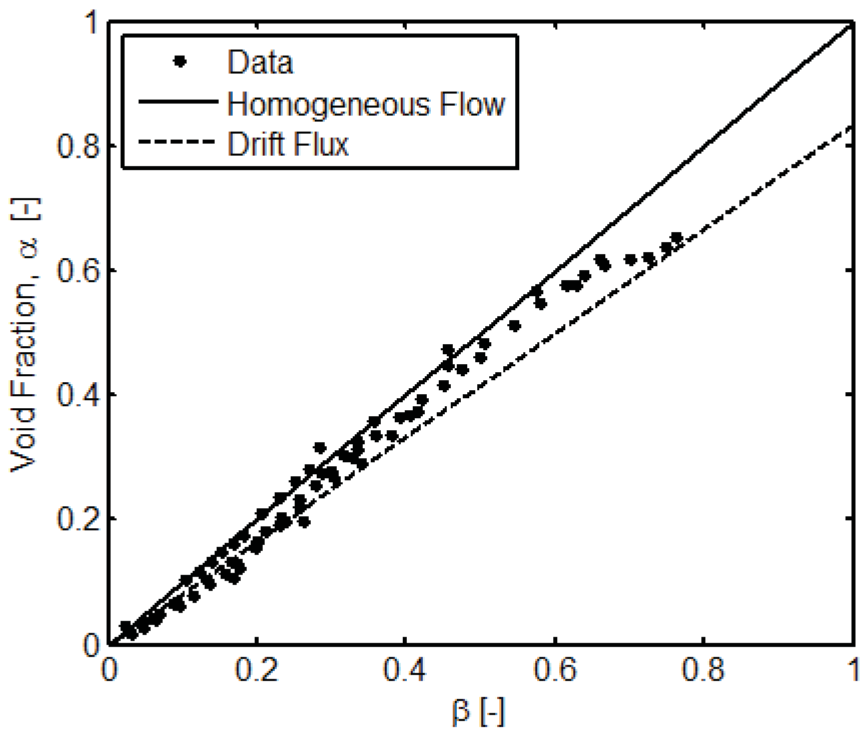


Figure 3.8. Comparison of the measured void fraction with the homogeneous equilibrium and drift-flux models.

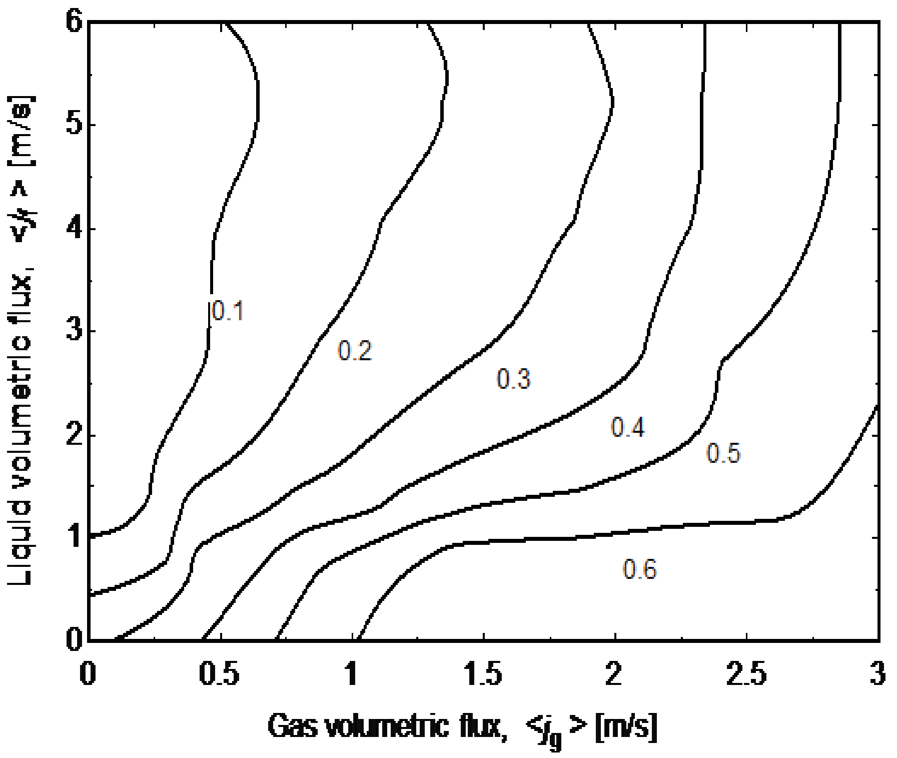


Figure 3.9. Contours of the void fraction.

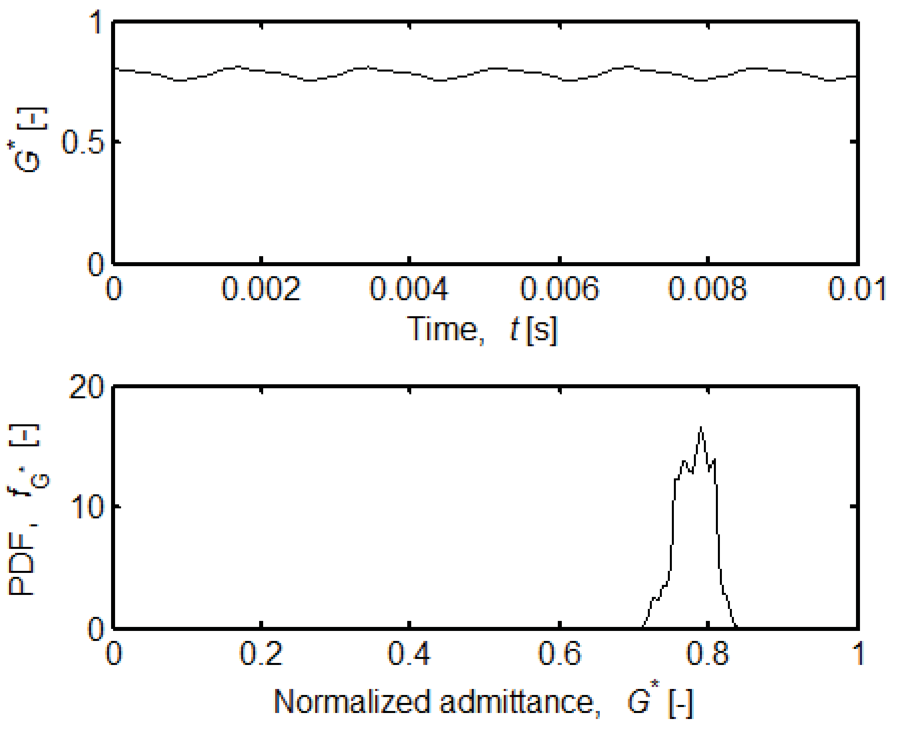


Figure 3.10. Impedance meter signal and its PDF for bubbly flow, $\langle j_g \rangle$ = 0.29 m/s, $\langle j_f \rangle$ = 0.83 m/s, $\langle \alpha \rangle_{V,t}$ = 0.20.

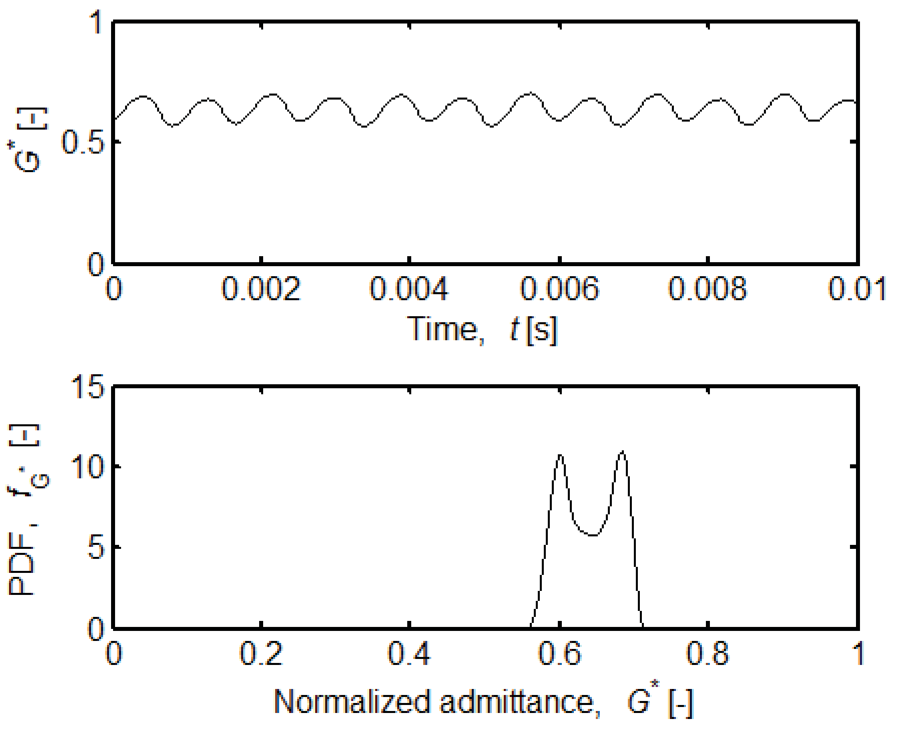


Figure 3.11. Impedance meter signal and its PDF for cap-bubbly flow, $\langle j_g \rangle$ = 0.56 m/s, $\langle j_f \rangle$ = 0.83 m/s, $\langle \alpha \rangle_{V,t}$ = 0.37.

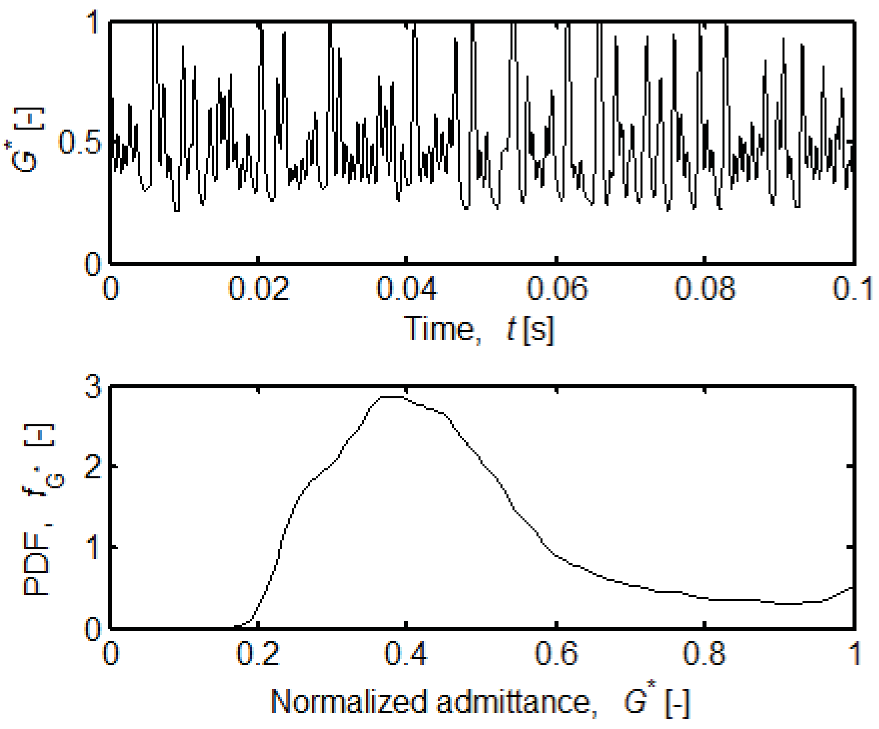


Figure 3.12. Impedance meter signal and its PDF for slug flow, $\langle j_g \rangle$ = 1.39 m/s, $\langle j_f \rangle$ = 0.83 m/s, $\langle \alpha \rangle_{V,t}$ = 0.58.

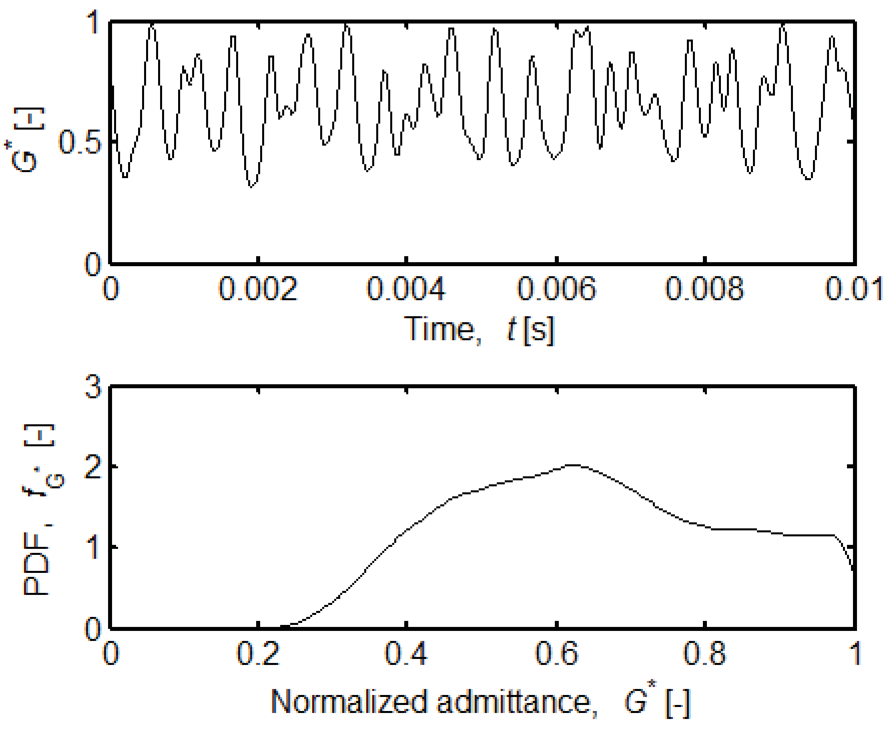


Figure 3.13. Impedance meter signal and its PDF for churn-turbulent flow, $\langle j_g \rangle$ = 2.26 m/s, $\langle j_f \rangle$ = 4.08 m/s, $\langle \alpha \rangle_{V,t}$ = 0.36.

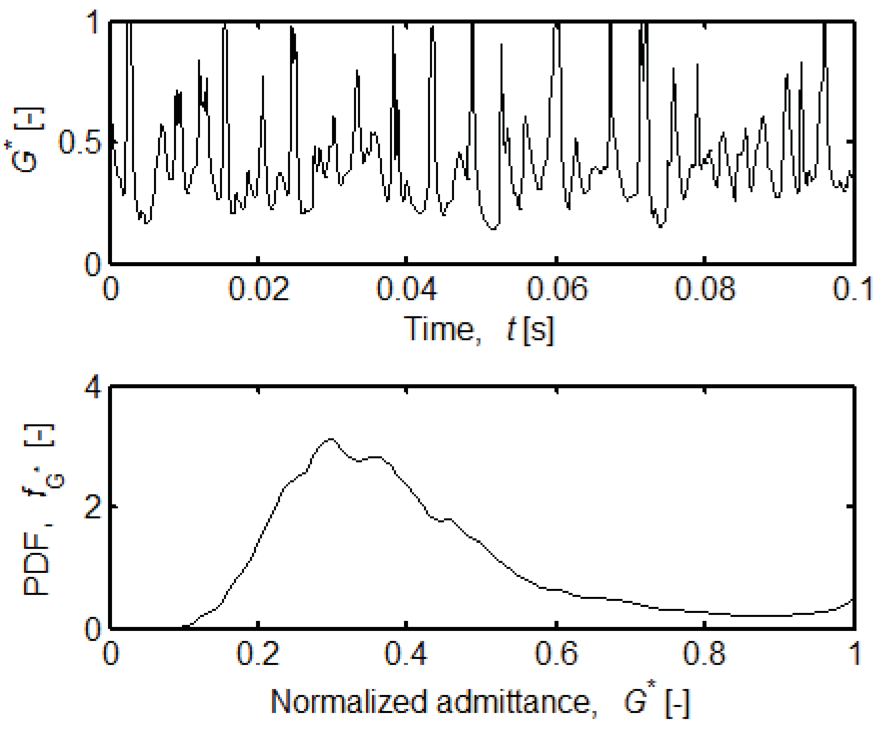


Figure 3.14. Impedance meter signal and its PDF for long slug flow, $\langle j_g \rangle$ = 2.65 m/s, $\langle j_f \rangle$ = 0.82 m/s, $\langle \alpha \rangle_{V,t}$ = 0.65.

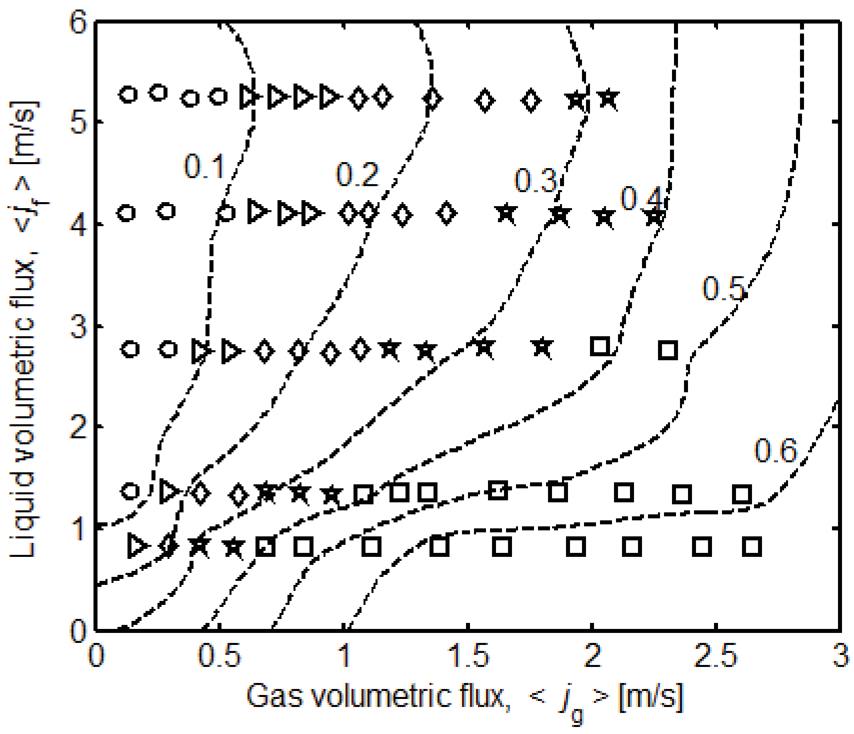


Figure 3.15. Flow regime map obtained using the impedance void fraction meter signals. ○: Bubbly, ▷: Cap bubbly, ◊: Slug, ☆: Churn-turbulent, □: Long slug, Level lines: Void fraction contours.

CHAPTER 4. NUMERICAL SIMULATIONS OF CROSSWISE ELECTRODE EXPERIMENTS

The void fraction in microchannel two-phase flow can be measured using an electrical impedance-based void fraction sensor. The electrical impedance of the two-phase mixture is a function of the void fraction and the flow topology due to the difference in the electrical properties of the two fluids. Due to the complex geometry of the phase interfaces under various flow regimes, this relation cannot simply be expressed analytically. Hence, the response of a miniature impedance-based void fraction sensor using air-water two-phase flow under adiabatic conditions was numerically investigated by solving the Laplace equation for the electrical potential. This chapter studies a range of flow regimes via two-dimensional (2D) and three-dimensional (3D) simulations using Fluent [83] and MATLAB [74]. The numerical results are then compared to experimental results. Portions of the material in this chapter were published in *Measurement Science and Technology* [84].

4.1 Modeling and Simulation

4.1.1 Image Processing

In order to simulate realistic two-phase flow morphologies that occur in a microchannel, high-speed videos of air-water flow through a microchannel were recorded. A Photron Fastcam-Ultima APX high-speed digital video camera combined with a

Keyence VH-Z50L lens at 100X magnification was used for flow visualization, as outlined in Paranjape *et al*. [72]. Videos were recorded at 24,000 frames per second with a shutter speed of 120,000 Hz. The videos recorded are used to determine the air and water domain to be used in the numerical simulations.

The movie frames are taken through a number of image processing steps to automatically determine the air and water regions. The image processing steps are the same as those presented in the previous chapter and can be seen in Figure 3.4.First, each frame is rotated and cropped to the area of interest. In the 3D Fluent simulations, this area was limited to a square-shaped window (cube-shaped domain) with the same width as the flow channel. For the 3D MATLAB simulations, this area was a rectangular-shaped window with the length three times the width of the flow channel. This was done in order to capture the additional effects of objects outside the crosswise electrodes on the electrical field. Next, the background was subtracted and the gray-scale image was passed through a threshold filter to obtain the negative of the image and increase the contrast. Then, the edges of the air regions were detected using the Canny algorithm implemented in MATLAB [74]. The interior boundaries due to light reflection were then removed, and then the flow objects were passed through a convex hull algorithm.

After the image processing steps are completed, the coordinates of the air-water boundaries are known. This information was used to generate 3D meshes in Gambit [85] and MATLAB. The Gambit meshes (for the use of Fluent as the solver) were generated using two methods, depending on the flow regime. For bubbly and slug flow, the air bubbles were assumed to have a perfectly circular cross section. Circles were plotted to represent the interface at each streamwise pixel location in the image; a skin was formed around the series of circles to create the air volume in the channel, as shown in Figure 4.1. For churn flow, the air region was assumed to fill the majority of the channel, so the air bubbles were modeled to have a square cross section. The same volume formation method was used. Once the air and water regions were defined, a tetrahedral volume mesh was created. To generate the meshes in MATLAB, first, a uniform grid of cube cells was created for the domain. Based on the centroid of each cell and the determined coordinates of the bubble boundaries, each cell was identified as belonging to either the air or the water region. Once again, bubbly and slug flows were assumed to have a circular cross section and churn flows were assumed to have a square cross section.

4.1.2 Numerical Methods

Fluent [83] was used for 2D and 3D simulations. A MATLAB [74] code was written for 3D time-averaged simulations, and can be found in Appendix B. This section describes the analysis used in both Fluent and MATLAB for the simulations. A form of the Laplace equation for electric potential is used to describe the domain.

$$\nabla \cdot \left(\frac{1}{\rho}\nabla\phi\right) = 0$$
$$\nabla \cdot (\varepsilon_0 k \nabla\phi) = 0. \tag{4.1}$$

Material properties and geometries are substituted into Ohm's law and the equation for charge to obtain the current and charge as a function of voltage. In this form, a heat transfer analogy,

$$\nabla \cdot J = \nabla \cdot \left(\frac{1}{\rho}\nabla \mathrm{V}\right), \nabla \cdot D = \nabla \cdot (\varepsilon_0 k \nabla \mathrm{V}) \sim \nabla \cdot q^{"} = \nabla \cdot (k\nabla T) \tag{4.2}$$

can be used to solve for the current and charge. By this analogy, Fluent can be used to solve the problem in this form via the energy equation.

To find the impedance in the channel, a Dirichlet voltage boundary condition was applied at the electrodes to provide a voltage difference of 4 V. All other boundaries used a Neumann boundary condition with the flux equal to zero. After finding the current and charge fluxes at the surface of the electrodes, the resistance, R, and capacitance, C, can be calculated respectively. These can then be used to find the magnitude of the impedance, $|Z|$. The impedance is calculated as if it were connected in parallel in a circuit, as

$$\begin{aligned} \Re &= \frac{R}{1+\omega^2 C^2 R^2} \\ X &= \frac{\omega C R^2}{1+\omega^2 C^2 R^2} \\ |Z| &= \sqrt{\Re^2 + X^2}. \end{aligned} \tag{4.3}$$

The impedances calculated are normalized with respect to a channel filled completely with water and air. Since the void fraction is positively correlated with one minus the admittance, $G = 1/|Z|$, it is chosen for presenting results. The normalization of G is

$$1 - G^* = 1 - \frac{G_1 - G}{G_1 - G_0} = \frac{G - G_0}{G_1 - G_0}, \tag{4.4}$$

where G_0 is the admittance for a channel filled only with water and G_1 is the admittance for a channel filled only with air. The final value, $1 - G^*$, is used for correlating with the void fraction.

The Fluent finite volume solver is used for the 2D and 3D meshes created in Gambit. The simulation was run by solving the energy equation. The thermal material properties in Fluent were changed to be the electrical properties of air and water and the analogy was used to convert the heat transfer parameters into their appropriate electrical analogs.

For the MATLAB solver written in house, the finite volume method is used to discretize the above equations for use in a uniform 3D mesh. Kirchhoff's current law is used to balance the currents between a cell and its neighbors as

$$\begin{aligned} a_{i,j,k}\phi_{i,j,k} = b_{i,j,k}\phi_{i+1,j,k} + c_{i,j,k}\phi_{i-1,j,k} + d_{i,j,k}\phi_{i,j+1,k} + e_{i,j,k}\phi_{i,j-1,k} \\ + f_{i,j,k}\phi_{i,j,k+1} + g_{i,j,k}\phi_{i,j,k-1} + h_{i,j,k}, \end{aligned} \tag{4.5}$$

where $a_{i,j,k}$ is the coefficient of the ith, jth, kth point, $b_{i,j,k}$ is the coefficient of the i+1, jth, kth point, etc., $h_{i,j,k}$ is the source term, and ϕ is the voltage (or charge) for each cell. The coefficients a through g are inverse resistance (or capacitance). For an interior point in the domain, this equation can be written and rearranged as

$$\begin{aligned} -k_e\Delta y\Delta z\frac{\phi_r - \phi_p}{\Delta x} + k_e\Delta y\Delta z\frac{\phi_p - \phi_l}{\Delta x} - k_e\Delta x\Delta z\frac{\phi_t - \phi_p}{\Delta y} + k_e\Delta x\Delta z\frac{\phi_p - \phi_m}{\Delta y} \\ - k_e\Delta x\Delta y\frac{\phi_f - \phi_p}{\Delta z} + k_e\Delta x\Delta y\frac{\phi_p - \phi_b}{\Delta z} = 0, \end{aligned} \tag{4.6}$$

where k_e is the effective electrical conductivity (or electrical permittivity) and Δx, Δy, and Δz are the dimensions of each cell.

The system of equations for the 3D domain is solved using a plane-by-plane Tri-Diagonal Matrix Algorithm (TDMA) method. The TDMA is a form of Gaussian elimination used to solve a tri-diagonal system of equations exactly [86]. This one-dimensional method is expanded into three dimensions by taking one line of cells in the volume at a time and solving them with the TDMA. The neighboring cells not included in the line are added into the source term. The method first takes one plane and performs a TDMA on each row and column within the plane. It first sweeps the rows and columns within a k plane in the positive i direction, the positive j direction, the negative i direction, then the negative j direction. It moves on to the next k plane and repeats. The method loops through all the k planes in the volume, then repeats in the other two coordinate directions, until the solution converges.

The material properties are contant for all simulations. Water has an electrical conductivity of 100 μS/cm and a dielectric constant of 80. Air has an electrical conductivity of 2.5×10^{-10} μS/cm and a dielectric constant of 1. The simulations were run under steady state conditions and the electrical impedance was calculated.

4.2 Results and Discussion

4.2.1 2-Dimensional Simulations in Fluent

2D simulations were run using Fluent [83]. These simulations consisted of a square cross section of a water-filled channel with electrodes on opposite sides. Both circular and square air regions were placed in the center of the channel as shown in Figure 4.2. The void size was varied to obtain a range of void fractions. The circular cross section voids ranged in void fraction from zero (no void) to 0.75. The geometric upper limit of the circular voids is 0.785, which corresponds to the area fraction of a circle inscribed in a square. The square voids ranged in void fraction from zero to 1.0 (no liquid).

For all cases, the relationship between the void fraction and the impedance is not linear. For void fractions less than 0.6 both the circular and square bubbles provided the same normalized impedance for a given void fraction, which is shown in Figure 4.3. Above a void fraction of 0.6, the normalized impedance calculated for circular bubbles is slightly higher than that for square bubbles. This is due to the difference in the thickness of the liquid film next to the electrodes for the two cases.

A third case consisted of four circular voids that varied in distance from one another inside the channel as shown in Figure 4.3. The radii of the voids did not change and thus the total void fraction remained constant at 0.256. The centroids of the voids were placed at three different spacings within the channel and the impedance was calculated. The ratio of the void diameter to the channel width was 0.29 and the ratio of the distance between void centers to the channel width was 0.31, 0.5, and 0.69. As shown in Figure 4.3, the change in the spacing of the four voids led to a very small change in impedance. In addition, the normalized impedances calculated for this case are approximately equal to the single square and circular void cases. This indicates that the shape and distribution of voids representative of the expected flow regimes have no significant effect on the resulting electrical impedance of the system, for voids modeled in parallel.

4.2.2 3-Dimensional Simulations in Fluent

3D simulations were run using Fluent [83]. These simulations consisted of a water-filled channel with electrodes on opposite faces. The length of the channel in the flow direction was taken to be the width of the electrodes such that the simulation domain is cubic. Using the image processing method previously described, air voids of real bubbles were placed in the channel.

It is important to note that void configurations were randomly chosen from the high-speed videos and the data is neither sequential nor time averaged for the 3D Fluent simulations. Three flow regimes were simulated: bubbly, slug, and churn flow. The void fraction estimated from the images ranged from zero to 0.72 for the random flow field snapshots chosen for analysis. The results are shown in Figure 4.4 and Figure 4.5. The first figure shows the magnitude of the impedance calculated for a given void fraction. The relationship between the two values is parabolic and the points lie on a single line regardless of the flow regime. The second figure shows the void fraction plotted with the normalized impedance. As with the 2-dimensional case, the relationship between the void fraction and the impedance is not linear. Additionally, despite snapshots taken from bubbly and slug flow regimes overlapping, all of the points appear to lie closely on a single line, continuing to indicate that the shape and distribution of the voids have no significant effect on the resulting electrical impedance for voids modeled in parallel.

4.2.3 3-Dimensional Simulations in MATLAB

3D simulations were run using the MATLAB code described previously. MATLAB was used in order to streamline the entire analysis from image processing through simulation without changing programs. These simulations consisted of a larger portion of a water filled channel with electrodes on opposite faces. Unlike the simulations in Fluent, the length of the channel was taken to be three times the width of the electrodes so that a portion of the channel before and after the electrodes was also modeled. This was done to capture a larger range of the electric field, specifically the lines that bend around air regions outside of the electrodes. Using the same image processing method previously described air regions corresponding to real bubbles were modeled in the channel.

Figure 4.6 shows the constant potential lines superposed onto the image of a slug bubble used in a simulation. The void fraction and simulated impedance for this video is 0.33 and 0.41, respectively. Instead of randomly chosen void configurations, sequential frames within videos were chosen. The total number of frames was chosen for each video to capture one period of the flow regime passing through the domain. For churn flow, a regime period is not identifiable and therefore a large number of frames were chosen so that it was representative of the particular case. Once again, three flow regimes were simulated: bubbly, slug, and churn flow. The estimated void fraction and the normalized impedance are shown in Figure 4.7 for all the cases ran. This data represents the instantaneous values for each image. Like before, the relationship between these two values is not linear. Since consecutive frames were taken for each video, the void fraction and calculated impedance varied over a range for each case. For bubbly and slug flows in which the flow alternates between large volumes of air and water, some of the frames contained partial bubbles that were not located centrally between the electrodes and instead were either entering or exiting the area of interest. The presence of these bubbles contributed to the calculation of the void fraction, but had a smaller effect on the impedance calculated due to their position away from the electrodes. Thus, for bubbly and slug flows having void fractions less than 0.4, a large spread can be seen in the data. In churn flow this happens much less often and the instantaneous data lie closely around a single line.

The void fraction and normalized impedance data calculated for the images taken from each video were averaged to generate a time-averaged dataset. This data is shown in Figure 4.8. The relationship between the time-averaged void fraction and the normalized impedance is once again non-linear. However, the shape of the curve is slightly different from the instantaneous data and different trends can be seen for different flow regimes. Bubbly flow, characterized by small void fractions below 0.2, produced impedance values that are slightly lower than a linear relationship. Slug flow, having void fractions between 0.1 and 0.4, produced impedance values that are slightly higher than a linear relationship. Churn flow, having void fractions above 0.3, produced impedance values that are even higher than a linear relationship would predict and appear to be more similar to the instantaneous data previously shown compared to the other two flow regimes.

The time-averaged simulated impedance data was plotted against the experimental data discussed in the previous chapter and is shown in Figure 4.9. For slug flow, the data matches well within 30% error bands and the mean average error for the data is 7.6%. However, the numerical simulations underpredict the impedance for bubbly flow and overpredict the impedance for churn flow.

4.3 Conclusions

Numerical simulations were performed with two-phase air-water adiabatic flow in a microchannel to predict the response of an impedance-based void fraction sensor. 2D and 3D simulations of the channel were run in Fluent for instantaneous responses of the electrical impedance sensor. MATLAB was used for 3D simulations for both instantaneous and time-averaged responses in a larger domain. It was found that the shape and distribution of the voids had no significant effect on the simulated impedance for void modeled in parallel. In addition, the relationship between the void fraction and the impedance is non-linear for all cases. Time-averaged 3D simulations were compared to experiments with good agreement and a mean average error of 7.6%.

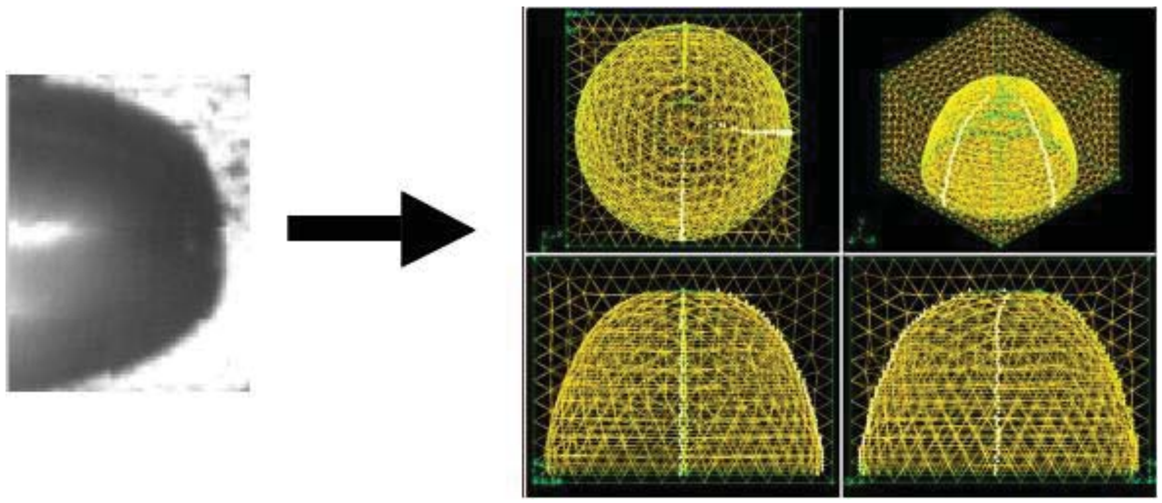


Figure 4.1. Using image processing to create 3-dimensional domains of real bubbles.

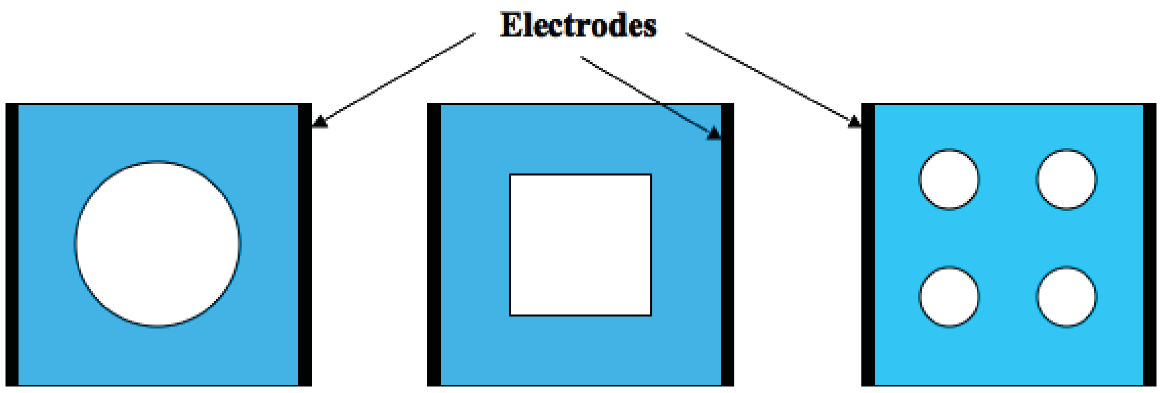


Figure 4.2. Cross sectional channel geometry with circular (left), square (middle), and four circular (right) air voids.

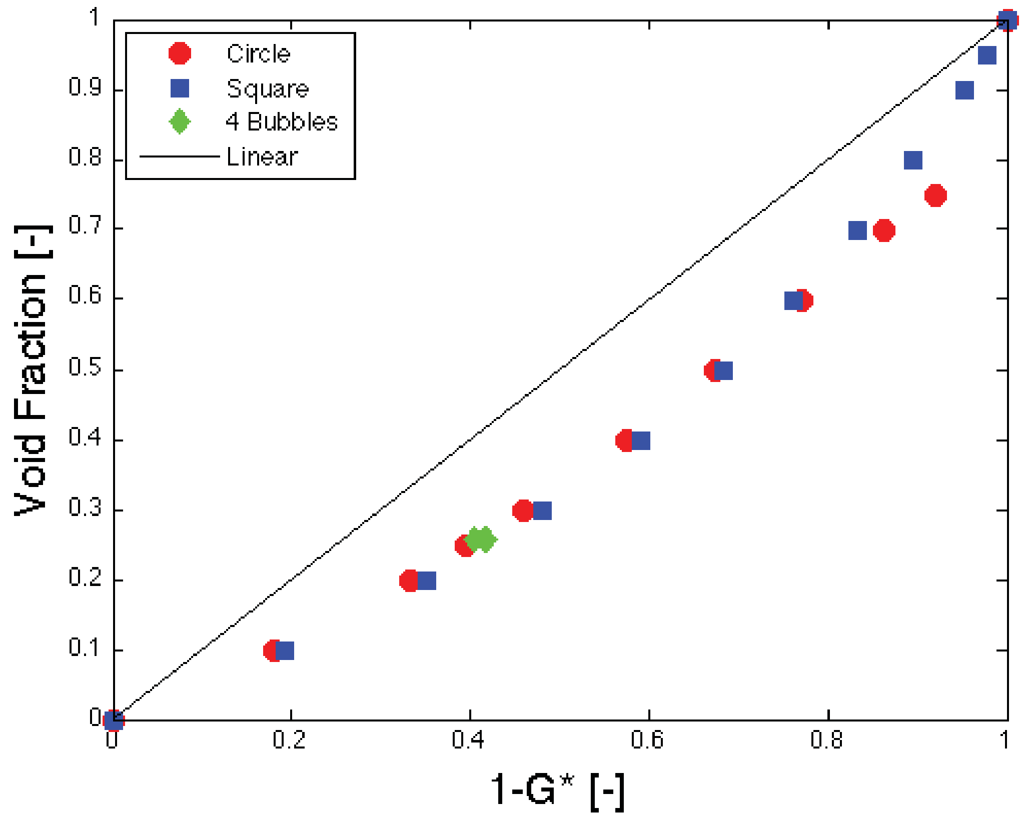


Figure 4.3. The results of the 2-dimensional simulations using Fluent.

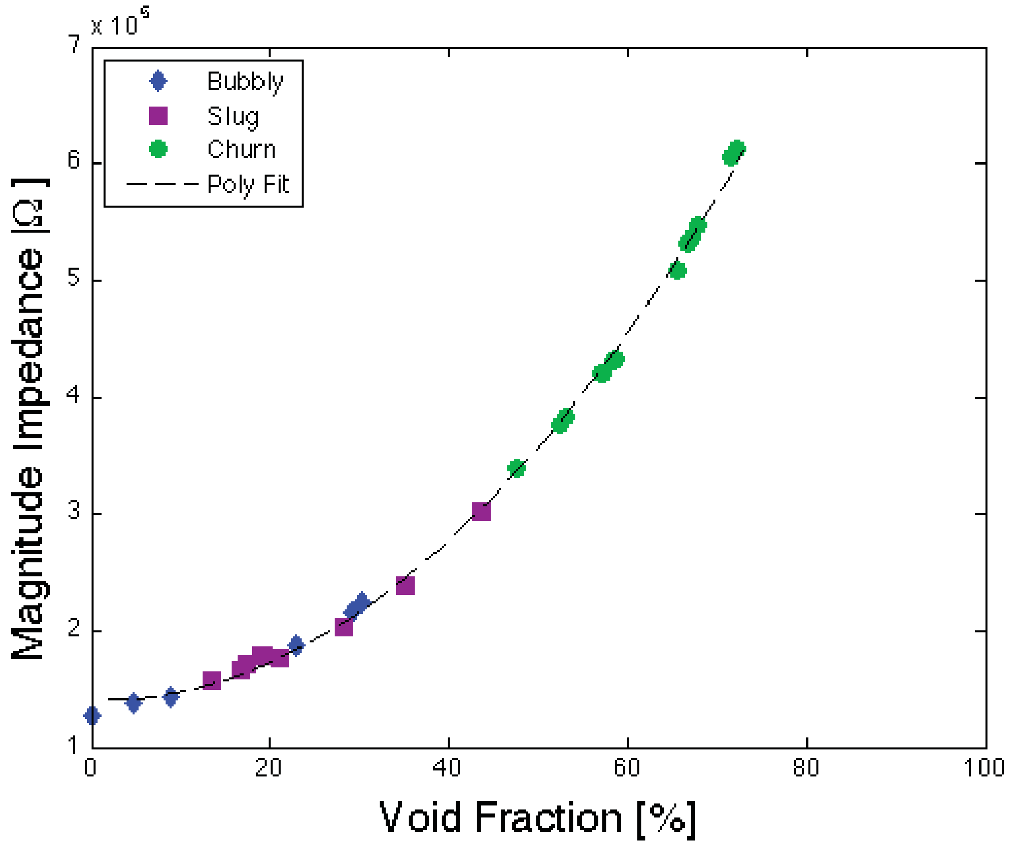


Figure 4.4. The magnitude of the impedance as a function of the void fraction for the 3-dimensional simulations using Fluent.

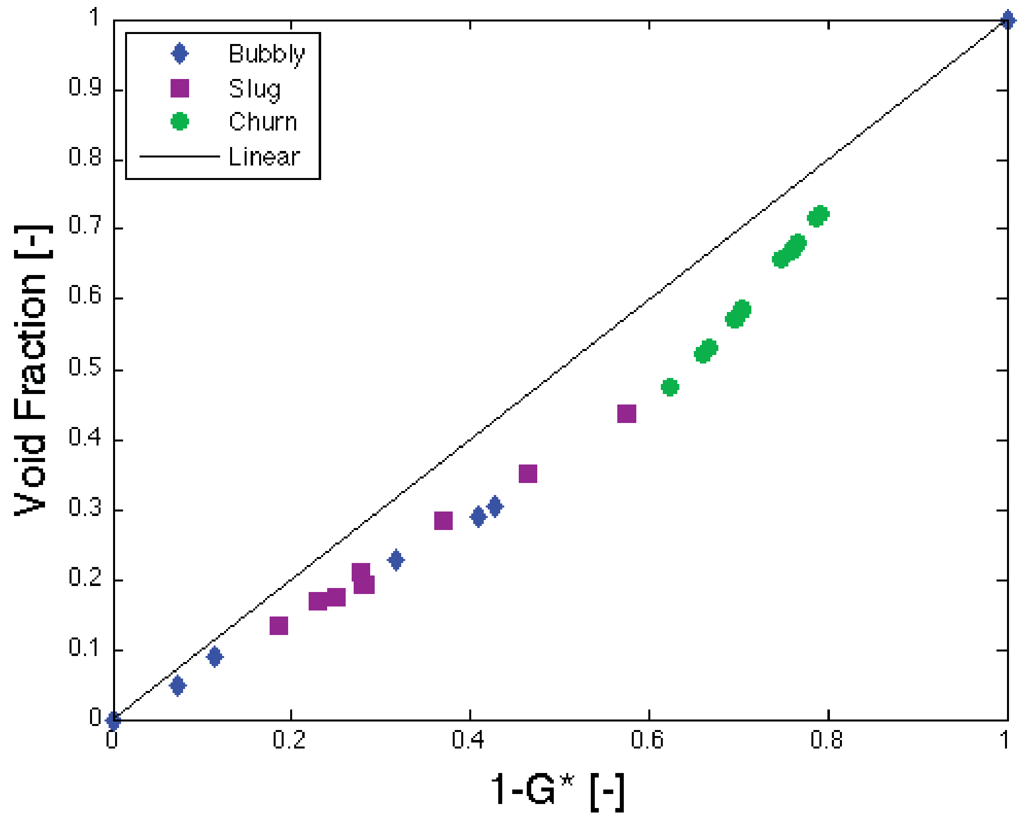


Figure 4.5. The void fraction plotted with the normalized impedance data for the 3-dimensional simulations using Fluent.

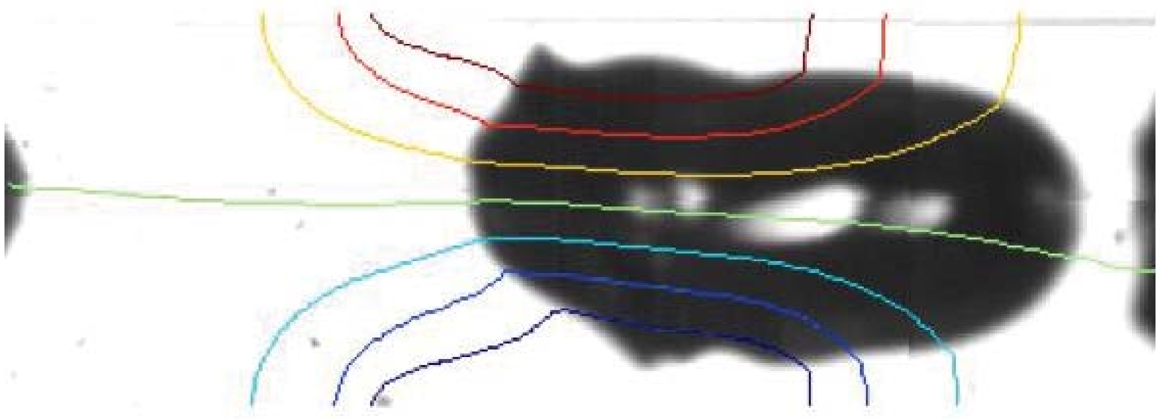


Figure 4.6. Constant potential lines superposed on an image of a slug bubble used in a simulation. The void fraction for this video is 0.33 and the simulated impedance is 0.41.

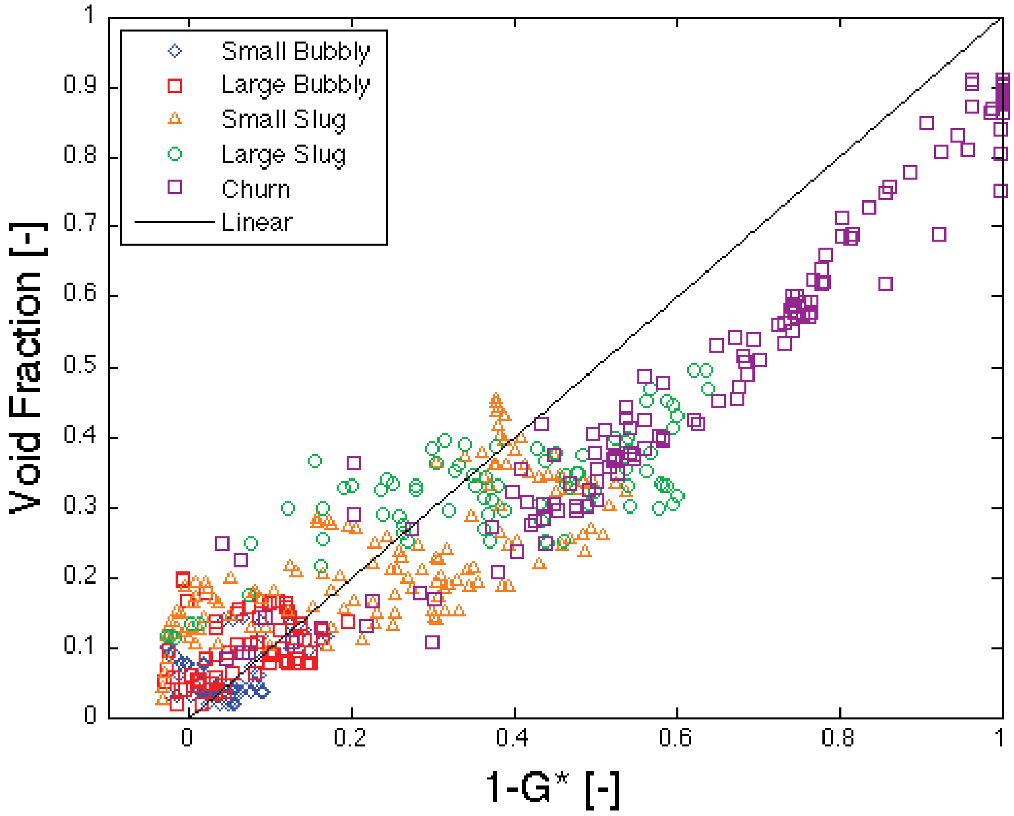


Figure 4.7. The instantaneous void fraction plotted with the normalized impedance data for the 3-dimensional simulations using MATLAB.

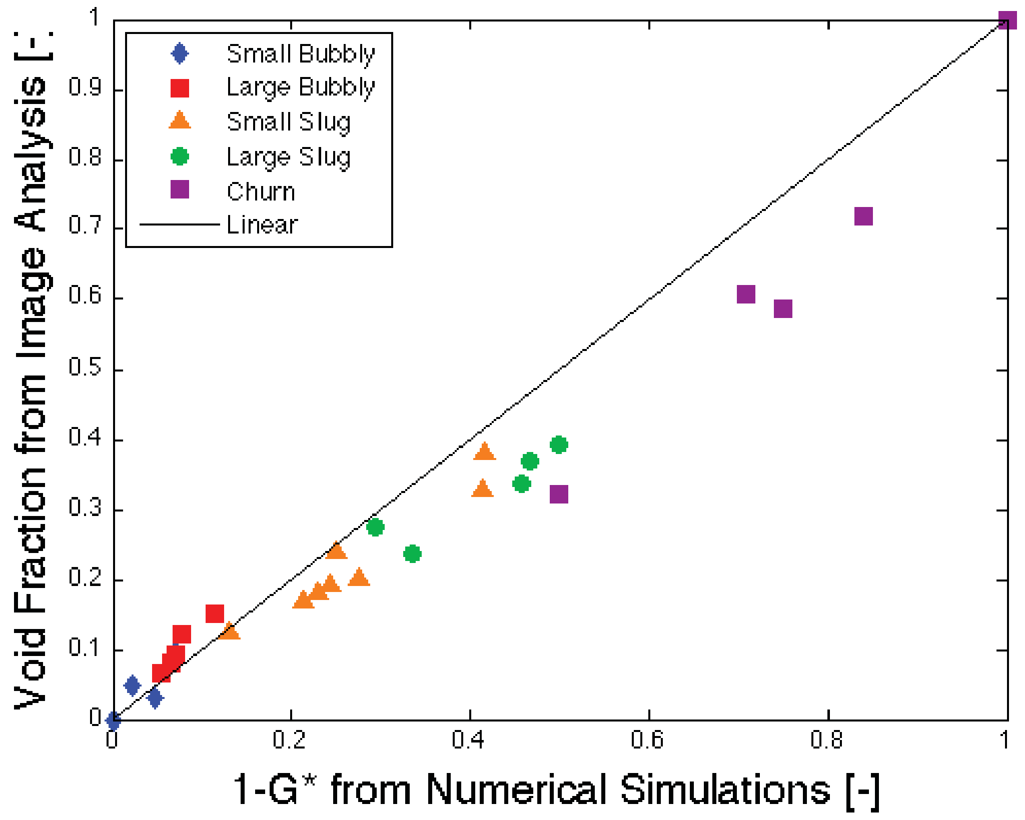


Figure 4.8. The time-averaged void fraction plotted with the normalized impedance data for the 3-dimensional simulations using MATLAB.

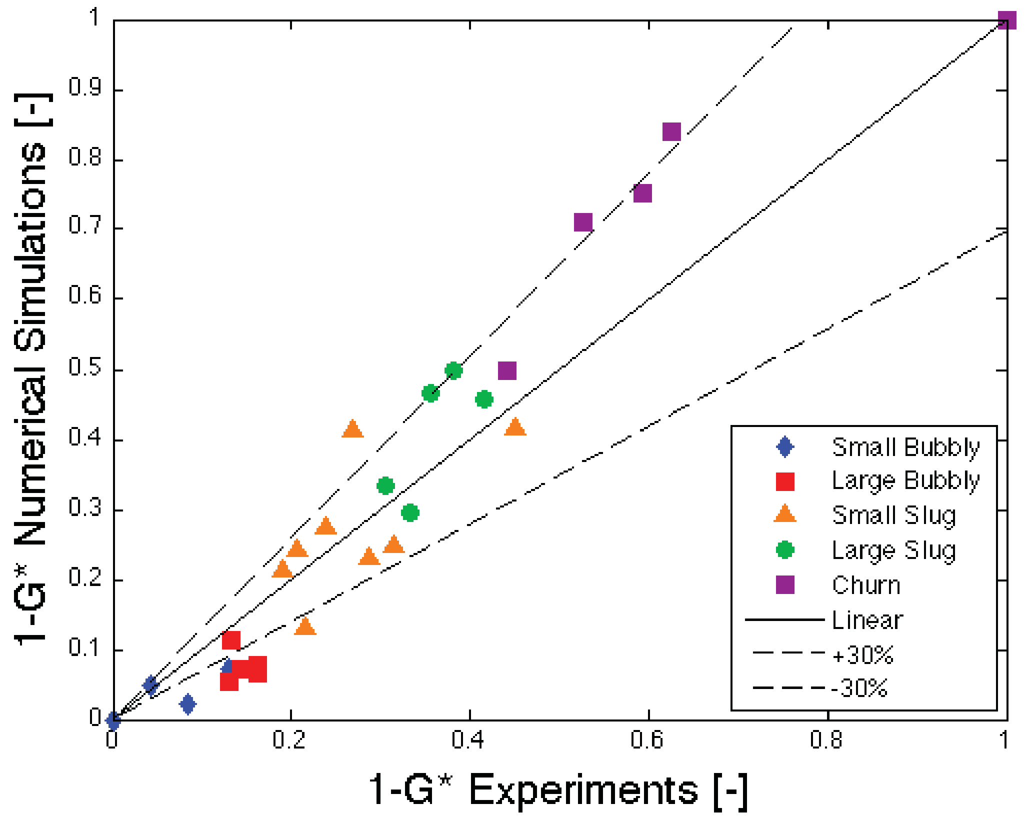


Figure 4.9. The time-averaged simulated impedance plotted with the measured impedance from experiments. The mean average error is 7.6%.

CHAPTER 5. VOID FRACTION MEASUREMENT USING STREAMWISE ELECTRODES

An electrical impedance-based void fraction sensor is used to measure the void fraction of air-water two-phase flow under adiabatic conditions in a microchannel. The electrical impedance of the mixture is dependent on the void fraction, flow topology, and the electrical properties of the fluids. Previous chapters studied a miniature electrical impedance-based void fraction meter with electrodes arranged in a crosswise geometry. The sensor was modified to place electrodes spaced at various separation distances along the streamwise direction that were flush mounted to the top of a 780 micron square channel. A high-speed camera was used to obtain flow visualizations to determine the time-averaged void fraction in order to calibrate the sensor. As an extension of the previous chapters, an experimental investigation of the impedance void fraction meter response with the electrodes arranged in a streamwise geometry is performed. The effects of the flow conditions, electrode spacings, and water electrical properties on the sensitivity of the instrument were explored.

5.1 Experimental Methods

5.1.1 Test Section

An experimental test section for void fraction measurements in air-water two-phase flow was fabricated in clear transparent acrylic to allow for high-speed imaging.

The experiment was designed by Sidharth Paranjape. A photograph of the test section is shown in Figure 5.1. This test section was designed to be the same as the crosswise electrodes test section discussed previously in 3.1.1with the exception of the placement of the electrodes. A square-shaped flow channel with side dimensions of 780 microns is cut into the base plate. The length of the channel is approximately 50.8 mm. Four 302 stainless steel electrodes are embedded in the top plate such that the faces of the electrodes are flush-mounted to the top wall of the channel. The electrodes are located in the middle of the flow stream, span the width of the channel, and have different streamwise gaps of 780 microns, 1560 microns, and 2340 microns (*i.e.*, one, two, and three channel widths, respectively). These electrode gaps can be tested by connecting different electrodes to the impedance measurement circuit. The width of the electrodes is identical to the width of the flow channel; the exposed area of each electrode to the channel is the same as the cross section of the channel. Inlet and outlet plenums are machined into the top cover plate to route water flow into the channel. Air is directly injected into the flow channel through a 0.3 mm diameter orifice at the bottom of the channel. The air inlet orifice is located 10 mm from the inlet of the flow channel. The electrodes are connected to the electronic circuit via alligator clips.

The same flow loop is used as in the previous crosswise electrodes experiments and is shown in Figure 3.2. De-ionized water is used for the liquid stream; morpholine and ammonium hydroxide are once again added to the water in order to increase its electrical conductivity while keeping the pH value near 7. The effect of various electrical conductivities of the water was explored and the exact amounts of the chemicals varied between tests. The impact of the addition of these chemicals on the flow regimes (through an effective change in the surface tension) is negligible [23]. The specific conductivity of water was maintained between 50 and 300 μS/cm. The water flow loop is equipped with a frequency-controlled water pump and a needle valve to control the water flow rate. The water flow rate is measured with two McMillan micro-turbine flow meters with ranges of 0 to 100 mL/min and 0 to 200 mL/min. Air flow is provided by a compressed air cylinder equipped with a pressure regulator. Two Omega FMA6700 series air mass flow sensors with ranges of 0 to 200 mL/min and 0 to 500 mL/min were used to measure the air flow rate through the test cell. The flow sensors also measure the temperature and pressure of the gas at the flow meter. The measured temperature and pressure are used to correct the mass flow rate from standard conditions since the flow sensor is factory-calibrated at standard temperature and pressure. The air flow rate is controlled by a needle valve. The storage tank is open to the atmosphere and also serves as an air-water flow separator. Special care is taken to avoid flow instabilities occurring due to the accumulations of air in various tube fittings in the exit section of the flow loop. In order to achieve this, flexible tygon tubing is used to connect the exit of the test section to the storage tank, which is located at a higher elevation than the test section.

5.1.2 Impedance Void Fraction Meter

The same impedance void fraction meter used in the crosswise electrodes experiments is again used here. A description of the sensor can be found in 3.1.2 and the signal-processing scheme is shown in Figure 3.3.

5.1.3 Image Analysis

A high-speed Photon Fastcam-Ultima APX digital video camera along with a Keyence VH-Z50L lens at 100X magnification was used for flow visualization. The videos are recorded at 30,000 frames per second with a shutter speed of 30,000 Hz. A Henke-Sass Wolf illumination source is used to illuminate the microchannel for visualization. This combination provides a special resolution of approximately 8 μm per pixel. The digital videos are acquired for 4 seconds for each flow condition. The optical images are processed in MATLAB in order to obtain an experimental void fraction measurement. The same image processing techniques used in the crosswise electrodes experiments are again used here. A description of the algorithm can be found in 3.2.1. Figure 3.4 shows the morphological operations performed for each step.

5.1.4 Procedure

When running a test, a water flow rate is chosen and remains the same throughout the test. A full air and full water reading is taken at the beginning and end of each test so that the impedance meter output can be normalized. The water flow rate is kept constant, and the air flow rate is incrementally increased to vary the flow regime; data is recorded once the flow conditions are stable at the desired test point.

5.2 Results and Discussion

5.2.1 Calibration of the Impedance Void Fraction Meter

The void fraction from image analysis was plotted against the air volumetric flow fraction, β, and is shown in Figure 5.2. The legend shows the water volumetric flow rate in mL/min, the gap spacing of the electrodes (1CW means one channel width), and the electrical conductivity of the water in μS/cm, respectively. The color of the markers indicates the water electrical conductivity and the shape indicates the spacing of the electrodes. The data is also plotted with the homogenous equilibrium and drift flux models as described in Section 3.3.1. All of the data lie close to the two models and there is not a significant difference by varying the spacing of the electrodes, the water flow rate, or the water conductivity.

The void fraction was measured under a variety of different flow conditions. Each flow condition was characterized by the velocity inlet boundary conditions: the volumetric flux of gas, $\langle j_g \rangle$, and volumetric flux of liquid, $\langle j_f \rangle$. The ranges of flow conditions covered in these experiments are 0.17 m/s < $\langle j_g \rangle$ < 13.7 m/s and 0.68 m/s < $\langle j_f \rangle$ < 5.48 m/s.

The measured, instantaneous impedance is normalized as

$$G^* = \frac{G_m - G_1}{G_0 - G_1}, \tag{5.1}$$

where G_m is the instantaneous two-phase mixture admittance, G_0 is the admittance with a void fraction of zero (water only) and G_1 is the admittance with a void fraction of one (air only). The liquid fraction is a monotonically increasing function of normalized admittance, G^*, which means the void fraction is proportional to $\alpha_{imp} = 1 - G^*$. The impedance meter is calibrated in a time-averaged sense. That is, the time-averaged value of the impedance meter reading $\langle \alpha_{imp} \rangle_t$ is compared with the time-averaged void fraction $\langle \alpha \rangle_{V,t,im}$ obtained by flow visualization.

Figure 5.3 shows the data from the impedance meter versus the void fraction obtained from image processing for a variety of water electrical conductivities. Like the previous figure, Figure 5.2, the legend shows the water volumetric flow rate, the electrode spacing, and the electrical conductivity of water. It was found that the water flow rate and electrode spacing had almost no effect on the results; however the electrical conductivity of the water produced a significant change. Very high conductivities of 200 μS/cm and higher as well as a low conductivity of 50 μS/cm produced very low impedance values even at high void fractions. Other conductivities between 50 and 200 μS/cm produced impedance values as expected.

5.2.2 Dependence on Water Electrical Conductivity

The data show that the instrument has a nearly linear response, but the slope depends on the electrical conductivity of the water. High and low conductivities of approximately 50, 200, and 300 μS/cm produced a large slope while intermediate conductivities of approximately 100, 125, 150, and 175 μS/cm produced a slope close to one. This indicates that there is an optimal range for maximum sensitivity of the instrument.

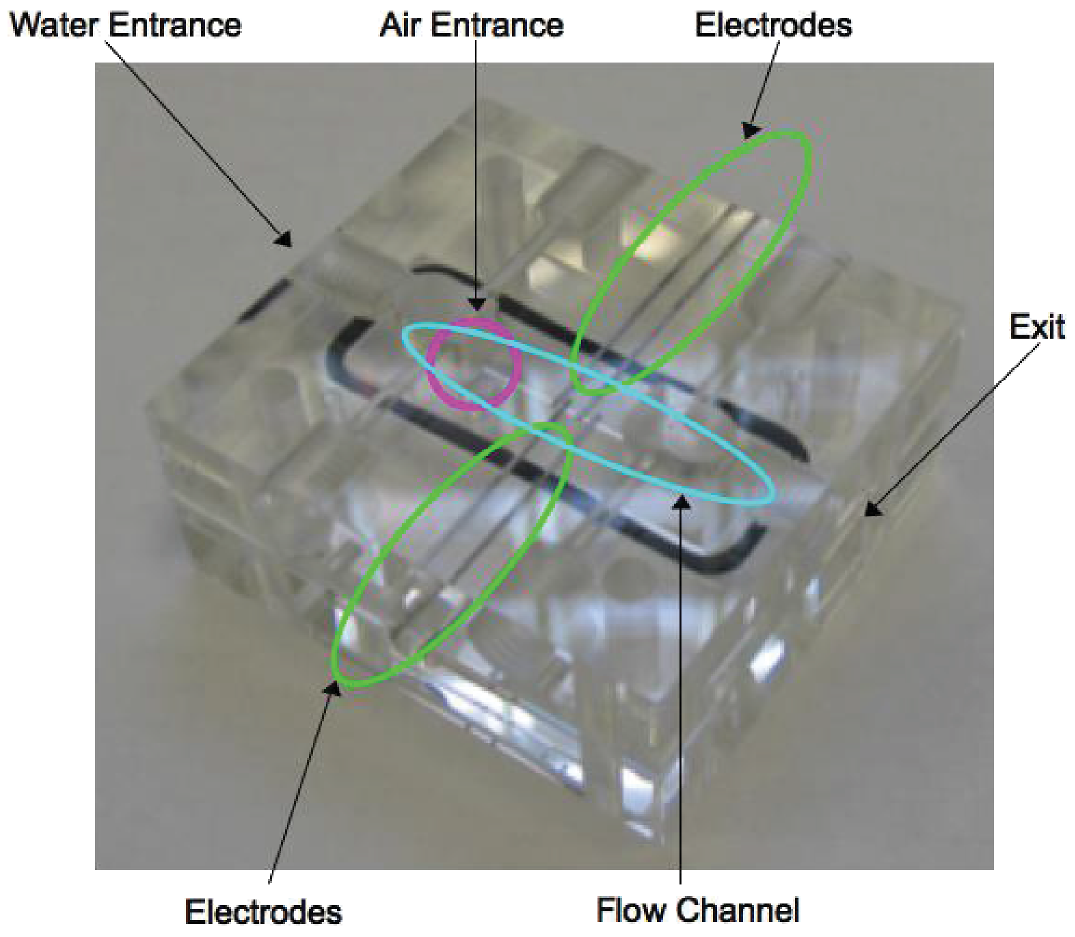
The sensitivity is defined as the percent of the total range of the impedance meter that is used. This can be found by finding the inverse of the slope of a linear best-fit line for a single test. For example, the test run with a water flow rate of 100 mL/min, electrode spacing of one channel width, and a water conductivity of 171 μS/cm, has a linear best-fit line with a slope of 1.86 which means the sensitivity of the instrument at that conductivity is about 53.8% by this definition. Figure 5.4 shows the sensitivity of the impedance meter plotted as a function of water electrical conductivity. Conductivities between 100 and 175 μS/cm had the highest sensitivities of about 55% while all other conductivities produced low sensitivities below 20%. The optimal range of the water electrical conductivity for this instrument is between 100 and 175 μS/cm.

The impedance meter reading was correlated to the water conductivity as shown in Figure 5.5. A linear line was fit to the data and is shown in addition to 30% error bands. Most of the data in this figure lie between the bands regardless of water flow rate or electrode spacing. In addition, the data show a much larger spread and utilize almost the entire range of the normalized axes. If it is desired to operate outside of the optimal range of water conductivities, it can be done by simply adjusting the impedance meter output according to the water conductivity.

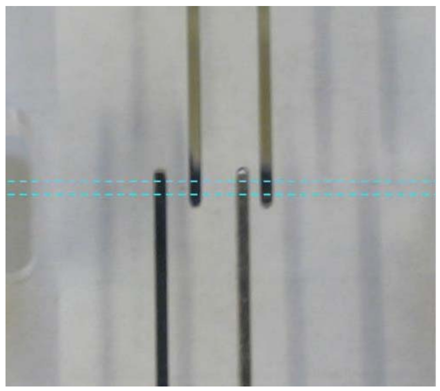
The impedance meter with a streamwise electrode geometry can easily be implemented in a microchannel heat sink under boiling conditions. Although the electrical conductivity of the fluid varies with temperature, the range in fluid temperatures in this scenario is quite small leading to a relatively constant conductivity [84]. Even though the calibration of the impedance meter has been conducted under adiabatic conditions, the behavior of the sensor is expected to remain the same.

5.3 Conclusions

The void fraction was measured in air-water two-phase flow in a microchannel with a 780 × 780 square cross section using a custom-designed impedance void fraction meter. The measurements from the impedance void fraction meter were plotted against the time-averaged void fraction determined from flow visualization using a high-speed camera. Multiple water electrical conductivities were tested and a clear dependence was shown. For maximum instrument sensitivity, an optimal range between 100 and 175 μS/cm was found. In addition, the impedance meter output can be adjusted according to the water conductivity to collapse all of the data onto a single line.



(a)



(b)

Figure 5.1. Impedance meter test cell. (a) Top view of the test cell. (b) Top plate with electrodes. The blue dashed lines indicate where the channel is located.

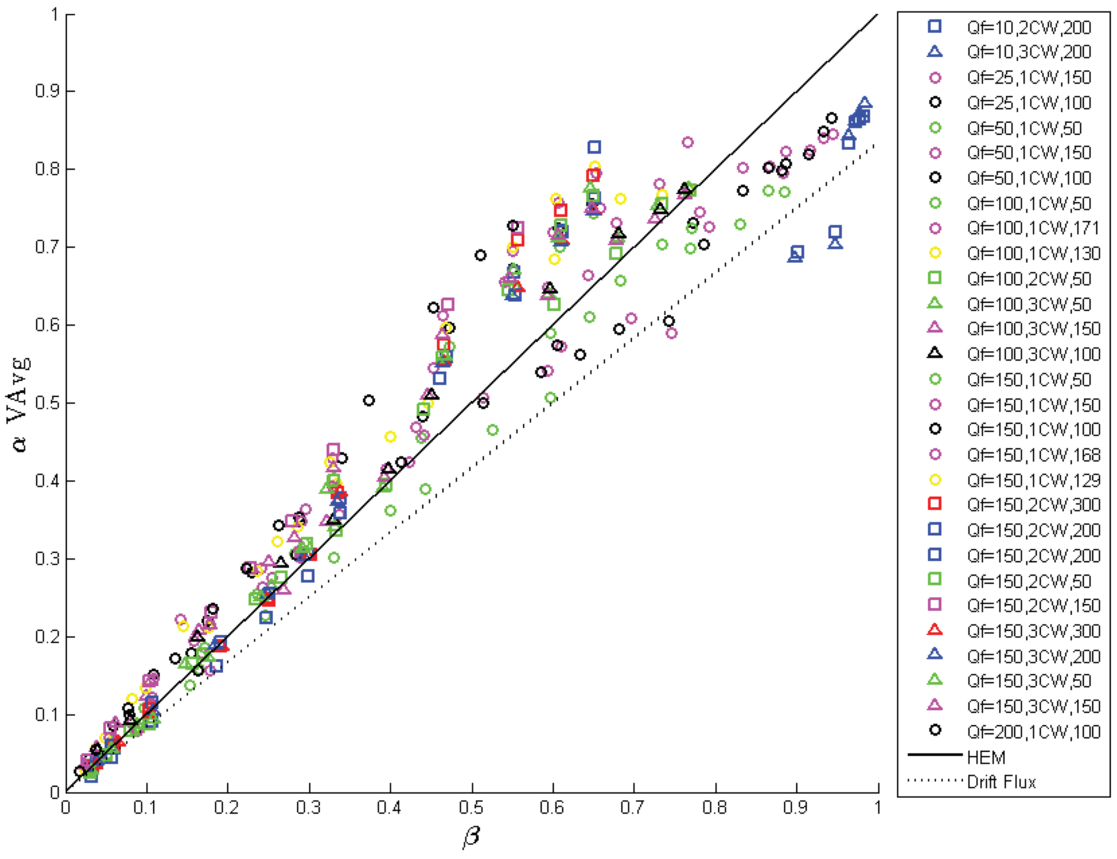


Figure 5.2. Void fraction from image analysis plotted against the air volumetric flow fraction for a variety of flow rates. The legend shows the water volumetric flow rate in mL/min, the spacing of the electrodes (1CW means one channel width), and the electrical conductivity of the water in μS/cm, respectively.

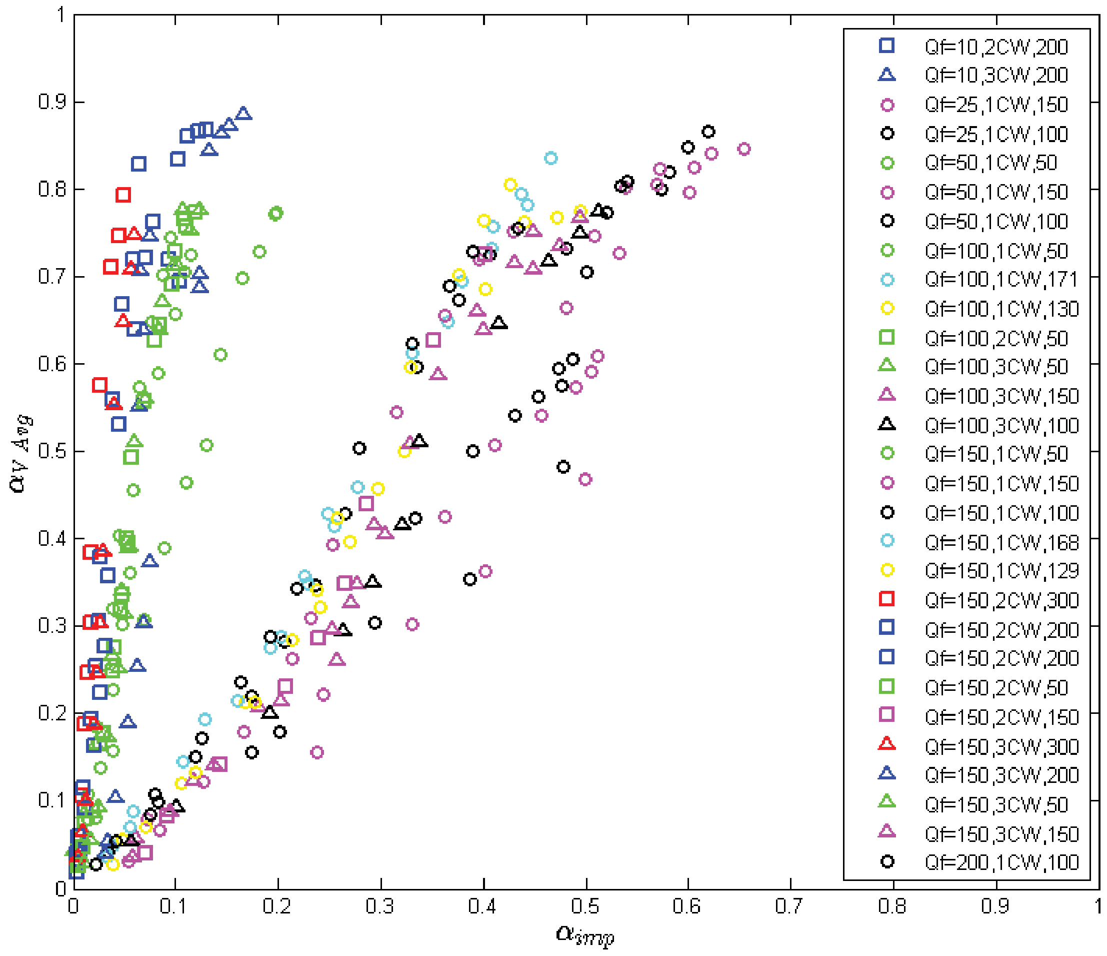


Figure 5.3. Void fraction from image analysis plotted against the normalized impedance void fraction meter output for a variety of water conductivities. The legend shows the water volumetric flow rate in mL/min, the spacing of the electrodes (1CW means one channel width), and the electrical conductivity of the water in µS/cm.

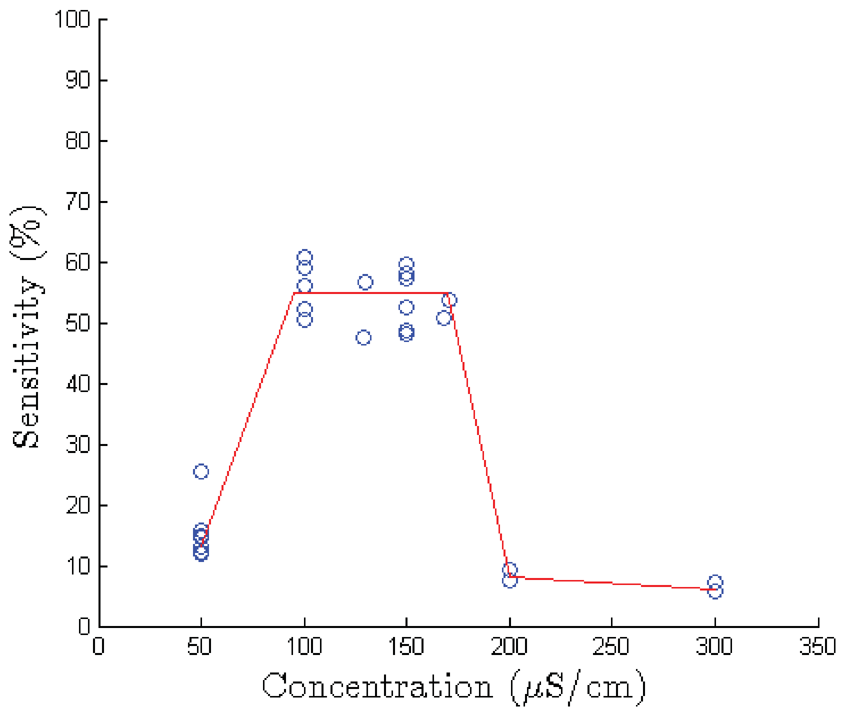


Figure 5.4. The sensitivity of the impedance void fraction meter as a function of the water electrical conductivity.

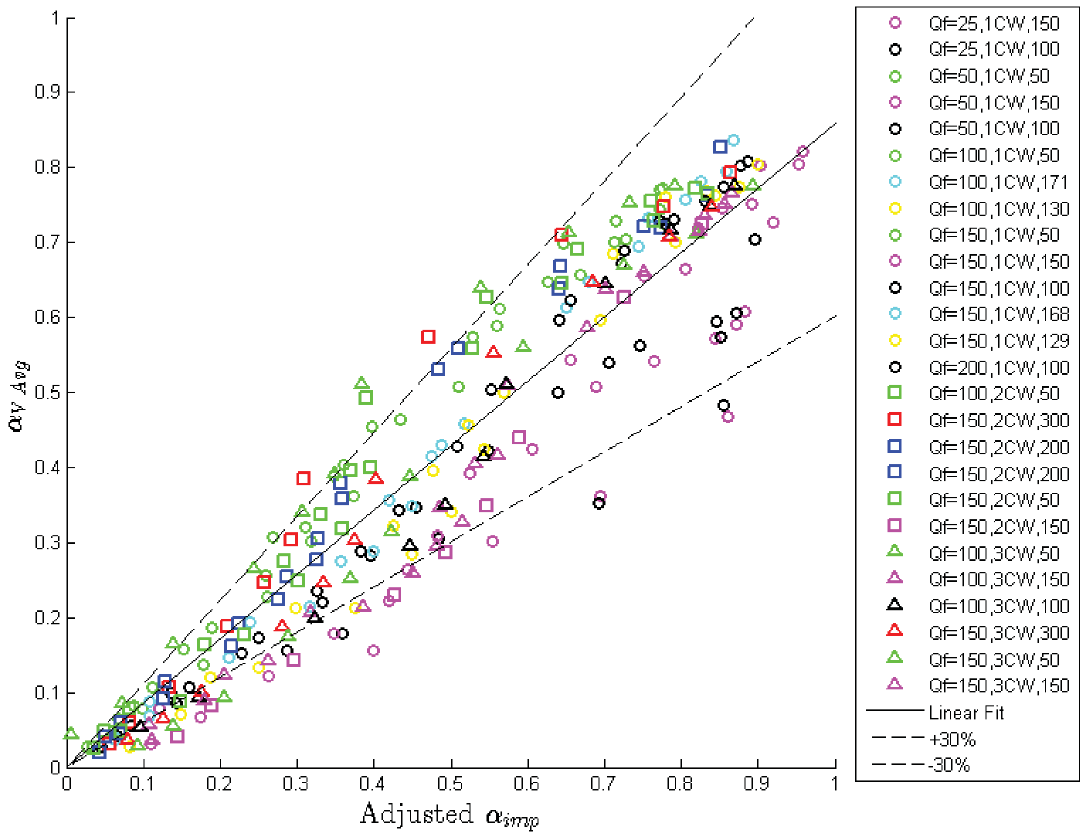


Figure 5.5. The void fraction from image analysis plotted against the adjusted normalized impedance void fraction meter output for a variety of water conductivities. The legend shows the water volumetric flow rate in mL/min, the spacing of the electrodes (1CW means one channel width), and the electrical conductivity of the water in µS/cm.

CHAPTER 6. EFFECTS OF NON-UNIFORM HEATING ON BOILING IN MICROCHANNELS

As electronics packages become increasingly thinner and more compact due to size, weight, and performance demands, the use of large intermediate heat spreaders to mitigate heat generation non-uniformities are no longer a viable option. Instead, non-uniform heat flux profiles produced from chip-scale variations or from multiple discrete devices are experienced directly by the ultimate heat sink. In order to address these thermal packaging trends, a better understanding of the impacts of non-uniform heating on two-phase flow characteristics and thermal performance limits for microchannel heat sinks is needed. This chapter studies flow boiling phenomena in a microchannel heat sink with local hotspots, as well as increasingly non-uniform peak-heating profiles across the heat sink, both in the flow direction and perpendicular to it, with respect to thermal performance and flow boiling phenomena. This work enables better assessment of existing heat transfer models for prediction of non-uniform heating profiles. The material in this chapter was presented at the *ASME International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems* in July 2013 and published in the proceedings [87]. It was later refined and published in the *International Journal of Heat and Mass Transfer* [88].

6.1 Experimental Methods

6.1.1 Test Section

The microchannel test section used in the experiments was described in detail by Harirchian and Garimella [11]; it was modified for the purposes of the current study and is shown in Figure 6.1. A transparent, polycarbonate manifold cover plate seals and routes the working fluid through a silicon microchannel heat sink with a base area of 12.7 mm × 12.7 mm. The total silicon thickness is approximately 650 μm. The heat sink is mounted on a printed circuit board (PCB)[1] that is offset from an electrical quick-connect board with an insulating G10 glass-epoxy composite layer. An insulating 0.4 mm thick borosilicate glass sheet is sandwiched between the microchannel heat sink and cover plate to protect the polycarbonate (rated to a temperature of 115-130 °C), and forms the rigid top wall of the microchannels. The fluid enters the channels through an inlet header section with a flow length of 10 mm, width of 12.7 mm, and a height equal to that of the heat sink plus borosilicate glass thickness.

Parallel microchannels are cut into the top surface of the silicon chip using a dicing saw, and are shown in Figure 6.2. A single heat sink with 35 microchannels was used for the experiments (240 μm channel width, 370 μm channel depth, and 100 μm fin width). Each channel was cut with a number of passes, which created some waviness on the bottom surface. The average channel bottom roughness in the region of a single cut is 0.2 μm, and the overall average surface roughness of the bottom and sides of the channels are 0.82 μm and 0.1 μm, respectively.

[1] The author would like to thank Bruce Myers and Darrel Peugh of Delphi Electronics and Safety, Kokomo, Indiana, for providing the silicon microchannel heat sink.

A 5 × 5 array of resistance heaters and temperature-sensing diodes is fabricated on the bottom side of the heat sink, as shown in Figure 6.2. Since the individual heater resistances are nearly identical, a single voltage can be applied across multiple heaters in parallel to provide a uniform flux over a desired area. Up to two DC voltage power supplies are connected to provide the customized, non-uniform heat flux profiles applied to the underside of the microchannels investigated in the current study. The heat generated and local temperature at each element are calculated based on the calibrated heater/sensor resistance and the applied voltage. The relationship between the voltage and temperature of each sensor is calibrated in a convection oven. More details about the calibration procedure for each element can be found in [89].

6.1.2 Flow Loop

The experimental flow loop used is the same as that described by Harirchian and Garimella [89], and a schematic diagram is shown in Figure 6.3. The dielectric fluid, FC-77, is circulated through the flow loop using a Micropump 415A magnetically coupled gear pump. A preheater sets the fluid to the desired inlet temperature upstream of the test section. Downstream of the test section, a liquid-to-air heat exchanger cools the fluid back to room temperature before it enters the reservoir. A McMillan Flo-114 liquid flow meter, with a range of 20-200 mL/min, measures the liquid flow rate through the loop. T-type thermocouples are located upstream of the preheater, upstream and downstream of the test section, and downstream of the heat exchanger. A 2200 series Omega differential pressure transducer measures the pressure drop across the test section.

High-speed visualization is performed with a Photron Fastcam Ultima APX high-speed digital video camera and a Nikon ED 200 mm lens. A Sunoptics Titan 300 xenon arc lamp is used for inline illumination of the test chip for the visualizations. Images are extracted from high-speed videos captured at 6,000 frames per second with a shutter speed of 6 kHz.

6.1.3 Test Procedure

Before running a test, the liquid is degassed using an expandable reservoir and a vacuum pump. The degassing procedure and the design of the expandable reservoir are adapted from [90]. The test fluid, FC-77, contains 41% air by volume, or 283 ppm, at ambient temperature and pressure. An expandable container with a locking mechanism allows expansion and contraction of the reservoir to control the system pressure. First, the reservoir is expanded to create a gas space at the top of the reservoir. A vacuum pump connected to the top of the reservoir lid is turned on for 5 minutes to remove air and the FC-77 vapor that has collected in the gas space. The reservoir is left expanded and at a vacuum pressure for one hour to allow air to diffuse from the liquid into the gas space. The process is repeated until the pressure in the reservoir remains constant with time, indicating that air is not actively dissolving out of the liquid in the reservoir. The fluid is cycled through the loop and the reservoir degassing process is repeated several times. To ensure the fluid is fully degassed, the system is set to atmospheric pressure, fluid is pumped through the loop, and the preheater is used to boil the fluid. The measured preheater fluid temperature at incipience is confirmed to be the saturation temperature of FC-77 (97 °C).

Experiments are conducted at a single mass flux of 890 kg/m^{2}s. Fluid is pumped through the loop at a constant flow rate and preheated to approximately 91 °C, which corresponds to a subcooling of 6 °C at the inlet to the heat sink. The flow rate and inlet temperature are maintained at a constant values throughout the test. The expandable reservoir is used to set the system at atmospheric pressure prior to turning on the heater elements. During testing, the system pressure increases slightly due to the bulk temperature rise of the fluid; however, this increase is minor (12.4 kPa), and smaller than is practically correctable with the expandable reservoir system.

6.1.4 Test Cases

A variety of heating cases were investigated as summarized in Figure 6.4. The heat transfer coefficients, wall temperatures, fluid temperatures, and the locations of boiling via high-speed imaging are obtained for each case.

The first cases correspond to hotspots that span either the width or length of the microchannel heat sink in transverse and streamwise directions: singular central transverse (1a), central streamwise (1b), inlet transverse (1c), and two transverse hotspots (1d) at the inlet and outlet. The hotspot heater locations are turned on (shown in red) while the rest are powered off (shown in gray). For these hotspot heating cases, the heat supplied to the strips of active heaters is incremented from zero until the maximum heat flux for the test is reached. The maximum heat flux limit is reached when the wall temperature reaches 140 °C, to prevent the solder bumps in the test chip from degrading.

The second set of test cases consider a non-uniform heating condition where a peak heat input is imposed along the width and length of the microchannel in the transverse (2a) and streamwise (2b) directions. In these latter two non-uniform heating cases, the total power input to the chip remains constant, but the local power input distribution is adjusted to increase the disparity between the peak and background heat fluxes. The total constant power input in this second set of cases is the same as the maximum power input for the corresponding hotspot heating cases.

6.2 Data Reduction

The data reduction method presented here is a modified version of the one used by Harirchian and Garimella [9]. Key modifications to this process take into account the enhanced substrate spreading that occurs for non-uniform heating profiles. Pressure-dependent local fluid properties and saturation temperatures are accounted for in the data reduction procedure to account for variations along the flow length. A MATLAB script of the data reduction process can be found in Appendix C.

The local heat transfer rate from the microchannels to the fluid, $\dot{q}_{net}$, is calculated based on an energy balance for each heating element as

$$\dot{q}_{net,ij} = \dot{q}_{gen,ij} - \dot{q}_{loss,ij} - \dot{q}_{cond,ij}. \tag{6.1}$$

The energy generated by the heating elements is denoted as $\dot{q}_{gen}$ and is calculated as $\dot{q}_{gen} = V^2/R$. The heat loss from each heating element is by natural convection to the ambient air, radiation to the surroundings, and conduction from the microchannel heat sink to the cover plate and circuit board. A relationship between the base temperature and heat loss is experimentally obtained via measuring the amount of heat input that can be sustained before the test section is charged with coolant. A complete description of the procedure used to obtain the heat loss for each sensor is found in [91]. The energy conducted laterally from one heating element to the next is denoted as $\dot{q}_{cond}$. When non-uniform heating profiles are imposed, there is significant lateral conduction of heat through the silicon heat sink. Heat conduction between elements is calculated as

$$\dot{q}_{cond,ij} = k_{Si}\frac{L}{5}(t-d)\frac{\left(4T_{i,j} - T_{i+1,j} - T_{i-1,j} - T_{i,j+1} - T_{i,j-1}\right)}{L/5}, \tag{6.2}$$

where the total net conduction is dependent on the four neighboring elements to heater i,j.

During single-phase flow, the bulk fluid temperature above each heating element is calculated as

$$T_{f,ij} = T_{in} + \frac{\sum_i \dot{q}_{net,ij}}{GwdN/5c_p}, \tag{6.3}$$

where $\sum_i \dot{q}_{net,ij}$ is the sum of the net heat transfer to the fluid from the inlet to the heating element in question. The fluid temperature rise is based on the available sensible heat up until the saturation temperature is reached, at which point the fluid temperature is set equal to the saturation temperature.

The local wall temperature is corrected from the measured diode temperature by accounting for conduction from the substrate to the base of the microchannel, calculated as

$$T_{w,ij} = T_{d,ij} - \frac{q''_{b,ij}(t-d)}{k_{Si}}. \tag{6.4}$$

The heat flux through the base is calculated from the local net heat transfer rate as

$$q''_{b,ij} = \frac{\dot{q}_{net,ij}}{A_b/25}. \tag{6.5}$$

The local heat transfer coefficient for each heating element, which represents an average along the channel height at a particular point along the flow length, is calculated considering the microchannel walls as extended fins, according to

$$h_{ij} = \frac{q''_{w,ij}}{\eta_o\left(T_{w,ij} - T_{f,i}\right)}, \tag{6.6}$$

where q''_w is the wall heat flux calculated using the net heat transfer rate, $\dot{q}_{net}$, and the total wetted area of the microchannels, A_w. η_0 is the overall surface efficiency of the microchannel heat sink, defined as

$$\eta_0 = 1 - \frac{NA_f}{A_w}\left(1 - \eta_f\right), \tag{6.7}$$

where A_f represents the wetted area of a microchannel fin and η_f is the efficiency of a fin with an adiabatic tip. This adiabatic assumption is valid due to the heat transfer to the cover plate being significantly lower than the heat transfer to the liquid in the microchannels. It is calculated as

$$\eta_f = \frac{\tanh md}{md}, \tag{6.8}$$

where

$$m^2 = \frac{2h}{k_{Si}w_f}. \tag{6.9}$$

The heat transfer coefficient is initially calculated assuming an overall surface efficiency of 100% and is iterated until the value converges. The overall efficiencies of the microchannel heat sinks were found to be above 95.6% for all cases.

6.2.1 Uncertainty Analysis

Since the facility is the same as that used by Harirchian [89], the same uncertainty analysis is also used. The flow meter has a measurement uncertainty of 1% of full scale and the pressure transducers have a measurement uncertainty of 0.25% of full scale. The uncertainty measurements of the channels dimensions are ±15 μm, the T-type thermocouples ±0.3 °C, and the diode temperature sensors 0.3 °C. The microheater resistance measurement uncertainty is 0.002% and the applied voltage measurement uncertainty is 0.004%. The uncertainties for the wall heat flux and heat transfer coefficient were found using a standard uncertainty analysis [92] and are found to be 2.0 to 11.4% and 2.2 to 11.7%, respectively. These two uncertainties are largely affected by the uncertainties in the measurement of the channel area; the uncertainties in the net heat transfer rate, wall temperature, and saturation temperature are small in comparison. Please see Harirchian [89] for the detailed analysis.

6.3 Results and Discussion

The results are split into two heating cases as previously described: (1) hotspots that span the length or width of the heat sink tested with increasing power input against an unpowered background, and (2) non-uniform heating conditions with a peak along the width or length of the heat sink.

In cases 1a through 1c, 5 of the 25 individual heating elements are powered up to simulate a hotspot while the rest are unpowered. The total power supplied to these heating elements is incremented until the maximum allowable wall temperature is reached. In case 1d, a dual hotspot, 10 of the 25 individual heating elements are powered up. For the second set of non-uniform heating cases (2a and 2b), all of the heating elements are initially supplied the same power level, resembling a uniform heating case. The power to 5 of the 25 heating elements is proportionally incremented, while maintaining a constant total power input to the entire test section. A subset of the data is presented in the figures in this chapter. Figures containing the full dataset can be found in Appendix D.

6.3.1 Case 1: Hotspot Heating

The maximum total power input, maximum local heat flux at that power, and maximum local wall temperature are summarized for all cases in Table 6.1. For Case 1, as the power input increases, the heat flux to the fluid, q_w'', always reaches a maximum above the active heater elements. The individual trends for each single hotspot are described below. Case 1d is discussed in Appendix D.

6.3.1.1 Case 1a (Central Transverse Hotspot)

The first heating profile tested was with a central transverse hotspot. The five transverse heater elements located along the center of the flow length were supplied with power, while the remaining 20 were turned off. The maximum heat flux recorded is 24.23 W/cm^2. Even though heat is only generated in 5 of the 25 heater elements, significant lateral conduction causes the remaining 20 heater locations to also experience positive heat fluxes ranging from 0.29 W/cm^2 to 2.73 W/cm^2 for a power input of 32.4 W, with the value depending on distance from the heated elements. The heat flux transferred to the fluid along the flow length for increasing input power levels is shown in Figure 6.5a.

The wall temperature reaches a maximum at the central transverse strip of powered heater elements. The input power is incremented until the maximum temperature reaches 136.9 °C; further increases would damage the test chip and solder joint. The measured wall temperatures along the flow length for increasing input power levels are shown in Figure 6.5b. The wall temperatures downstream of the activated heater elements are higher than at the upstream elements; a difference of 7.93 °C exists between the inlet and outlet at a power input of 32.4 W due to the temperature rise of the bulk fluid. The maximum bulk fluid temperature is calculated to be 99.1 °C (the local saturation temperature at the measured pressure); the largest bulk fluid temperature gradient is observed as fluid flows over the hotspot.

Boiling curves are constructed from the heat flux transferred to the fluid and the wall excess temperature, and are shown in Figure 6.6 for sensors 3, 13, and 23. The wall excess temperature is calculated with respect to the local bulk fluid temperature in the case of single-phase flow, and the saturation temperature in two-phase operation. As the heat flux is increased, the slope of the curve is initially constant, reflecting the relatively constant single-phase heat transfer coefficient. For the upstream and downstream sensors, which are not actively powered, the heat flux is initially negative because the fluid is hotter than the wall and transfers heat to the substrate; this continues until a higher power input is reached and the active strip of heaters spreads heat to these locations. Boiling begins at the heated sensor location at a local heat flux of 16.8 W/cm^2 and a 38.2 °C excess temperature, and is indicated by the increased slope in the boiling curve. This incipience of boiling is confirmed via *in situ* visualization. Lower power input levels produced bubbly flow while and increased power input led to slug flow. At the largest power input, large vapor regions can be seen. All visualizations shown herein have a field of view that captures the boiling behavior over the entire test chip. After boiling incipience occurs at sensor 13, the downstream wall temperature at sensor 23 decreases; the increased (two-phase) heat transfer coefficient at the heated sensor location draws a larger percentage of the heat out of the center and keeps it from spreading by conduction to the outlet.

6.3.1.2 Case 1b (Central Streamwise Hotspot)

The next heating profile tested was with a central streamwise hotspot, with only the five streamwise heater elements located along the center of the heat sink powered. In the streamwise direction, the largest heat flux occurs at the inlet; while the heat flux has a local peak at the location of boiling, the global maximum occurs at the inlet due to entrance effects. As in Case 1a, there is significant lateral conduction through the chip, and the remaining 20 sensors have small positive heat fluxes ranging from 0.80 W/cm^2 to 4.37 W/cm^2 at a total power input of 25.6 W. The heat flux to the fluid is plotted across the central transverse temperature sensors for increasing power input levels in Figure 6.7a. The trends in the flow direction along the single strip of active heaters closely resemble the uniform heating trends presented later in this chapter for Case 2b; however, significant differences are observed transverse to the flow direction.

As the power input increases, the wall temperature is always highest at the hotspot. Along the hotspot in the streamwise direction, the highest wall temperature occurs at the outlet, as would be observed in a uniform heating case under similar conditions. The wall temperatures measured across the central transverse sensors at increasing input power levels to the hotspot elements are plotted in Figure 6.7b. The maximum allowable operating temperature in the chip is reached at a total power input that is lower by 26.6% for the streamwise hotspot compared to the transverse hotspot, due to the bulk fluid temperature increase along the flow length in the streamwise case. Along the hotspot, the fluid temperature reaches the saturation temperature roughly halfway along the flow length.

Boiling curves (wall heat flux versus excess temperature at the wall) are shown in Figure 6.8 for sensors 3, 13, and 23. Up to a total power input of 18.4 W, the streamwise hotspot channels exhibit single-phase operation. The steeper slope for the inlet sensor in the boiling curve is attributed to entrance effects. Boiling only occurs in the hotspot channels, and begins at a heat flux of 8.80 W/cm^2 and a wall excess temperature of 26.9 °C at the outlet. As the power level increases, the location of incipience of boiling advances closer to the inlet. As this occurs, the heat flux transferred to the fluid decreases at the outlet (while the wall temperature upon dryout continues to increase) due to conduction spreading toward the lower temperature upstream area.

6.3.1.3 Case 1c (Inlet Transverse Hotspot)

A transverse hotspot at the inlet, with the first row of elements activated, is considered net. The heat flux to the fluid is plotted across the central streamwise column at increasing power input levels in Figure 6.9a. It can be seen that 98% of the input heat is transferred to the fluid over the heated length, which is the first 2.54 mm, or 20% of the total flow length (compared to 77.6% for the centrally located heated length in Case 1a). There is less heat spreading in this case compared to Case 1a due to the absence of an upstream flow length to contribute to heat spreading. In addition, the fluid reaches the saturation temperature near the inlet, rendering the downstream portion of the heat sink less effective. This reduces heat spreading to the downstream locations. The flow length downstream of the hotspot is longer than in Case 1a, allowing the outlet wall temperature to decrease below the fluid saturation temperature. The wall temperatures measured along the central streamwise temperature elements with increasing power input levels are shown in Figure 6.9b. Boiling curves of the wall heat flux versus the excess wall temperature are shown in Figure 6.10 for sensors 3, 13, and 23. Boiling begins at the inlet hotspot at a heat flux of 23.1 W/cm^2 and a wall excess temperature of 42.5 ˚C. As in Case 1a, lower power input levels produce bubbly flow while an increased power input leads to slug flow. At higher power levels, long slugs of vapor form at the hotspot and begin to condense at the outlet.

6.3.2 Case 2: Non-Uniform Peak Heating

The degree of nonuniformity imposed in the distribution of a given total input power to different portions of the chip is quantified by comparing the amount of peak heating to the background heating through the parameter

$$\Phi = \frac{\frac{Q_{in,high}}{N_{high}} - \frac{Q_{in,low}}{N_{low}}}{\frac{Q_{in,total}}{N_{high}}}, \tag{6.10}$$

where Q_{in} refers to the total power input to the heater elements in a region, and N refers to the number of heater elements in that region. The subscripts $high$ and low refer to the heater element regions at peak and background power inputs, respectively. With this definition, a uniform heating case gives Φ = 0 while a hotspot case gives Φ = 1.

6.3.2.1 Case 2a (Non-Uniform Transverse Peak)

For the central transverse peak heating case, 17 discrete Φ values were imposed at an average constant total input power level of 33.0 W. The total power input for each of the peak heating cases studied, along with the maximum local heat fluxes for Φ = 1 (hotspot) and Φ = 0 (uniform heating), are summarized in Table 6.2. The heat flux to the fluid over the flow length for increasing Φ values is shown in Figure 6.11a. As the difference between the peak and background heater power levels increases (at a constant total power input), the heat flux to the fluid increases at the central transverse heater elements. The heat flux upstream of the transverse peak-heated strip is greater than that downstream due to the higher heat transfer coefficient at the inlet.

As the degree of nonuniformity Φ increases, the highest wall temperatures are seen along the transverse central heater elements; however at very low values of Φ, the wall temperature is highest at the outlet as would be expected for a uniform heating case with an increasing streamwise temperature for a single-phase fluid. The wall temperatures measured along the flow length for increasing Φ values are shown in Figure 6.11b. The maximum wall temperatures range from 128.3 °C for a uniform case to 136.5 °C for Φ = 1, and occur at different locations. In a uniform case the maximum wall temperature is located at the outlet, while for Φ =1 the maximum wall temperature is located above the peak heater element.

The heat transfer coefficient was also calculated along the flow length, and is shown in Figure 6.11c. As the input power nonuniformity Φ increases, the heat transfer coefficient above the peak-heated region increases. For nonuniformities with Φ > 0.38, the highest heat transfer coefficient is observed at the transverse central heater elements, where boiling occurs locally. At the central heater element (sensor 13), the heat transfer coefficient ranges from 1870 W/m^{2}K for a uniform case to 5970 W/m^{2}K at Φ =1. Boiling does not occur at the inlet for any of the Φ values investigated, and therefore the heat transfer coefficient remains unchanged at the upstream locations (sensors 1-10). Once vigorous boiling starts above the heated strip, the heat transfer coefficient at the outlet sees a significant drop. This is similar to the effect seen in the corresponding hotspot case, Case 1a. At large Φ values, more effective heat transfer at the heat sensor locations reduces the heat available for spreading to the outlet, reducing the local wall temperature and heat flux in the outlet region, but maintaining a high fluid temperature due to upstream boiling.

Figure 6.12 plots the heat transfer coefficient as a function of temperature difference between the wall and fluid for sensors 3, 13, and 23. Boiling occurs at the peak transverse heat input locations for all Φ values greater than zero. Therefore, as the surface temperature increases with increasing Φ, the heat transfer coefficient at sensor 13 increases significantly, as expected for a boiling regime. The single-phase heat transfer coefficient at the inlet sensor remains relatively constant. The heat transfer coefficient at the outlet sensor increases for the early part of the increase in Φ, and subsequently decreases, even as the wall excess temperature continually decreases. The increase at low Φ values occurs because of the relatively constant heat flux transferred to the fluid at that location due to heat spreading, coupled with a decrease in the difference between the wall and fluid temperatures. For large Φ values, the heat transfer coefficient reduction is likely due to a combination of a reduced heat flux (brought about by reduced heat spreading) and a high local vapor quality at the outlet.

High speed images extracted from videos at different degrees of nonuniformity Φ are shown in Figure 6.13 for a central transverse peak-heating profile. The images are extracted from videos recorded at 10,000 frames per second with a shutter speed of 10 kHz. In the figure the degrees of nonuniformity of 0.15, 0.38, 0.66, and 1.0 are shown; a significant difference in the number of active boiling channels can be seen over this range. For Φ = 0.15, boiling does not occur in all of the channels, and some channels display more vigorous boiling than others. As the local heat flux increases, boiling is observed in more of the channels for Φ = 0.38, and in all of the channels for Φ = 0.66 and for Φ = 1. Additionally, the location of boiling incipience moves toward the peak-heated sensors as Φ increases.

6.3.2.2 Case 2b (Non-Uniform Streamwise Peak)

Non-uniform peak heating in the orthogonal direction is tested in Case 2b with a central streamwise peak. Fifteen discrete Φ values were imposed with a constant total input power of 24.4 W. As with the transverse peak, the heat transferred to the fluid peaks at the central streamwise heater elements as the difference between the peak and background heater power Φ increases. The heat flux to the fluid in the central transverse heater elements for increasing Φ values is shown in Figure 6.14.

As Φ increases, the wall temperature becomes highest at the central heater elements, and increases in the streamwise direction. The maximum wall temperature ranges from 121.2 °C for a uniform case to 138.7 °C for Φ =1. The wall temperatures measured along the flow length for increasing Φ values are shown in Figure 6.15a. At low values of Φ, the wall temperature continually increases from inlet to outlet, indicating single-phase heat transfer. As Φ increases to 0.17, boiling occurs near the outlet and the wall temperatures for the last two sensors become constant while those near the inlet continue to rise. As the location of boiling advances toward the inlet (Φ = 0.61), dryout conditions occur at the outlet and the outlet wall temperature begins to rise again.

As the degree of nonuniformity Φ is increased, the associated spatial variation of heat transfer coefficient yields insights into the underlying heat transfer mechanisms. The heat transfer coefficients along the flow length are shown for increasing Φ values in Figure 6.15b. Initially, for uniform heating conditions (Φ =0), the flow remains entirely in the single-phase regime along the entire channel length. The heat transfer coefficient is greatest at the inlet (3870 W/m^{2}K) due to entrance effects, and asymptotically decreases to a fully-developed, constant value (1290 W/m^{2}K). The relative magnitude of this entrance-effect enhancement is similar to that observed in a previous study [93] for uniform heating conditions. With an increase in Φ to 0.17, the upstream trend remains similar; however, boiling incipience occurs near the outlet, and the heat transfer coefficient increases at this location. As Φ increases further, the location of boiling incipience advances upstream, and the associated heat transfer coefficient increase propagates in the same direction. Ultimately, boiling occurs at the inlet, and a maximum heat transfer coefficient of 4440 W/m^{2}K is observed at this location for $\Phi = 1$. At the outlet, while the heat transfer coefficient initially increases as boiling occurs and moves upstream, the heat transfer coefficient decreases as the nonuniformity reaches $\Phi = 0.61$. This is indicative of partial dryout in the downstream ends of the central channels.

The heat transfer coefficient is shown as a function of the wall excess temperature in Figure 6.16 for sensors 3, 13, and 23. In this case, boiling begins at the outlet at a low value of Φ and the location moves upstream at higher Φ values. As the location of boiling moves upstream, the heat transfer coefficient at the middle sensor increases sharply. The heat transfer coefficient at the outlet sensor peaks and then begins to decrease at higher Φ values as the more effective boiling incipience regime moves upstream. Boiling only occurs at the central strip of streamwise heater elements shown in the figure.

Images extracted from high-speed videos at different degrees of nonuniformity Φ are shown in Figure 6.17 for a central streamwise peak-heating profile; degrees of nonuniformity of 0.01, 0.23, 0.61, and 1.0 are shown. Boiling does not occur in all of the channels for all values of Φ. At high Φ values, significant flow reversal can be seen in the central channels above the peak-heated sensors, causing flow maldistribution in the heat sink and partial dryout at the outlet. At these large values the heated channels contain a very large amount of vapor while neighboring channels exhibit bubbly flow. Boiling in the channels associated with the peak-heated elements causes an increase in the local pressure drop, forcing both liquid and vapor bubbles back into the inlet manifold. Vapor in the inlet manifold reroutes to channels with lower flow resistance where little or no boiling occurs. A reduced flow rate in the channels above the peak-heated sensors causes the remaining liquid to vaporize entirely, causing partial dryout. Once a significant amount of vapor leaves the channel through the outlet, the pressure equalizes, liquid flows back into the channels above the peak-heated sensors, and the process repeats.

6.4 Conclusions

In this chapter the effects of non-uniform hotspots and heating profiles in a microchannel heat sink on heat transfer coefficients, wall temperatures, and the location of boiling incipience were investigated. To properly assess the local heat dissipation under non-uniform heating conditions, lateral conduction through the microchannel heat sink base was taken into account. Experimental results show that even with a very thin substrate, significant lateral conduction occurs in the base of the heat sink.

Single hotspots that span the width or length of a silicon microchannel heat sink were investigated as a function of increasing local heat flux. In the case of a transverse hotspot in the center of the heat sink, once boiling begins in the heated sensor location, the wall temperature at the outlet decreases and conduction away from the center is mitigated due to reduced convection thermal resistance. In the case of a streamwise hotspot along the central column of the heat sink, conduction causes some lateral heating, but boiling *only* occurs in the channels located above the hotspot. In this configuration, the maximum sustainable total power input achieved is reduced by 26.6% compared to the transverse hotspot case. In the case of a transverse hotspot located at the inlet, although the maximum sustainable total power input is similar to the central transverse hotspot, the local maximum heat flux is increased by 35.7% as a result of significantly reduced upstream heat spreading. These test cases show that the same total power input distributed in different locations and configurations across the heat sink can cause significantly different limits on the maximum heat fluxes and wall temperatures that can be supported.

A second non-uniform heating condition was investigated to understand the effect of the degree of nonuniformity imposed in the distribution of a given total input power to different portions of the chip, by incrementing the nonuniformity between the peak and background heat flux values. For non-uniform transverse peak-heating profiles, an increase in the heating nonuniformity results in significant boiling at the location of the peak heat input, whereas no boiling occurs under uniform heating conditions. For non-uniform streamwise peak heating profiles, an increase in the heating nonuniformity for a constant total power input results in boiling at the location of the peak heat input location; the location of boiling incipience moves upstream as the nonuniformity increases. For both hotspot and peak heating in the streamwise direction, significant flow reversal is observed leading to dryout in the channels above the peak heated region. In both cases, the local heat transfer coefficients and wall temperatures deviate significantly from a uniformly heated case. Local heat flux concentrations result in high local two-phase flow heat transfer coefficients, but at the expense of increased wall temperatures.

Table 6.1. Summary of results for the hotspot heating cases.

	Maximum total power input (W)	Maximum local heat flux (W/cm^2)	Maximum local wall temperature (˚C)
Case 1a: Central transverse hotspot	32.4	24.23	136.9
Case 1b: Central streamwise hotspot	25.6	16.14	146.3
Case 1c: Inlet transverse hotspot	35.8	32.89	138.8
Case 1d: Dual transverse hotspots	65.0	32.21	133.7

Table 6.2. Summary of results for the peak heating cases.

	Total power input (W)	Maximum local heat flux, $\mathbf{\Phi}$=1 (W/cm^2)	Maximum local heat flux, $\mathbf{\Phi}$=0 (W/cm^2)
Case 2a: Non-uniform transverse peak	33.0	23.70	8.17
Case 2b: Non-uniform streamwise peak	24.4	15.29	6.23

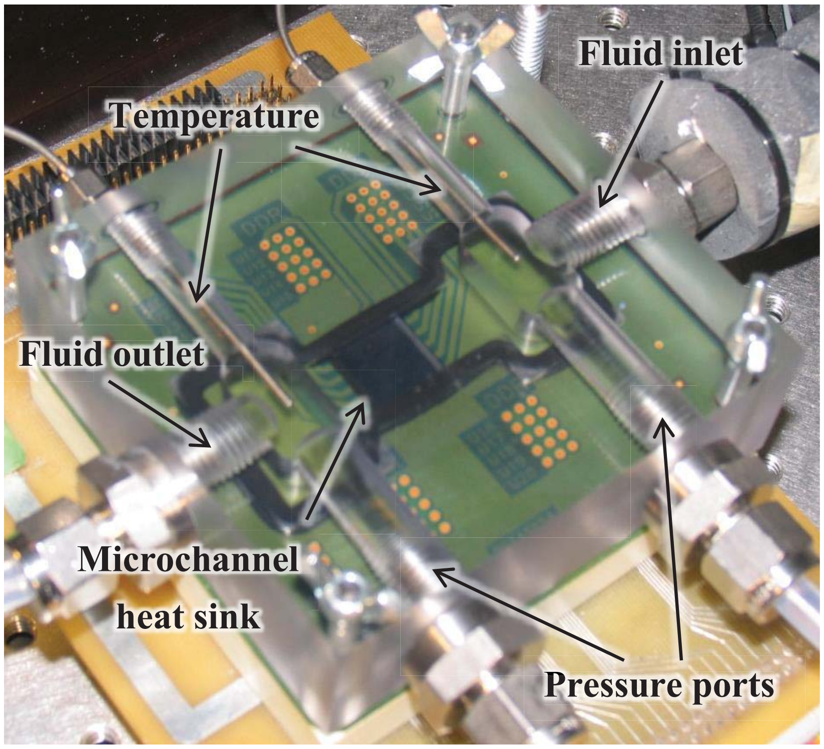
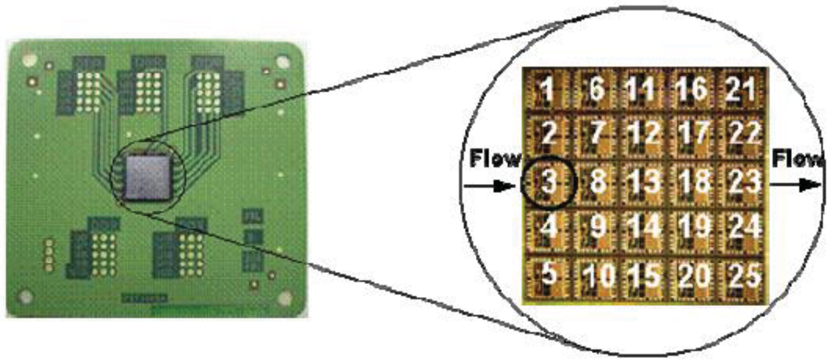


Figure 6.1. Image of the microchannel test section.



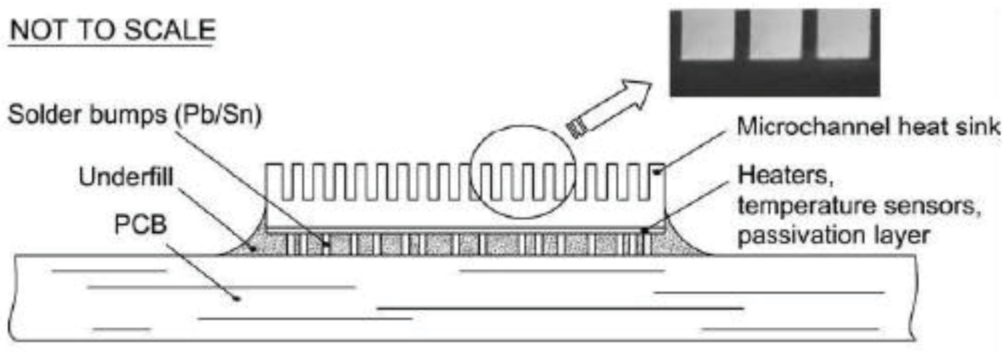


Figure 6.2. Images of the 5 × 5 array of heater elements and a schematic diagram of the microchannel heat sink.

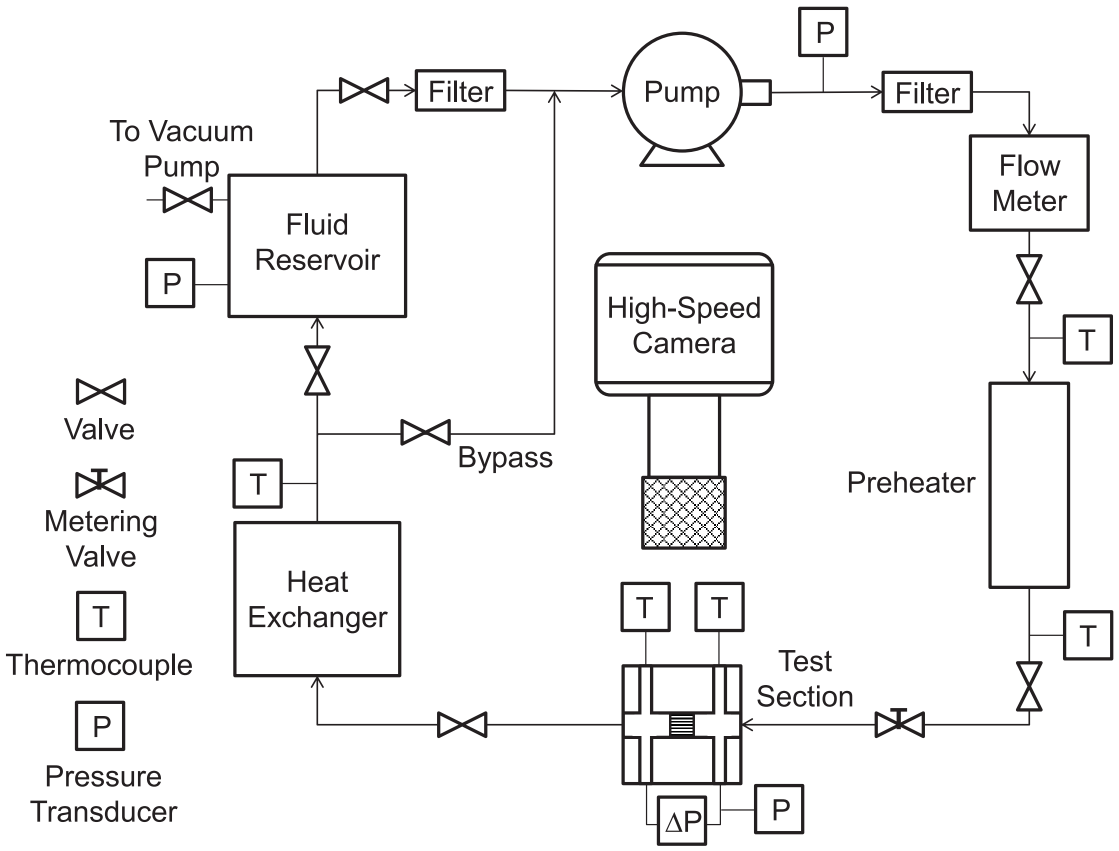


Figure 6.3. Schematic diagram of the experimental setup showing the flow loop components and high-speed visualization optics.

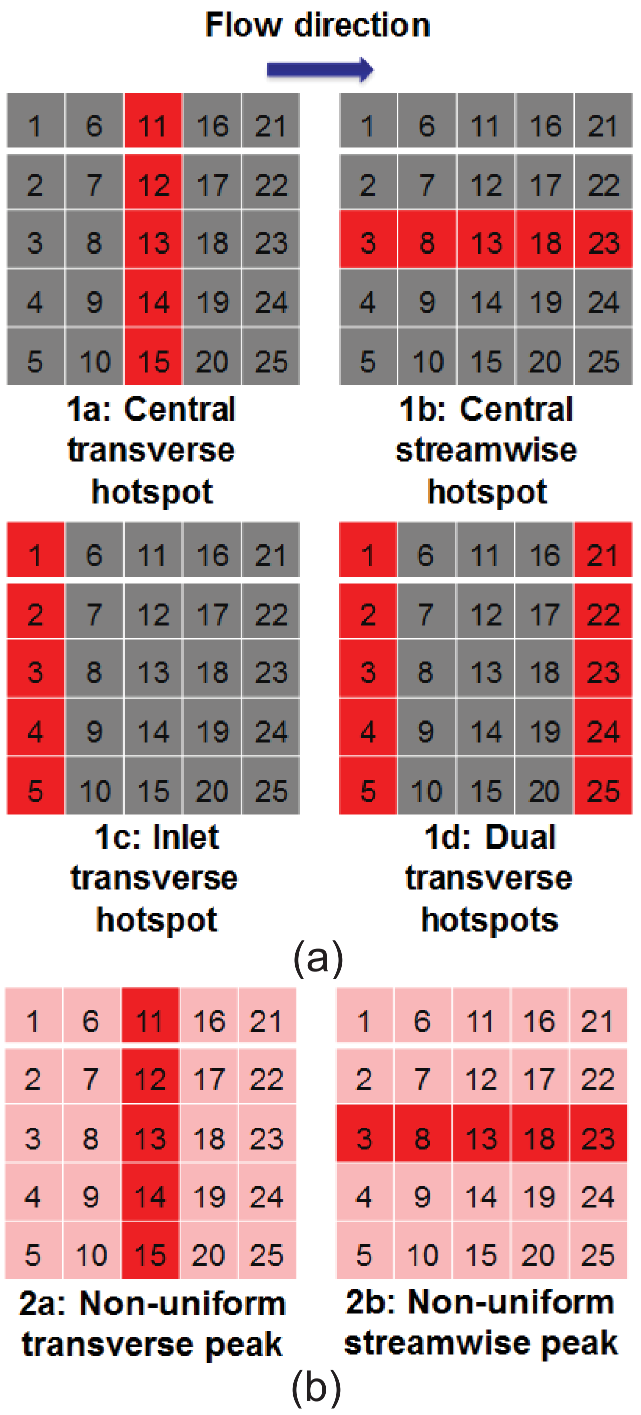

Flow direction
1a: Central transverse hotspot
1b: Central streamwise hotspot
1c: Inlet transverse hotspot
1d: Dual transverse hotspots
(a)
2a: Non-uniform transverse peak
2b: Non-uniform streamwise peak
(b)

Figure 6.4. (a) Hotspot, and (b) non-uniform peak-heating profile configurations investigated.

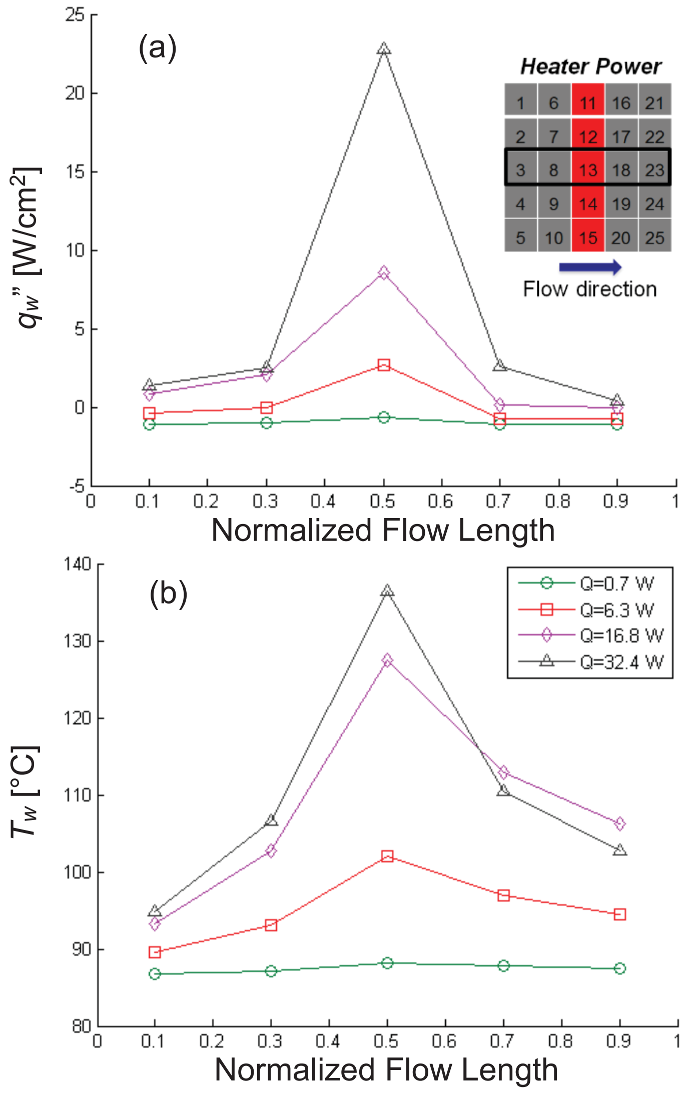


Figure 6.5. (a) Local heat flux transferred to the fluid, and (b) wall temperature along the flow length at increasing power input levels for a central transverse hotspot. The local quantities are presented for the central streamwise elements, as indicated by the dark black rectangle in the heater power diagram.

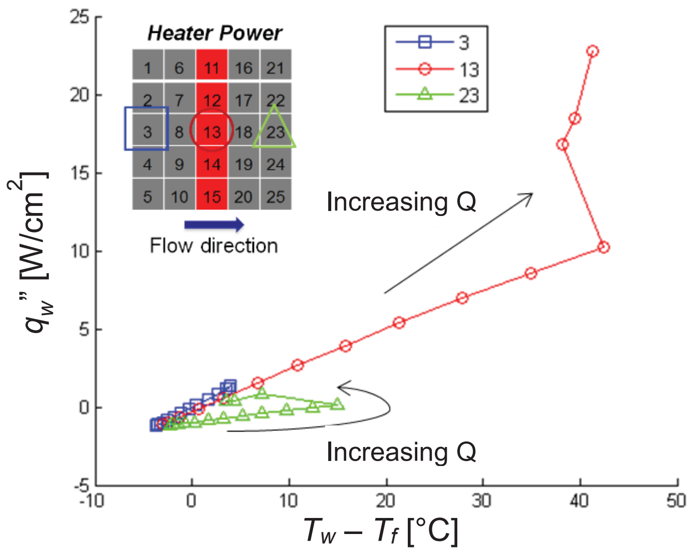


Figure 6.6. Heat flux transferred to the fluid plotted against the wall excess temperature for sensors 3, 13, and 23 for a central transverse hotspot.

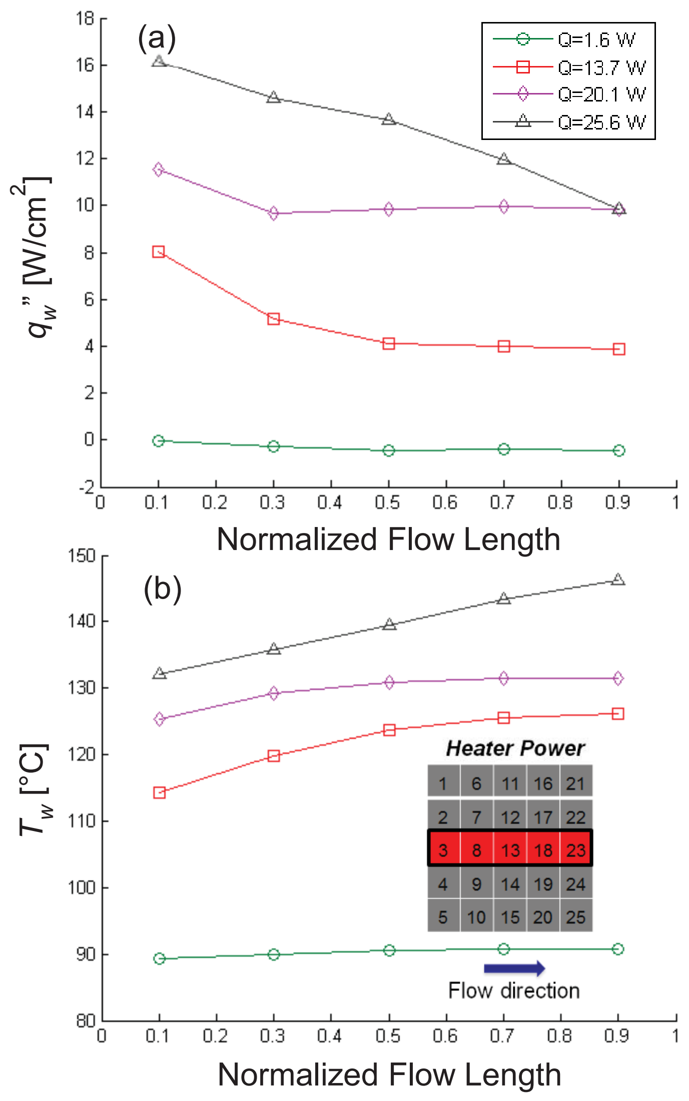


Figure 6.7. (a)Local heat flux transferred to the fluid, and (b) wall temperature over the width of the chip for increasing power levels for a central streamwise hotspot. The local quantities are presented for the transverse elements, as indicated by the black line on the heater power diagram.

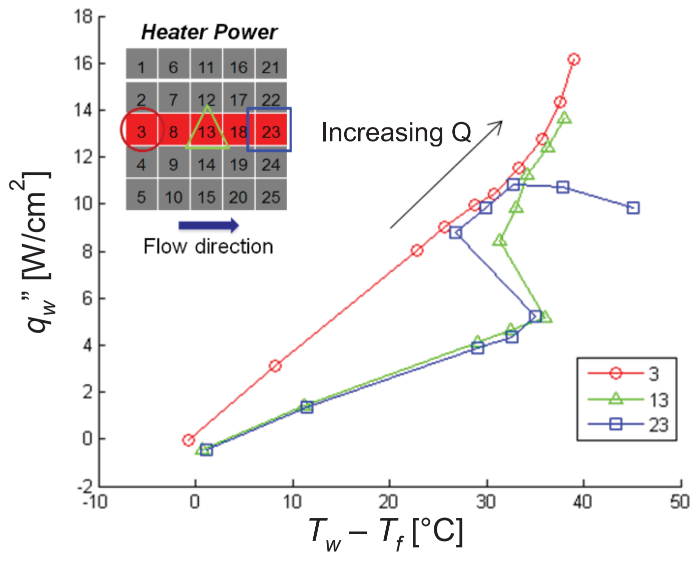


Figure 6.8. Heat flux transferred to the fluid plotted against the wall excess temperature for sensors 3, 13, and 23 for a central streamwise hotspot.

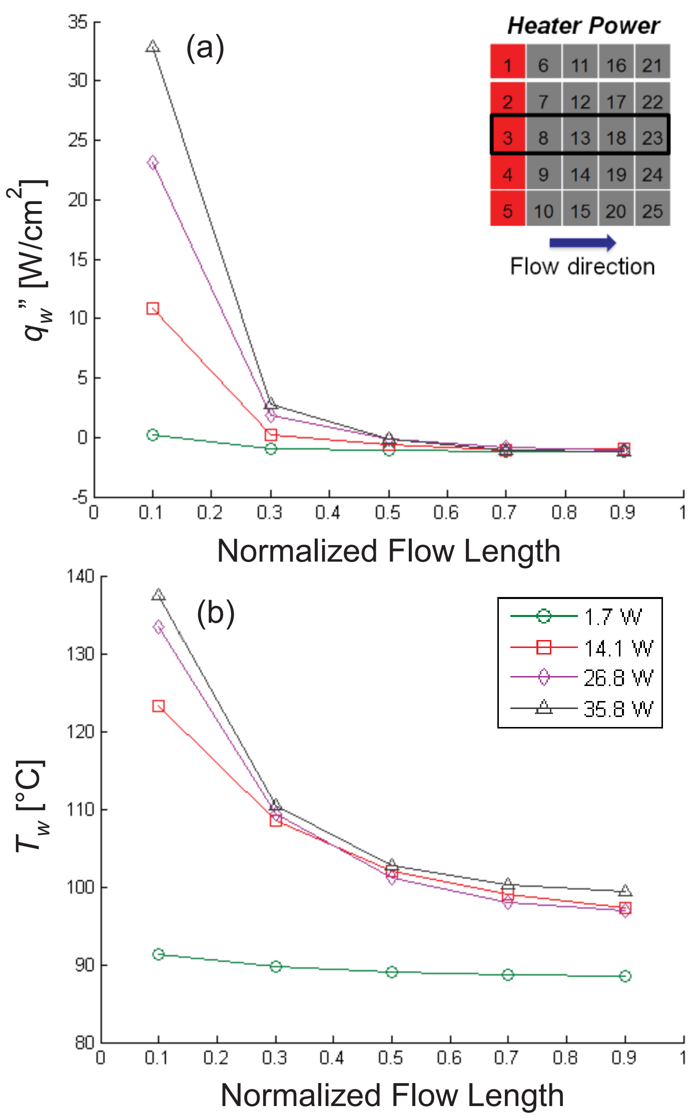


Figure 6.9. (a) Local heat flux transferred to the fluid, and (b) wall temperature over the length of the chip for increasing power input levels for an inlet transverse hotspot. The local quantities are presented for the streamwise elements, as indicated by the black line on the heater power diagram.

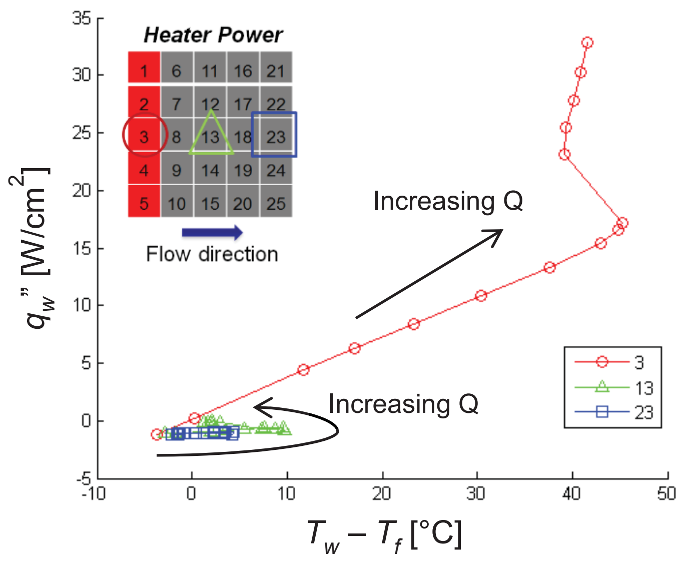


Figure 6.10. Heat flux transferred to the fluid plotted against the wall excess temperature for sensors 3, 13, and 23 for an inlet transverse hotspot.

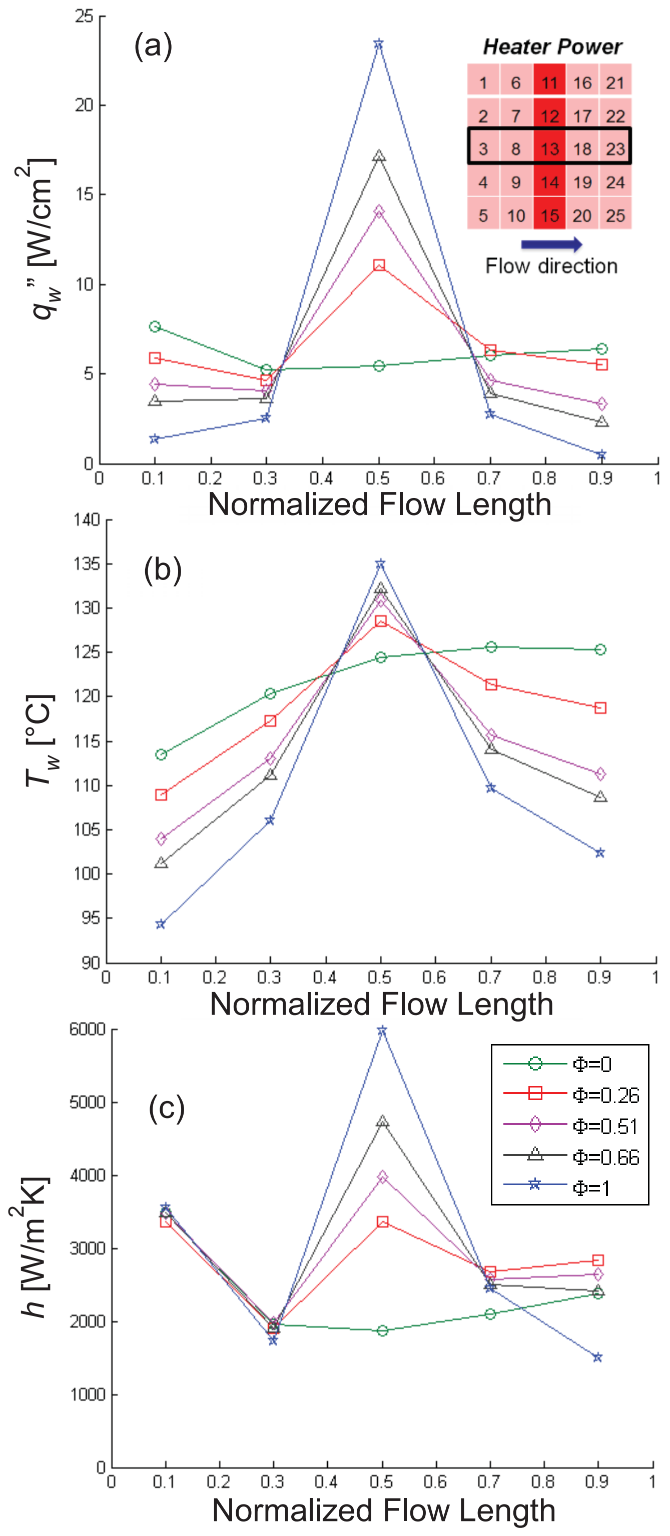


Figure 6.11. (a) Local heat flux transferred to the fluid, (b) wall temperature, and (c) heat transfer coefficient over the flow length at increasing degrees of nonuniformity between the heat flux at the peak and the background heater locations for Case 2a.

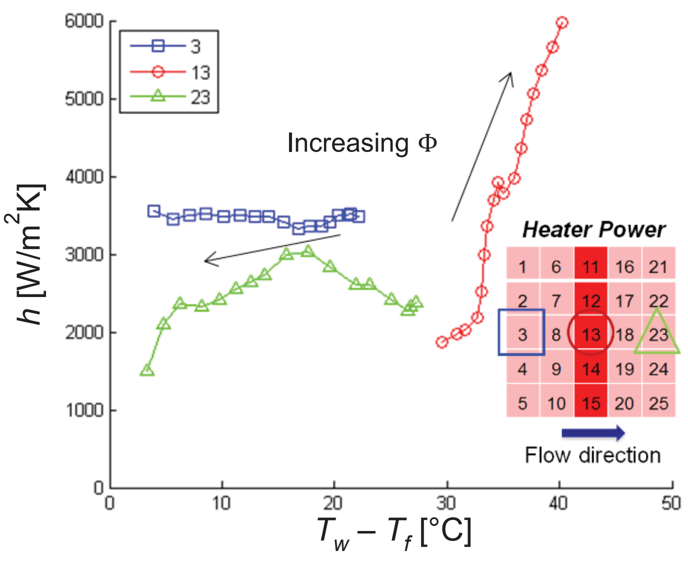


Figure 6.12. The heat transfer coefficient as a function of excess wall temperature for sensors 3, 13, and 23 for Case 2a.

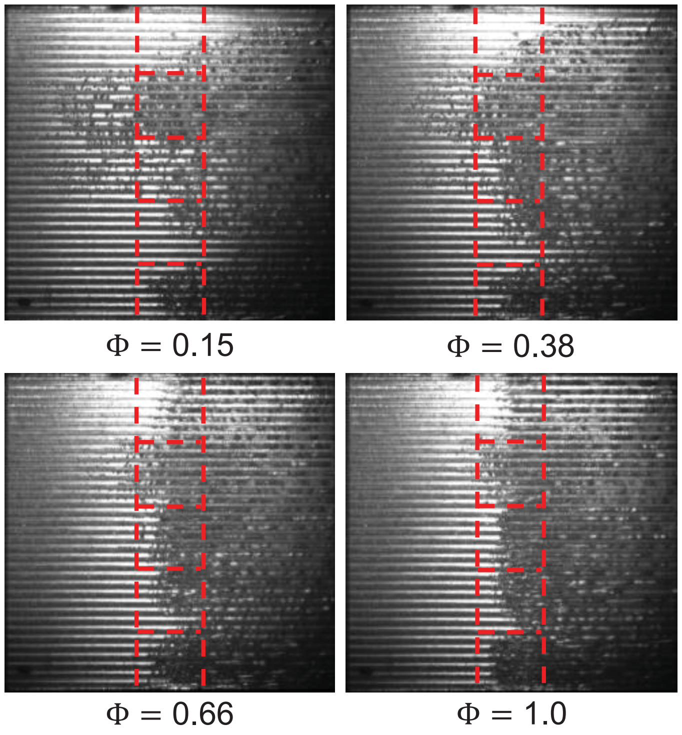


Figure 6.13. Images at increasing Φ values for a central transverse peak extracted from high-speed video. Red lines indicate the locations of the peak heated sensors.

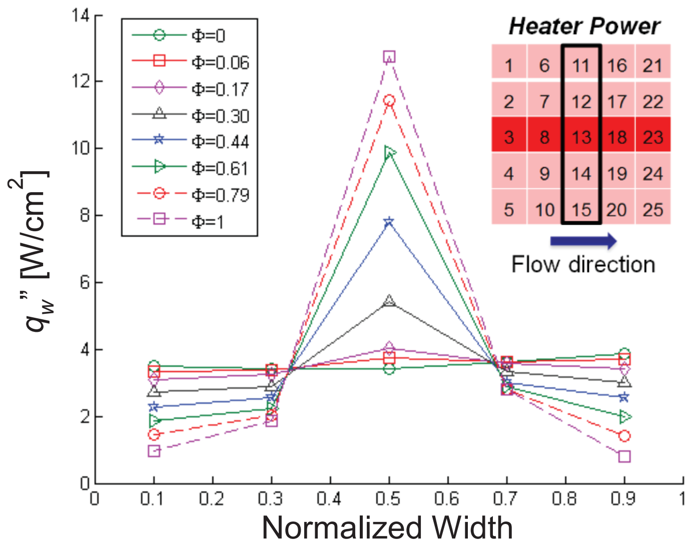


Figure 6.14. The local heat flux transferred to the fluid over the width of the chip at increasing degrees of nonuniformity between the heat flux at the peak and background heater location for Case 2b.

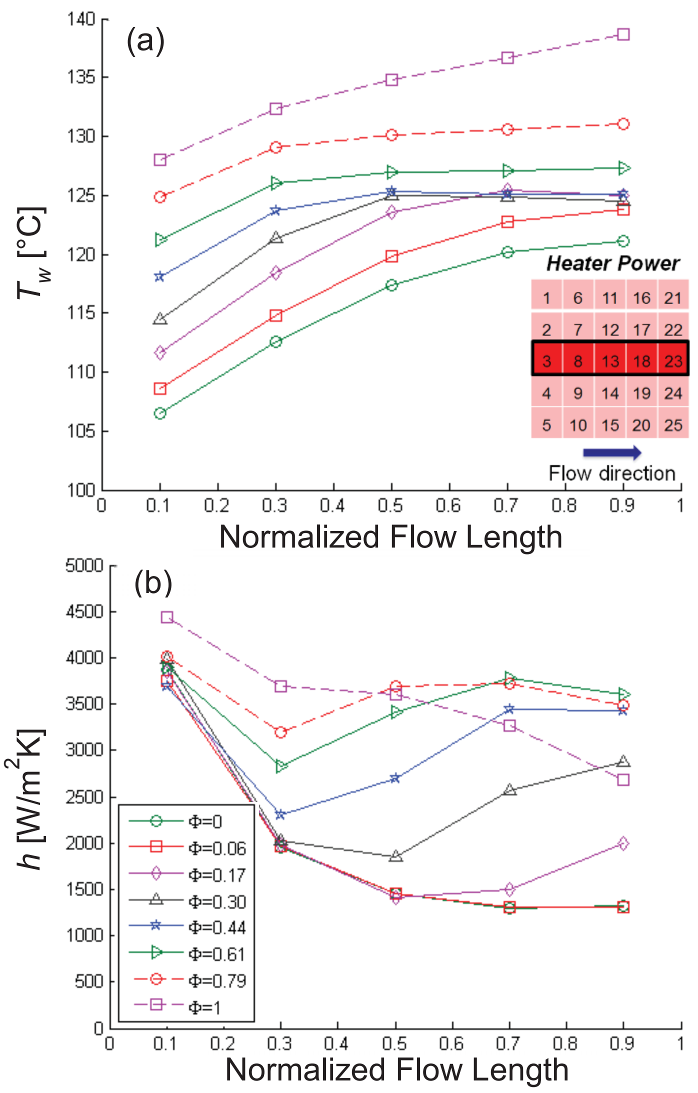


Figure 6.15. (a) Local wall temperature, and (b) heat transfer coefficient over the flow length at increasing degrees of nonuniformity between the heat flux at the peak and background heater locations for Case 2b.

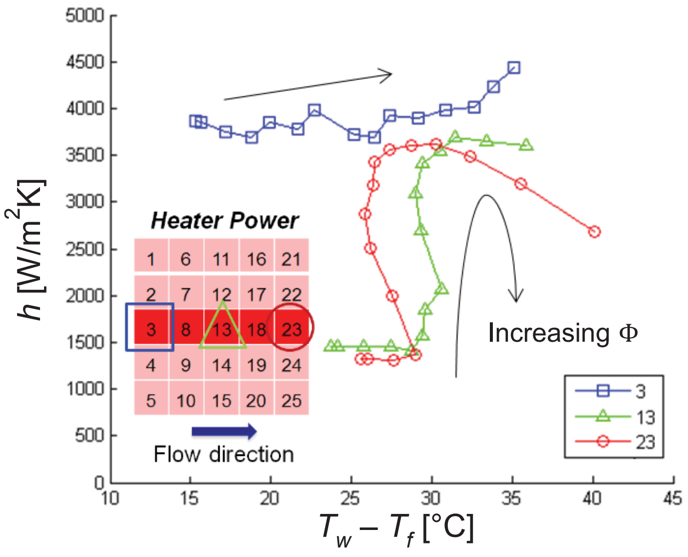


Figure 6.16. The heat transfer coefficient plotted against the wall excess temperature for sensors 3, 13, and 23 for Case 2b.

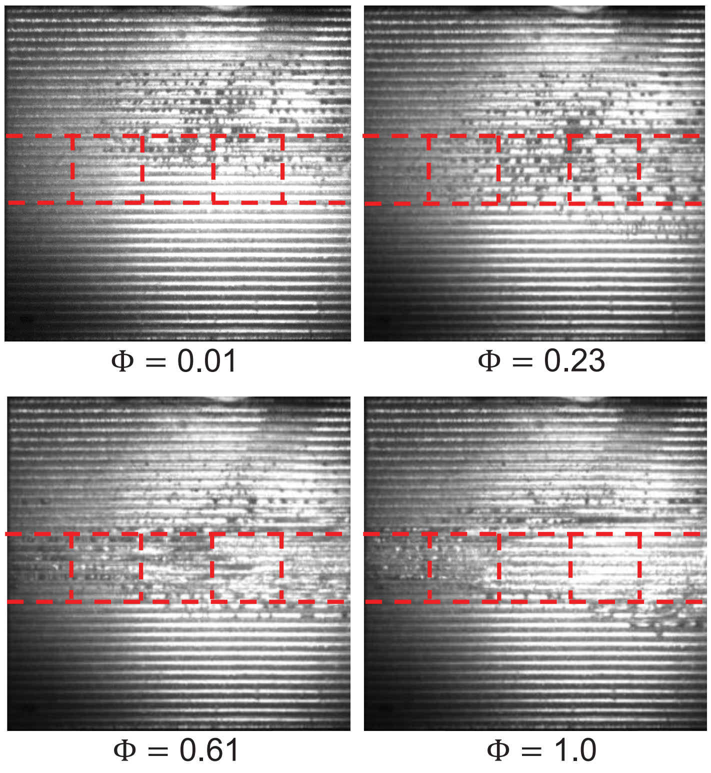


Figure 6.17. Images at increase Φ values for a central streamwise peak extracted from high-speed video. Red lines indicate the locations of the peak heated sensors.

CHAPTER 7. COMPUTATIONAL MODEL TO PREDICT NON-UNIFORM HEATING RESULTS

The effects of non-uniform heating cases other than those presented in the previous chapters on the thermal performance of a microchannel heat sink is not well known. Conducting experiments using a large number of non-uniform heating cases to study the effects on local wall temperatures and heat transfer coefficients is both time consuming and cumbersome. A simplified computational model that can predict the effects in a microchannel heat sink allows rapid analysis of multiple different cases. This chapter presents a computational model that was developed to predict the results obtained from experiments presented in CHAPTER 6. The model includes three-dimensional (3D) conduction in the base of the microchannel heat sink, as well as a fin analysis, to determine the local temperatures and heat fluxes throughout the domain. The results are then compared to the previously obtained experimental results.

7.1 Modeling and Simulation

A MATLAB [74] code was written in-house to model non-uniform heating in a microchannel heat sink and can be found in Appendix G. The author would like to thank Professor Tine Baelmans of KU Leuven for providing some of the logic in the code. A flow chart of the model algorithm is shown inFigure 7.1. First, the user inputs the parameters for a case in a graphical user interface (GUI). This includes the dimensions of the microchannel heat sink, flow conditions, and a heating profile. Next, the code generates a mesh for the conduction domain inside the base of the heat sink. After the mesh is generated and the variables are initialized, the code loops through calculations of conduction in the base, pressure drop in each channel, local heat transfer coefficients as determined from correlation, local wall heat fluxes, and local quality. The code then checks the maximum change percentage of the temperature in the domain to see if the solution has converged. Otherwise, it uses the calculated values to initialize a new iteration. After the solution has converged, the code displays the output using a GUI. Images of the input and output GUIs are shown in Figure 7.2.

7.1.1 Numerical Methods

The computational model was split into several subfunctions that each perform different calculations. The portion of the model that calculates 3D conduction in the substrate uses an energy balance on each cell to calculate the local temperatures in the domain. Fourier's Law is used to calculate the conduction between cells. The cells on the bottom surface (z = 0) have a heat flux imposed based on the heating profile specified. Heat loss from the heat sink is also imposed on these cells. The heat loss from the heat sink is calculated based on local temperatures using a correlation derived from experimental data calculated as

$$\dot{q}_{loss} = aT + b. \tag{7.1}$$

The cells on the side surfaces of the heat sink use a Neumann boundary condition with the heat flux equal to zero. The cells on the top surface are connected to the channels where they experience convection with the fluid, and to fins where they experience conduction through the channel walls. Values for the heat transfer coefficient from previous iteration are used to calculate the local wall heat flux in these cells.

The finite-volume method is used to discretize the energy balance equations used in the 3D mesh. The general equation for a cell in the domain is represented as

$$\begin{aligned} a_{i,j,k}T_{i,j,k} = b_{i,j,k}T_{i+1,j,k} + c_{i,j,k}T_{i-1,j,k} + d_{i,j,k}T_{i,j+1,k} + e_{i,j,k}T_{i,j-1,k} \\ + f_{i,j,k}T_{i,j,k+1} + g_{i,j,k}T_{i,j,k-1} + h_{i,j,k}, \end{aligned} \tag{7.2}$$

where $a_{i,j,k}$ is the coefficient of the cell in question, $b_{i,j,k}$ is the coefficient of the right neighbor, $c_{i,j,k}$ is the coefficient of the left neighbor, etc., and T is the temperature for each cell. $h_{i,j,k}$ is the source term representing the heat input, heat loss, or convection. For a cell on the bottom surface of the domain, this equation can be written and rearranged as

$$\begin{aligned} k\Delta \mathrm{y}\Delta \mathrm{z}\frac{T_P - T_R}{\Delta x} + k\Delta y\Delta \mathrm{z}\frac{T_P - T_L}{\Delta x} + k\Delta x\Delta z\frac{T_P - T_T}{\Delta y} + k\Delta x\Delta z\frac{T_P - T_M}{\Delta y} \\ + k\Delta x\Delta y\frac{T_P - T_F}{\Delta z} + k\Delta x\Delta y\frac{T_P - T_B}{\Delta \mathrm{z}} + aT_P - b + q_{in} = 0, \end{aligned} \tag{7.3}$$

where k is the thermal conductivity of silicon, Δx, Δy, and Δz are the dimensions of each cell, and q_{in} is the energy into the cell based on the heating profile.

The system of equations for the 3D domain is solved using a plane-by-plane Tri-Diagonal Matrix Algorithm (TDMA) method. The TDMA is a form of Gaussian elimination used to solve a tri-diagonal system of equations exactly [86]. This is the same method utilized in Section 4.1.

7.1.2 Heat Transfer Correlations

The local heat transfer coefficients are calculated using well knowncorrelations found in the literature that apply to internal flow in microchannels. The correlation by Lee and Garimella [5] was used for single-phase thermally developing flow and is calculated as

$$h_{sp} = \left[1.766\left(Re_f Pr_f \frac{D_h}{L}\right)^{0.378} \alpha^{0.1224}\right]\frac{k_f}{D_h}. \tag{7.4}$$

This correlation was used until the single-phase flow was determined to be thermally fully developed. The dimensionless thermal entrance length, z^*_{th}, was calculated with a correlation by Lee and Garimella [94] using the aspect ratio as

$$\begin{aligned} z^*_{th} = {} & -1.275 \times 10^{-6}\alpha^6 + 4.709 \times 10^{-5}\alpha^5 - 6.902 \times 10^{-4}\alpha^4 \\ & + 5.014 \times 10^{-3}\alpha^3 - 1.769 \times 10^{-2}\alpha^2 + 1.845 \times 10^{-2}\alpha \\ & + 5.691 \times 10^{-2} \end{aligned} \tag{7.5}$$

Once the single-phase flow was determined to be thermally fully developed, a correlation by Lee and Garimella [94] was employed and is calculated as

$$h_\infty = 8.235\frac{k_f}{D_h}\left(1 - \frac{2.0421}{\alpha} + \frac{3.0853}{\alpha^2} - \frac{2.4765}{\alpha^3} + \frac{1.0578}{\alpha^4} - \frac{0.1861}{\alpha^5}\right). \tag{7.6}$$

To determine when the flow through each channel reaches saturation, the local thermodynamic vapor quality was calculated at each axial location as

$$x_{i,j} = \frac{1}{h_{fg}}\left(\frac{\sum_{i,j} q_{i,1\rightarrow j}}{\dot{m}} - c_p(T_{sat} - T_{in})\right), \tag{7.7}$$

where $\sum_{i,j} q_{i,1\rightarrow j}$ is the sum of the heat transfer rate to the ith channel from the inlet to location j. Once $x_{i,j}$ is positive, the flow is considered to be in saturated two-phase flow.

A correlation by Bertsch *et al*. [95] was used for the saturated two-phase flow heat transfer coefficient and is calculated as

$$h_{FB} = h_{NB}(1-x) + h_{conv,tp}[1 + 80(x^2 - x^6)e^{-0.6Co}], \tag{7.8}$$

where h_{NB} is the nucleate boiling heat transfer coefficient from the Cooper correlation [96] and $h_{conv,tp}$ is the convective heat transfer coefficient. It is calculated using the vapor quality as

$$h_{conv,tp} = h_{conv,l}(1-x) + h_{conv,v}x. \tag{7.9}$$

The liquid and vapor convective heat transfer coefficients are calculated using the Hausen correlation [97] for developing laminar flow and given as

$$h_{conv,n} = \left(3.66 + \frac{0.0668\frac{D_h}{L}RePr}{1 + 0.04\left[\frac{D_h}{L}RePr\right]^{2/3}}\right)\frac{k_n}{D_h}, \tag{7.10}$$

where n denotes the liquid or vapor phase.

To prevent a large discontinuity in the heat transfer coefficients when moving between single- and two-phase flow regions, a smoothing function was used and is calculated as

$$h = h_{sp}(1-x^*) + h_{tp}x^*, \tag{7.11}$$

where h_{sp} is the heat transfer coefficient calculated for single-phase flow and h_{tp} is the heat transfer coefficient calculated for two-phase flow. A nondimensional weighting factor, x^*, is calculated as

$$x_j^* = \frac{x_j - x_{ONB}}{x_{sat} - x_{ONB}}, \tag{7.12}$$

where x_j is the current location in the channel, x_{ONB} is the location of the onset of nucleate boiling, and x_{sat} is the location where saturated boiling begins, which is determined to be where the vapor quality is positive. The superheat criteria to determine the location of the onset of nucleate boiling is calculated using the correlation by Liu *et al.* [4] as

$$\sqrt{T_w} - \sqrt{T_s} \geq \sqrt{\frac{2\sigma C}{\rho_v h_{fg}} \frac{q_w''}{k_f}}. \tag{7.13}$$

After the local heat transfer coefficients are calculated, the local wall heat flux is calculated using a fin analysis. The heat flux to the fluid for a fin is calculated as

$$q_w'' = \eta_o h\left(T_w - T_f\right), \tag{7.14}$$

where η_o is the overall surface efficiency calculated as

$$\eta_o = 1 - \frac{NA_f}{A_t}\left(1 - \eta_f\right), \tag{7.15}$$

which is based on the area of a single fin, A_f, the total heat transfer area, A_t, and the fin efficiency, η_f, calculated for an adiabatic tip as

$$\eta_f = \frac{\tanh mL}{mL}. \tag{7.16}$$

7.1.3 Flow Maldistribution

When non-uniform heating profiles in the streamwise direction are imposed on a microchannel heat sink, as is the case for a central streamwise hotspot, significant flow maldistribution is observed to occur between the channels in experiments. At high power input levels, flow reversal can even be seen in the channels above the hotspot. Cases are performed with and without considering flow non-uniformities to assess the amount of error introduced into the calculated results.

The flow maldistribution was estimated by image processing high-speed video frames of boiling in the heat sink. The videos used in the estimation are discussed in Section 6.3. A modified version of the MATLAB image processing code discussed in Section 3.2 was utilized for this task. First, each frame is rotated and cropped to the area of interest. Second, the background is subtracted and the gray-scale image is passed through a threshold to produce a negative image with increased contrast. Next, characteristic edges of the two-phase flow are detected using the Canny algorithm implemented within MATLAB [74] and a binary image is produced. For the image processing code to calculate a void fraction, the exact air-water interface needs to be resolved. However, to estimate the flow velocities, the exact liquid-vapor interface is not needed, but only characteristic shapes of the two-phase flow. Figure 7.3a shows an example of the characteristic shapes that were detected in one test case. The bubble velocity in each channel is estimated using a two-dimensional cross-correlation algorithm within MATLAB between frames. A minimum of 50 sequential frames is used and the average velocity among the frames is estimated. The average bubble velocity for the test case is shown in Figure 7.3b. Once the two-phase flow velocity in each channel is known, the mass flux is estimated. The remaining mass flux is assumed to be evenly distributed among channels that contain only liquid.

7.2 Results and Discussion

The results of the computational model were compared with the experimental results presented in Section 6.3. Three cases are discussed: a uniform heating profile, a central transverse hotspot, and a central streamwise hotspot. The central streamwise hotspot was tested with and without taking flow maldistribution into consideration while the other cases were tested without flow maldistribution.

The base temperatures, wall heat fluxes, and heat transfer coefficients for a uniform heating profile with a total power input of 33 W is shown in Figure 7.4. In a uniform case, the computational model underpredicts the base temperature and overpredicts the heat transfer coefficient, but the trends for all three values match those for the experiments. This result is indicative primarily of the accuracy of the heat transfer coefficient correlations, not the model itself; since all three of these variables are coupled, a correction in the heat transfer coefficient would ultimately improve the results for the base temperature.

For the central transverse hotspot case at a power input of 33 W the base temperatures, wall heat fluxes, and heat transfer coefficients are shown in Figure 7.5. The computational model matches the experimental results with good agreement. As in the uniform case, the heat transfer coefficient is overpredicted by the model while the base temperature is slightly underpredicted. Even though the correlations used to calculate the heat transfer coefficient were originally developed for a uniformly applied heat flux, the trends predicted by the model still match the experimental trends well.

The base temperatures, wall heat fluxes, and heat transfer coefficients for a central streamwise hotspot are plotted in Figure 7.6. The plots show the experimental data as well as the results from the computational model with and without flow maldistribution taken into account. For the flow maldistribution case, the middle 7 channels located directly above the hotspot used a mass flux of 75% of the average mass flux in a channel. The remaining 28 non-boiling channels used a mass flux of 106.25% of the average mass flux in a channel so that the total mass flux remained the same as in the case without flow maldistribution. Like the previous cases, the model predicts the trends well, however the base temperature is still underpredicted while the heat transfer coefficient is overpredicted. Accounting for decreased flow in the channels above the hotspot improves the results slightly. Additionally in this case, the flow maldistribution predicts the location of boiling incipience in the channel further upstream, as observed in the experiments.

7.3 Conclusions

In this chapter a simple computational model was developed to predict the thermal performance of a microchannel heat sink exposed to non-uniform heating profiles. The predictions from the model were compared to experimental data for four cases with reasonable agreement. While the model underpredicts the base temperatures and overpredicts the heat transfer coefficients; this is suspected to be due to the accuracy of the correlations for predicting the heat transfer coefficient under the current experimental conditions. Results for a central streamwise hotspot were compared with and without considering flow maldistribution. Taking into account flow maldistribution in the channels improved the match between the results from the model and experimental data, demonstrating that large amounts of flow maldistribution cannot be ignored. Even though the model does not fully capture flow instabilities within a microchannel heat sink, it is still a useful tool for predicting the effects of non-uniform heating on thermal performance.

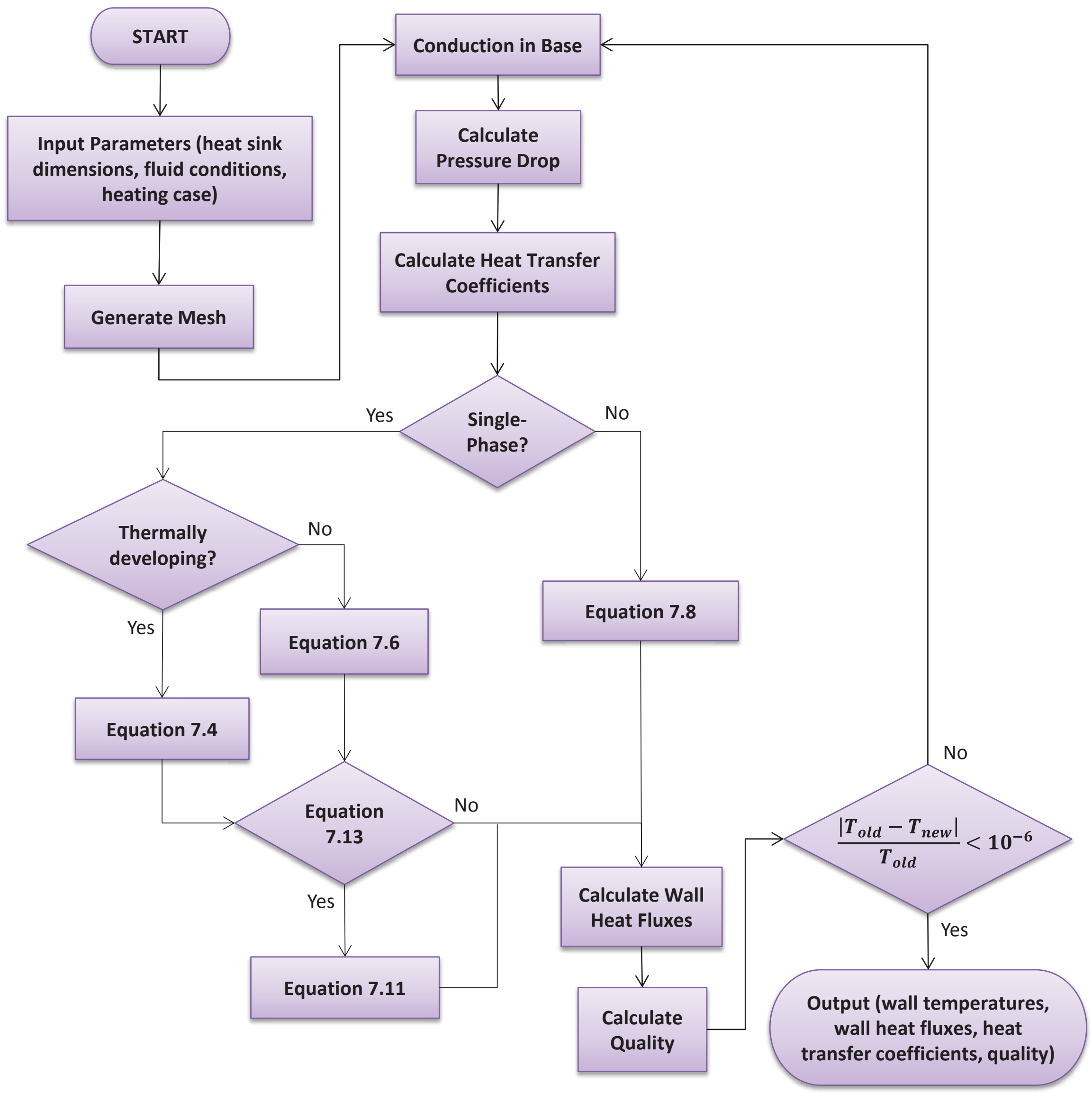
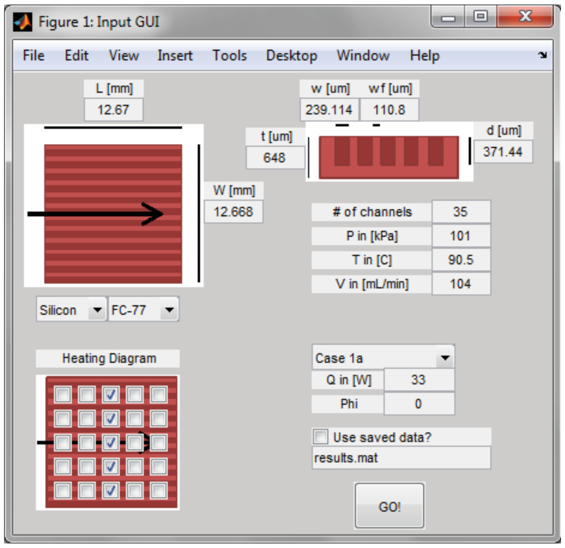

START
Input Parameters (heat sink dimensions, fluid conditions, heating case)
Generate Mesh
Conduction in Base
Calculate Pressure Drop
Calculate Heat Transfer Coefficients
Single-Phase?
Yes
No
Thermally developing?
No
Yes
Equation 7.6
Equation 7.4
Equation 7.8
Equation 7.13
No
Yes
Equation 7.11
Calculate Wall Heat Fluxes
Calculate Quality
$\frac{|T_{old} - T_{new}|}{T_{old}} < 10^{-6}$
No
Yes
Output (wall temperatures, wall heat fluxes, heat transfer coefficients, quality)

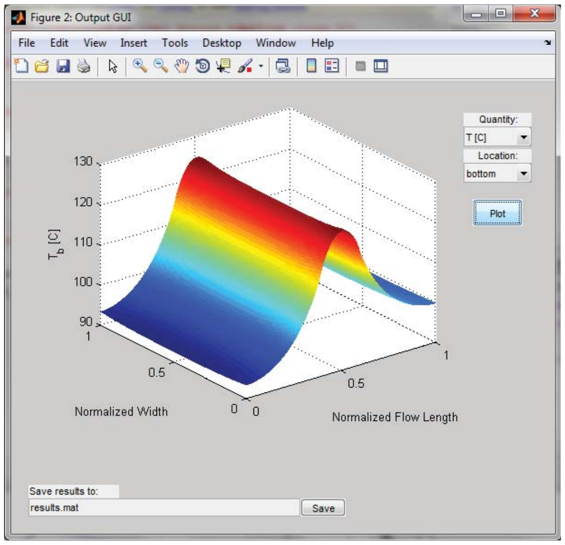
Figure 7.1. The algorithm used in the computational model.



L [mm]	12.67
w [um]	239.114
wf [um]	110.8
t [um]	648
d [um]	371.44
W [mm]	12.668
# of channels	35
P in [kPa]	101
T in [C]	90.5
V in [mL/min]	104
Q in [W]	33
Phi	0

Silicon
FC-77
Case 1a
Use saved data?
results.mat

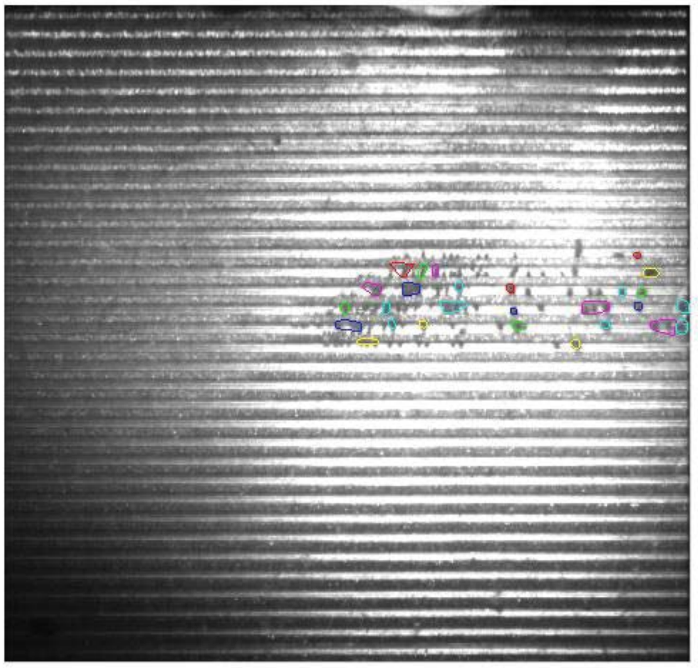
(a)



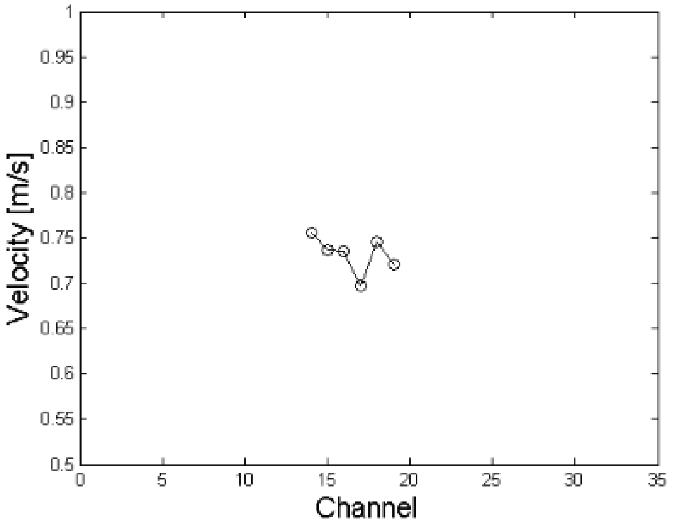
Quantity: T [C]
Location: bottom
Save results to: results.mat

(b)

Figure 7.2. The (a) input and (b) output GUIs of the computational model.



(a)



(b)

Figure 7.3. (a) Characteristic shapes detected to calculate the (b) bubble velocity in each channel.

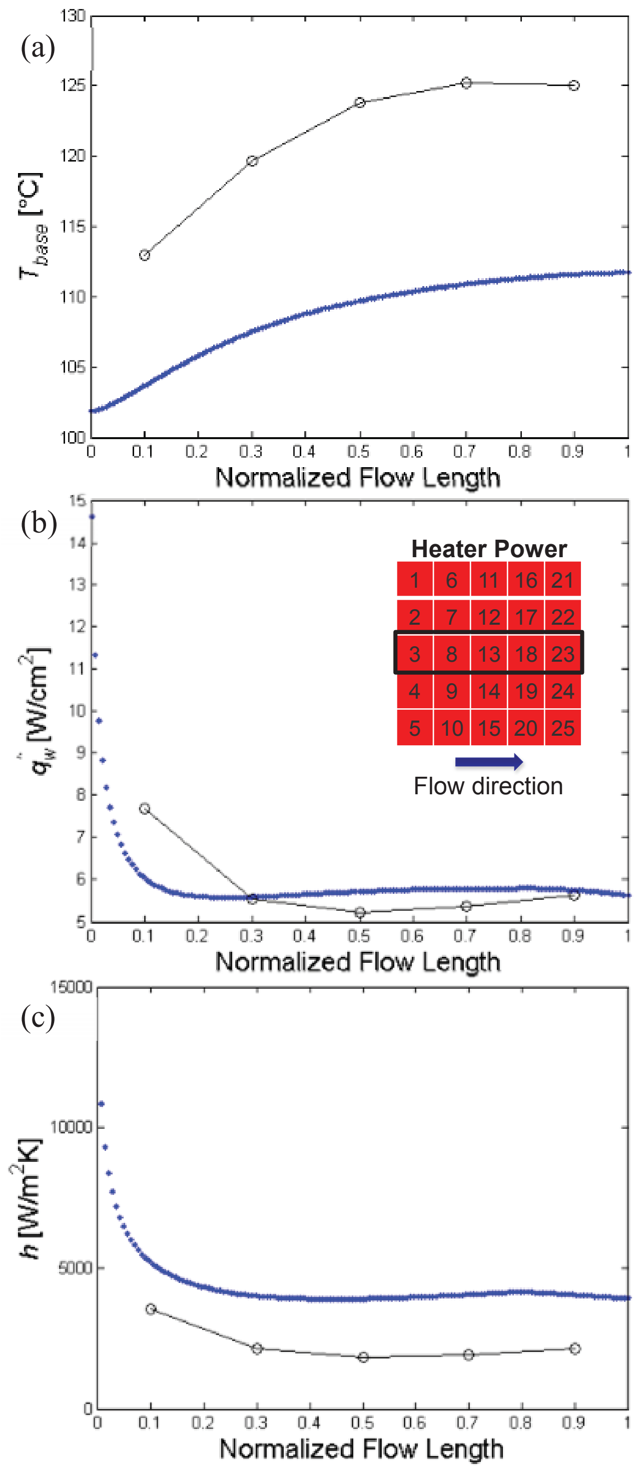


Figure 7.4. (a) The base temperature, (b) wall heat flux, and (c) heat transfer coefficients for a uniform heating case with a power input of 33 W.

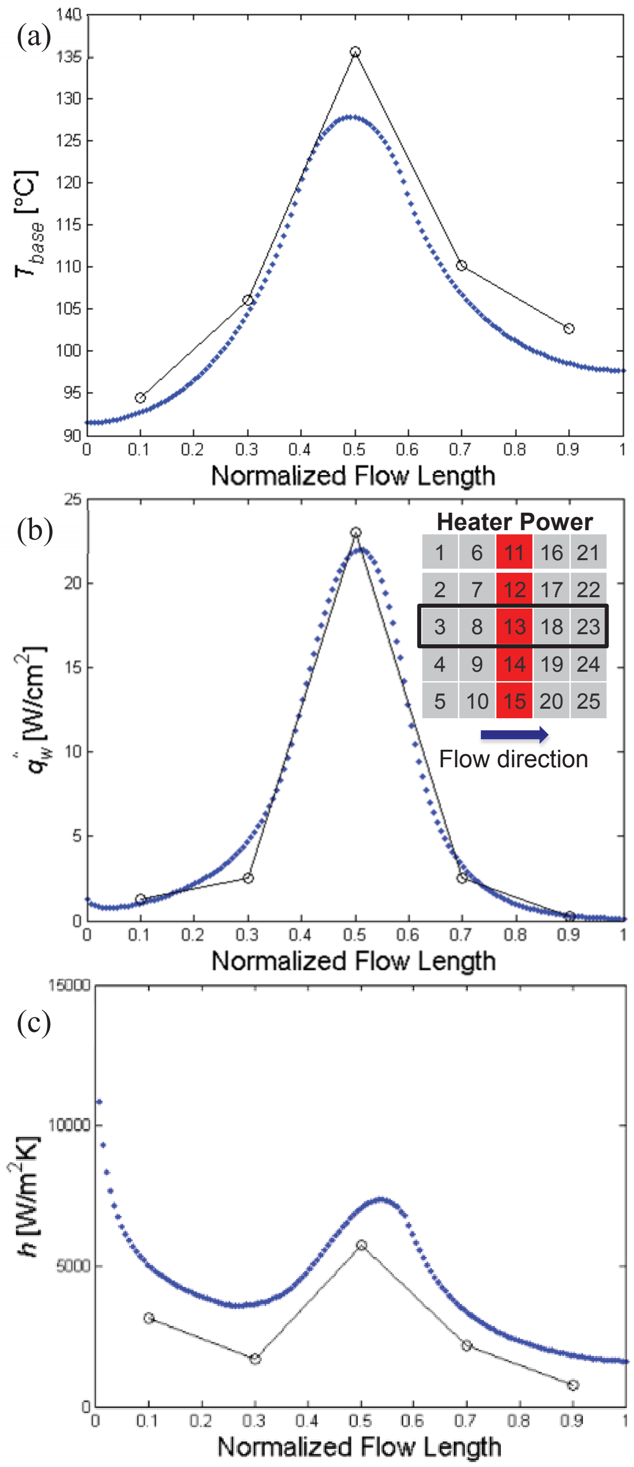


Figure 7.5. (a) The base temperature, (b) wall heat flux, and (c) heat transfer coefficients for a central transverse hotspot with a power input of 33 W.

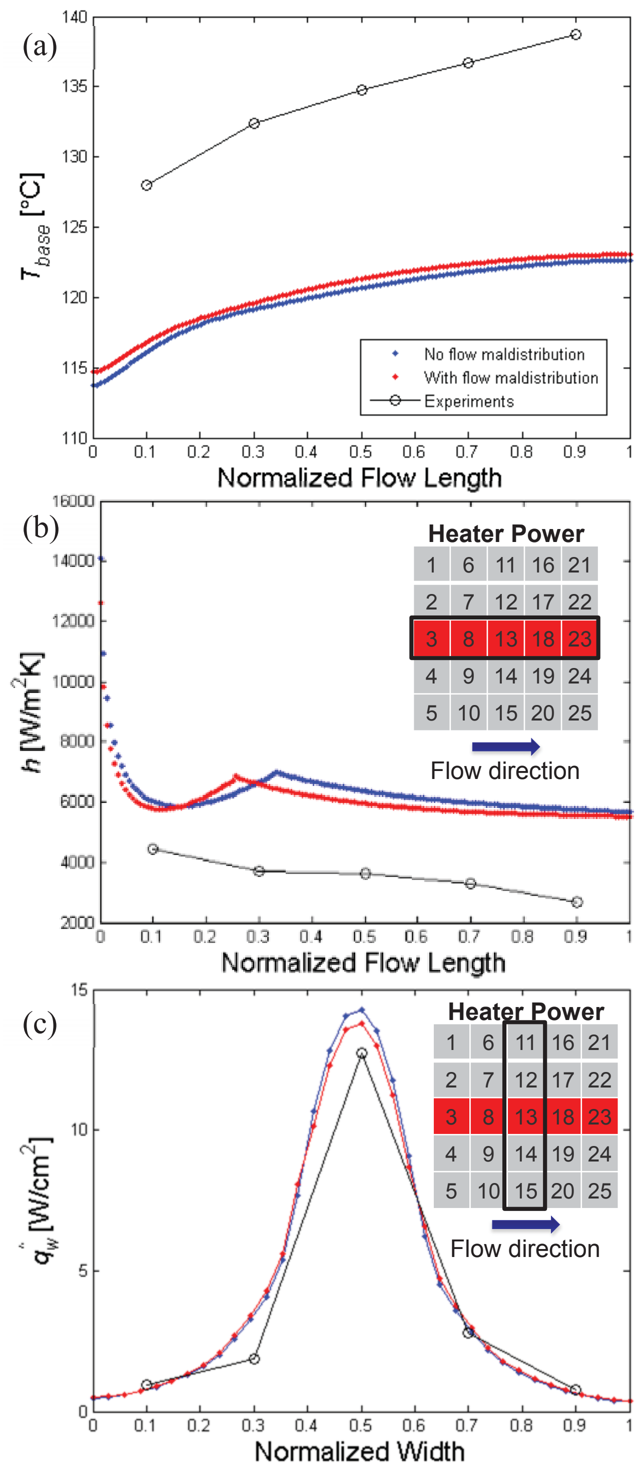


Figure 7.6. (a) The base temperature, (b) heat transfer coefficients, and (c) wall heat flux for a central streamwise hotspot with a power input of 24.4 W with and without flow maldistribution.

CHAPTER 8. EFFECTS OF NON-UNIFORM HEATING ON THE CRITICAL HEAT FLUX

The demanding performance, weight, and size constraints placed on modern electronics systems has driven packages and components increasingly thinner and more compact; incorporation of thick heat spreaders to mitigate propagation of heat generation non-uniformities is an obsolete thermal management strategy in high-performance systems. Instead, non-uniform heat flux profiles must be directly accommodated by the heat sink. Microchannel heat sinks are an excellent choice due to their ability to handle high heat fluxes, but an improved understanding of the effects of non-uniform heating profiles on the heat dissipation limits of microchannel heat sinks is needed to address these thermal packing trends. This chapter studies the location and magnitude of the critical heat flux in a microchannel heat sink with several canonical hotspot heating cases. This work gives a better understanding of how non-uniform heating profiles change the critical heat flux as compared to a uniform case. The material in this chapter was submitted to be considered for publication in the *International Journal of Micro-Nano Scale Transport*.

8.1 Experimental Methods

8.1.1 Test Section

The microchannel test section used in the current study is the same as that described in Section 6.1.1 and is shown as an inset in the flow loop schematic diagram Figure 8.1, a summary of the details are provided here. The working fluid routes through a silicon microchannel heat sink with a base area of 12.7 mm × 12.7 mm via a transparent manifold cover plate made of polycarbonate. The silicon heat sink is mounted directly on a printed circuit board (PCB) that is connected to an electrical quick-connect board; a G10 glass-epoxy composite layer is placed in between the PCB and quick-connect board as thermal insulation. A 0.8 mm-thick silicone rubber sheet is sandwiched between the microchannel heat sink and cover plate to insulate the polycarbonate from thermal damage (rated to a temperature of 115-130 °C), and forms a seal from the cover plate to the top walls of the microchannel fins to prevent cross flow between the channels. The manifold has inlet and outlet header sections, each with a flow length of 10 mm, width of 12.7 mm, and height of approximately 1.4 mm.

The silicon heat sink is manufactured by cutting microchannels in parallel into the top surface of a 650 μm-thick chip via a dicing saw. A single heat sink with 35 microchannels is used in the current study; the channel widths are 240 μm, channel depths are 370 μm, and fin widths are 110 μm. The channels were cut with multiple passes creating waviness on the bottom channel surface. The average roughness of a single cut is 0.2 μm, and the overall average surface roughness of the bottom of the channels is 0.82 μm. The sides of the channels have a surface roughness of 0.1 μm [9].

A 5 × 5 array of resistance heaters and temperature-sensing diodes is located on the bottom side of the heat sink. The equivalent individual heater resistances allow a single voltage to be applied in parallel to produce a uniform flux over the desired area. Connector pins are used to connect a DC voltage power supply to specific heaters to provide customized, non-uniform heating profiles to the underside of the microchannel heat sink. From the applied voltage, the local temperature at each diode and heat generated by the resistors are calculated based on a calibration of the resistance of each sensor. Details about the calibration procedure for each sensor is found in [9].

To prevent damage to the test chip while investigating CHF, a cutoff sensor is connected to the power supply. When the sensor detects a chip temperature above a preset threshold, the power supply is almost immediately (within a few milliseconds) disconnected from the heaters. The preset threshold was chosen as the upper limit of safe operating temperatures for the test chip, approximately 140 °C.

8.1.2 Flow Loop

The experimental flow loop is a modified version of that described in Section 8.1.2, and a schematic diagram is shown in Figure 8.1. A Micropump 415A magnetically coupled gear pump circulates the dielectric fluid HFE-7100 through the loop and a preheater warms the fluid to the desired test section inlet temperature. The fluid HFE-7100 was chosen for its low boiling point (61 °C at 1 atm) so that experiments can be run up to CHF within the safe operating temperature range of the test chip; note this is a different fluid than that used previously in [88]. A liquid-to-air heat exchanger is placed downstream of the test section to cool the fluid before reentering the reservoir. The liquid flow rate is measured using a 20-200 mL/min McMillan S-114 microturbine flow meter. The fluid temperature is measured using T-type thermocouples upstream of the preheater, upstream and downstream of the test section, and downstream of the heat exchanger. The inlet pressure is measured using a 0-30 psia Gems Sensors 2200 series pressure transducer. The pressure drop across the test section is measured using a 0-10 psi Omega PX2300 series differential pressure transducer.

High-speed videos viewing normal to the top of the heat sink are captured using a Photron Fastcam Ultima APX high-speed digital video camera combined with a Nikon AF Micro-Nikkor 200 mm IF-ED lens. The microchannel heat sink is illuminated using a Sunoptic Technologies Titan 300 xenon arc lamp. High-speed videos are captured at 8,000 frames per second using a shutter speed of 8 kHz.

8.1.3 Test Procedure

Prior to collecting data, the fluid is degassed using a vacuum pump connected to an expandable reservoir. The expandable reservoir design is detailed in [90]. At ambient temperature and pressure, HFE-7100 contains 53% air by volume [98].The expandable reservoir is first fully expanded to create a gas space at the top. A vacuum pump is used to produce a vacuum pressure equal to the vapor pressure of the fluid for 2 min to remove the combination of air and HFE-7100 vapor that has collected in the gas space. The expanded reservoir is then left at a vacuum pressure for at least one hour to allow the dissolved air to diffuse from the liquid into the gas space. The process is repeated until the vacuum pressure in the reservoir is constant with time, which indicates that air is no longer diffusing out of the liquid. The fluid is then cycled through the flow loop and the process is repeated to ensure that all air is removed from the system. The fluid is considered fully degassed once the measured fluid temperature at incipience is equal to the saturation temperature of HFE-7100 when boiled in the preheater.

Experiments in the current study are performed at a single mass flux of 797 kg/m^{2}s. After a constant flow rate is maintained, the fluid is preheated to a test section inlet temperature of approximately 50.8 °C that remains fixed throughout the test. The expandable reservoir is allowed to expand freely in order to maintain a relatively constant system pressure near atmospheric.

Power is supplied to individual heating elements to simulate various hotspot configurations. The total power supplied to these heating elements is incremented from zero until the critical heat flux is reached. At each power input level, the system is allowed to reach steady state before moving on to the next. When CHF is reached, the local temperature rapidly increases and the cutoff sensor is triggered. When the cutoff is triggered the first time, the total power supplied is noted and the system is reset. After the system is reset, the power input is set to just below the level that first triggered the cutoff, such that CHF can be approached in fine power input increments. After successive honing in on the power input nearing CHF, the system is allowed to reach steady state at a heater power at which any perceptible increase will cause it to reach CHF. This procedure provides an accurate measurement of CHF and the system conditions just below CHF. Transient CHF data is not included due to the limited frequency of the chip temperature measurements. High-speed videos of the microchannel heat sink are recorded while CHF is reached to visually capture the phenomenon.

The cutoff sensor conveniently allows repeated activation of critical heat flux using a single test chip without damage. A test chip was sacrificed to demonstrate the damage done if CHF is allowed to occur uninterrupted. Fifteen of the temperature sensors were permanently damaged and the silicon heat sink itself was cracked. Images of an undamaged test piece and a test piece damaged by CHF are shown in Figure 8.2.

8.1.4 Data Reduction

Local heat fluxes, wall temperatures, and fluid temperatures are measured for all data at steady state. The data reduction method briefly summarized here is the same as that used in Section 6.2; fluid properties have been updated to account for the change in fluid from FC-77 to HFE-7100. This process takes into account heat spreading that occurs within the heat sink base for cases with non-uniform heating. A MATLAB script of the data reduction process can be found in Appendix C.

The net local heat transfer rate from the microchannel heat sink to the fluid $\dot{q}_{net}$ is calculated based on an energy balance for each heating element which consists of the energy generated, the heat loss, and lateral conduction between elements as

$$\dot{q}_{net,ij} = \dot{q}_{gen,ij} - \dot{q}_{loss,ij} - \dot{q}_{cond,ij}. \tag{8.1}$$

The test section heat loss is calibrated as a function of the base temperature. A complete description of the calibration procedure is found in [91]. Lateral conduction that occurs when non-uniform heating profiles are imposed is calculated between elements as

$$\dot{q}_{cond,ij} = k_{Si}\frac{L}{5}(t-d)\frac{\left(4T_{i,j} - T_{i+1,j} - T_{i-1,j} - T_{i,j+1} - T_{i,j-1}\right)}{L/5}, \tag{8.2}$$

where the total net conduction depends on the four neighboring elements to heater i,j.

The bulk fluid temperature above each heating element in single-phase flow is calculated as

$$T_{f,ij} = T_{in} + \frac{\sum_i \dot{q}_{net,ij}}{GwdN/5c_p}, \tag{8.3}$$

where $\sum_i \dot{q}_{net,ij}$ represents the sum total of net heat transferred to the fluid from the inlet to the heating element in question. After sufficient sensible heating to the saturation temperature, the fluid temperature is then maintained equal to the saturation temperature at the local pressure.

A corrected local wall temperature is calculated based on the measured diode temperature by accounting for conduction through the base of the microchannel heat sink and is calculated as

$$T_{w,ij} = T_{d,ij} - \frac{q''_{b,ij}(t-d)}{k_{Si}}. \tag{8.4}$$

The local base heat flux is calculated using the local net heat transfer rate as

$$q''_{b,ij} = \frac{\dot{q}_{net,ij}}{A_b/25}. \tag{8.5}$$

The local heat flux transferred to the fluid is also calculated using the local net heat transfer rate and is based on the wetted area of the channels as

$$q''_{w,ij} = \frac{\dot{q}_{net,ij}}{NL(w+2d)/25}. \tag{8.6}$$

8.1.5 Test Cases

A variety of non-uniform heating profiles were investigated and are shown in Figure 8.3. Heaters in the hotspot locations (displayed in red) are turned on while the remainder (displayed in gray) are powered off. The first case is a uniformly heated profile that serves as a basis of comparison for the remaining non-uniform cases. The next three cases correspond to hotspots that span the width of the heat sink in the transverse direction, placed at the inlet, center, and outlet of the flow path. The next case corresponds to a centered hotspot that spans the length of the heat sink in the streamwise direction. The last case corresponds to two transverse hotspots located at the inlet and outlet.

8.2 Results and Discussion

The total power input, local heat flux, and maximum wall excess temperature at the critical heat flux are summarized for all cases in Table 8.1. Selected cases from this table will be analyzed in greater detail in subsequent paragraphs to illustrate the key effects of non-uniform heating on CHF observed. Figures containing the full dataset can be found in Appendix F. For all cases, the local heat flux and wall temperature are maximum above the active heater elements, as expected. The influence of non-uniform heating on the trends in local wall temperatures, heat fluxes, and heat transfer coefficients leading up to CHF are explained in detail in Section 6.3. Interested readers should refer to this previous section for detailed discussion of these trends; the discussion herein is exclusive to the influence on CHF (note that a different working fluid is used in the current chapter).

Boiling curves for a central transverse hotspot (sensors 11-15) and a central streamwise hotspot (sensors 3, 8, 13, 18, and 23) are shown in Figure 8.4. These curves are produced using the local heat flux transferred to the fluid and the local wall excess temperature. For single-phase flow, the wall excess temperature is calculated using the local bulk fluid temperature, and for two-phase flow it is calculated using the saturation temperature.

In the central transverse hotspot case, the wall excess temperature initially increases with a constant slope as the heat flux is increased. This reflects the relatively constant heat transfer coefficient characteristic of single-phase flow. The lines for all heated sensors overlap in this region. Boiling incipience is indicated by the wall temperature reduction and increased slope of the lines, and is confirmed in the high-speed videos. In the two-phase region, lines corresponding to the three middle sensors overlap (12-14); the two sensors on the boundaries (11 and 15) show a larger wall excess temperature. In this case, CHF occurs above sensor 15 on the boundary; as CHF is approached, a simultaneous decrease in the heat flux transferred to the fluid and increase in wall excess temperature is observed. This behavior aids identification of the general location of CHF for all cases; the exact location is confirmed using visual evidence from the high-speed imaging. Throughout this study, CHF typically occurs in a location near the lateral boundaries of the microchannel heat sink (unless that area is not heated). This location can be confirmed for all cases via high-speed images and the boiling curve (as was demonstrated for the central transverse case here). It is likely that CHF occurs in these locations because of maldistribution caused by the inlet manifold geometry that reduces the flow into these channels [99].

High-speed images during the period where the temperature cutoff is triggered are shown in Figure 8.5; CHF is determined to occur in the bottommost channel in the images. The images show a portion of the flow length across the chip and only the 4 channels nearest the edge of the heat sink over hotspot sensor 15. At time t = 0 ms, the system is at a stable point just prior to the occurrence of CHF. In the bottommost channel, small bubbles nucleate over the hotspot, grow, and are carried downstream. A short time later (t = 44.9 ms), CHF is reached and larger bubbles form in the channel which coalesce to form a larger vapor region. This vapor region quickly expands in both directions until the channel is almost completely full of vapor (t = 50.5 ms). This sudden vapor expansion causes a local rapid temperature increase and triggers the cutoff sensor. Additionally, the local pressure within the channel increases preventing liquid from entering and creating local flow reversal. After the power is cut off and the pressure in the bottom channel has equalized, liquid is allowed to flush through the channel and the wall temperature is reduced (t = 58.5-82.4 ms). Sudden vapor expansion is characteristic of CHF for all cases; however, it was most cleanly observed in the high-speed videos of the central transverse hotspot case.

Boiling curves for the central streamwise hotspot case are shown in Figure 8.4b. Like the central transverse hotspot case, a single-phase region with constant slope is observed. This slope is largest for the heater furthest upstream (sensor 3), corresponding to the higher heat transfer coefficient in the entrance region of thermally developing flow, and decreases for sensors successively downstream. Once again, boiling incipience is indicated when the slope of the lines change, as is confirmed with high-speed videos. In this case, since the entire flow path of the middle channels over the hotspot is heated, the behavior is expected to be similar to a uniformly heated case where CHF always occurs at the outlet [54]. In the case of the central streamwise hotspot, the temperature rise from sensors 3 to 8 in the upstream portion of the heat sink generally follows the trends for a uniform heating case, however rapid vapor expansion leads to a rapid temperature increase (i.e., CHF) slightly upstream of the outlet, above sensor 18. This is reflected in the boiling curve where the largest wall excess temperature is measured at sensor 18 and corresponds with a drop in the heat flux transferred to the fluid. High-speed video evidence for this case also indicates that bidirectional vapor expansion in the channels begins upstream of the outlet, centered over sensors 13 and 18. In this study, CHF always occurs within the footprint of the hotspot, and unlike uniform heating, does not necessarily occur at the outlet (even if it is part of the hotspot).

The boiling curve results showed that the location where the cutoff sensor triggers, considered the location where CHF occurs, is strongly dependent on the heating profile. Further interrogation of the data shows that the maximum wall excess temperature at CHF also varies significantly based on the heating configuration and ranges from 54.1 °C to 73.9 °C, as seen in Table 8.1. One mechanism for these differences is heat spreading in the substrate. Transverse hotspot cases conduct heat through the silicon substrate both up and downstream; streamwise hotspot cases conduct heat transverse to the flow direction between channels. Another possible mechanism for this difference is advection of heated fluid to non-heated regions within a channel. Since transverse hotspot cases allow for this to happen, it is expected that a transverse hotspot would see lower maximum wall excess temperatures at CHF as compared to streamwise and uniform heating profiles, where heated fluid continues to flow over downstream heated regions. In fact, the only cases where this is true lack a hotspot at the outlet of the heat sink. The two transverse cases that have active heaters at the outlet, viz., a transverse hotspot at the outlet and the dual transverse hotspots, have wall excess temperatures similar in magnitude to that of a uniformly heated case.

To explore the effect of a hotspot with the same basic heating profile placed in different locations on the heat sink, a transverse hotspot was investigated at inlet, central, and outlet locations along the microchannel flow path. The differences between these three cases are summarized in Table 8.1. Critical heat flux is reached at the highest base heat flux of total power input (61.3 W) when the hotspot is located in the center (not to be confused with the highest local *wall* flux). This is because heat has a path to spread both up and downstream to non-heated regions within the heat sink, whereas the inlet and outlet cases limit heat spreading to one direction. When the hotspot is moved to the outlet or inlet, CHF is reached at a lower total power input.

The local wall critical heat flux itself also varies based on the streamwise location of the transverse hotspot, and is indicative of the different hydrodynamics for each location at CHF. Temperature and heat flux maps of the three transverse hotspot cases at CHF are shown in Figure 8.6. The location where CHF occurs is marked with an “X”. When the hotspot is located at the outlet, the local critical heat flux is the lowest of the three cases at 29.7 W/cm^2. When the hotspot is at the inlet, the local critical heat flux of 53.4 W/cm^2 is the highest for all the cases tested. In general, the critical heat flux decreases as the hotspot moves downstream, and the wall excess temperature increases. The reason for this behavior has to do with the ability of the location of the hotspot to communicate with the inlet manifold. When the hotspot is located at the inlet, quickly expanding vapor regions more easily are able to reverse back into the manifold, which allows fresh liquid to enter the channel before thermal runaway. When the hotspot is located further downstream, vapor that may expand upstream must travel a longer distance to reach the inlet manifold. In the current study, the expanding vapor region was never observed to reach the inlet manifold in either the central or outlet transverse hotspot cases; the critical heat fluxes are hence decreased compared to the inlet case.

8.3 Design Principles

When designing a microchannel heat sink for a given non-uniform heating configuration, there are a few generic design principles that should be followed. First, the heat sink should be oriented so that the hotspot is located at the inlet. This will increase the critical heat flux and recuce the wall excess temperature. Second, the heat sink should be oriented so that as many channels as possible are located above the hotspot, in a transverse configuration. This will reduce the flow maldistribution in the heat sink and prevent a decrease in the critical heat flux. Third, the heat sink should be oriented so that the heated length imposed on the channels is minimized. If the hotspot cannot be distributed amongst all of the channels, reducing the heated length will reduce the flow maldistribution and mitigate any decrease in the critical heat flux.

8.4 Conclusions

The effects of non-uniform heating profiles on the location and conditions for the critical heat flux in a microchannel heat sink were investigated. Local wall temperatures, local wall heat fluxes, and total power input were measured using an array of embedded temperature sensors and high-speed videos were obtained. Hotspots that spanned the width or length of a silicon microchannel heat sink were explored by increasing the supplied input power until CHF was achieved.

In terms of the total power input necessary to reach CHF, a central streamwise hotspot is the worst case tested. Transverse hotspots generally have a larger total power input than a streamwise hotspot at CHF; the maximum power input is dissipated at CHF when a transverse hotspot is located in the center of the flow length, due to the auxiliary capability to conduct heat upstream and downstream. It was found that the central streamwise case yielded the lowest critical heat flux and largest maximum wall excess temperatures at CHF in comparison to all transverse cases. This is due to active heaters present along the entire flow length of the heat sink creating flow maldistribution among the channels. The transverse hotspot cases produced higher critical heat fluxes and lower maximum wall excess temperatures at CHF; the critical heat flux decreased and temperature increased as the hotspot moved from the inlet to the outlet. The high critical heat flux and low wall excess temperature that occurs when the hotspot is located at the inlet is attributed to the ability for expanding vapor regions to communicate with the inlet manifold.

The critical heat flux was identified based on trends in the boiling curves and a rapidly increasing local wall temperature that tripped a power supply cutoff sensor; the location of critical heat flux was confirmed via high-speed movies. A rapid vapor expansion in one or more channels above the hotspot leads to CHF. The location of CHF depended on the heating configuration and only occurred at the location of the hotspot, and not necessarily at the outlet. Furthermore, in the current study, CHF typically occurred in channels within the regions of the hotspot located on the lateral boundaries of the heat sink. The repeatable occurrence of CHF in these same channel locations is very likely due to slight flow maldistribution caused by the inlet manifold geometry. Thus, the location of CHF can be anticipated based on a combination of the heating profile and flow maldistribution between the channels.

Table 8.1. Summary of results for all test cases.

	Total power input at CHF (W)	Local heat flux at CHF (W/cm^2)	Maximum wall excess temperature at CHF (°C)
Uniform	137.2	14.2	67.0
Inlet transverse	57.2	53.4	54.1
Central transverse	61.3	42.8	62.5
Outlet transverse	44.8	29.7	67.4
Central streamwise	34.9	8.7	73.9
Dual transverse	89.7	25.9	65.9

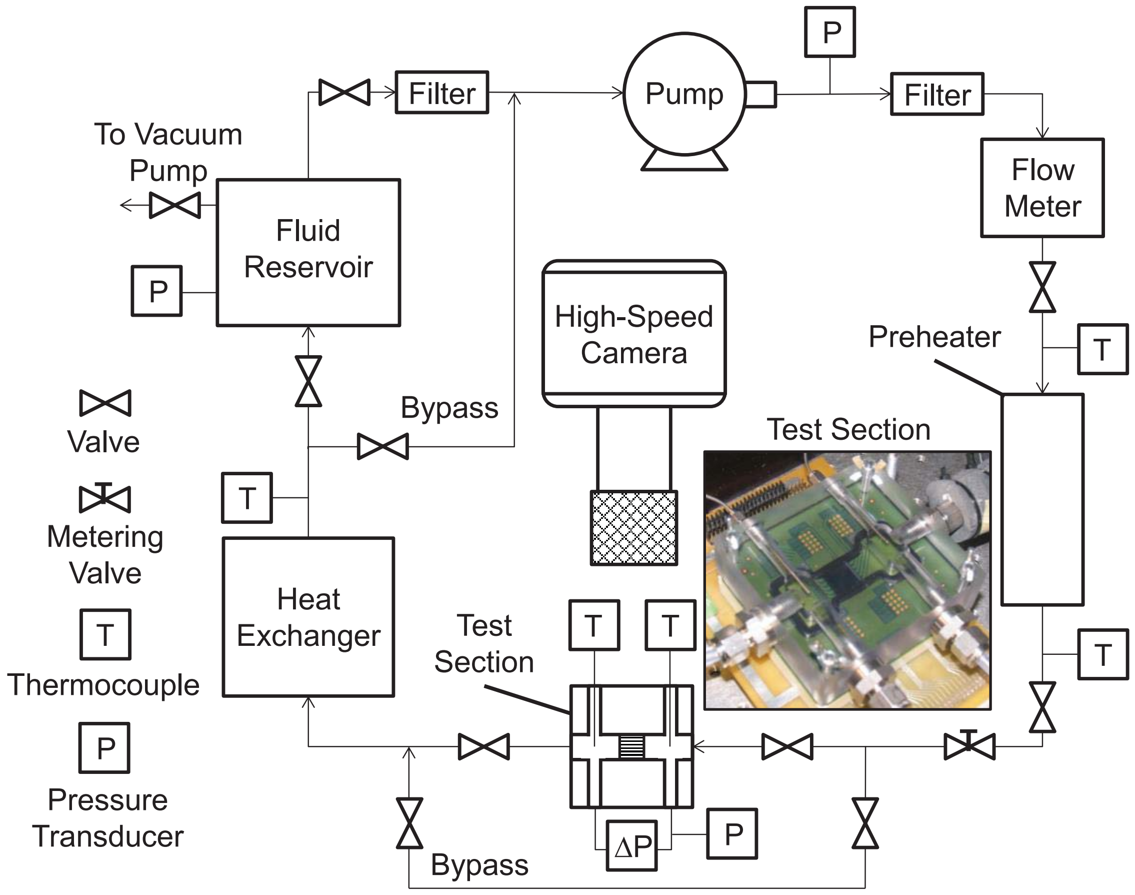


Figure 8.1. Schematic diagram of the flow loop; a photograph of the microchannel test section is inset.

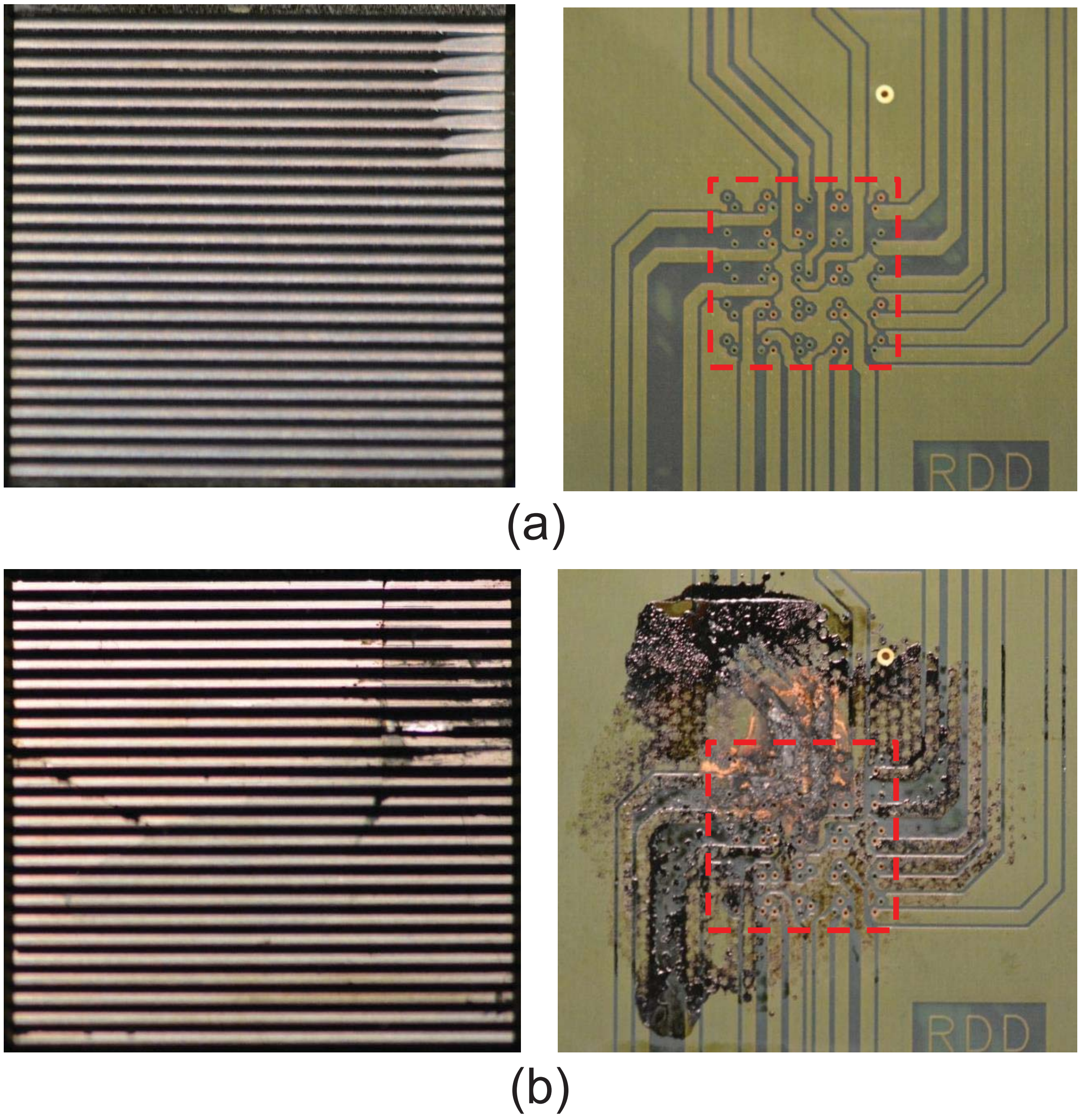


Figure 8.2. The microchannel heat sink and corresponding backside PCB traces shown for (a) an undamaged test chip and (b) a test chip damaged after CHF. Red lines indicate the location of the heat sink on the opposite side of the PCB to scale.

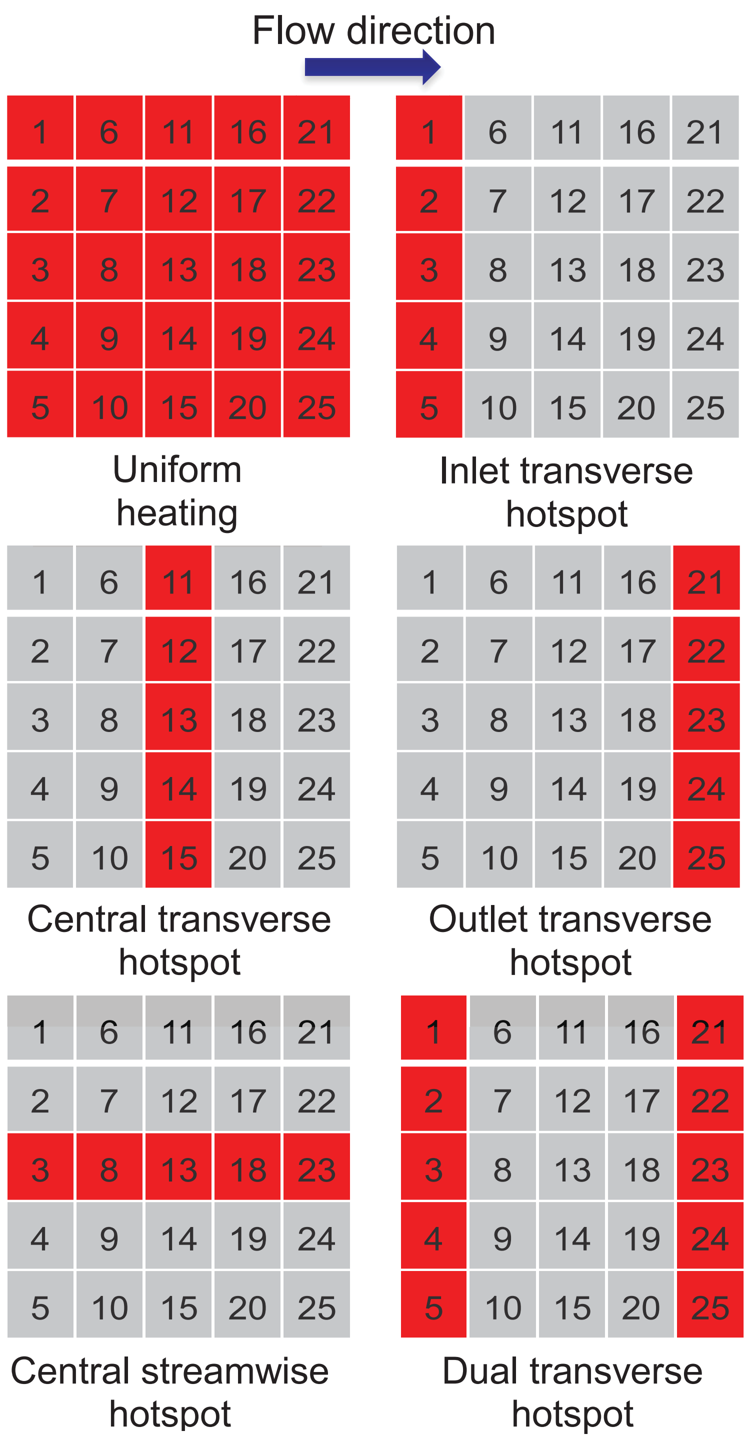


Figure 8.3. Non-uniform heating profiles investigated in the current study.

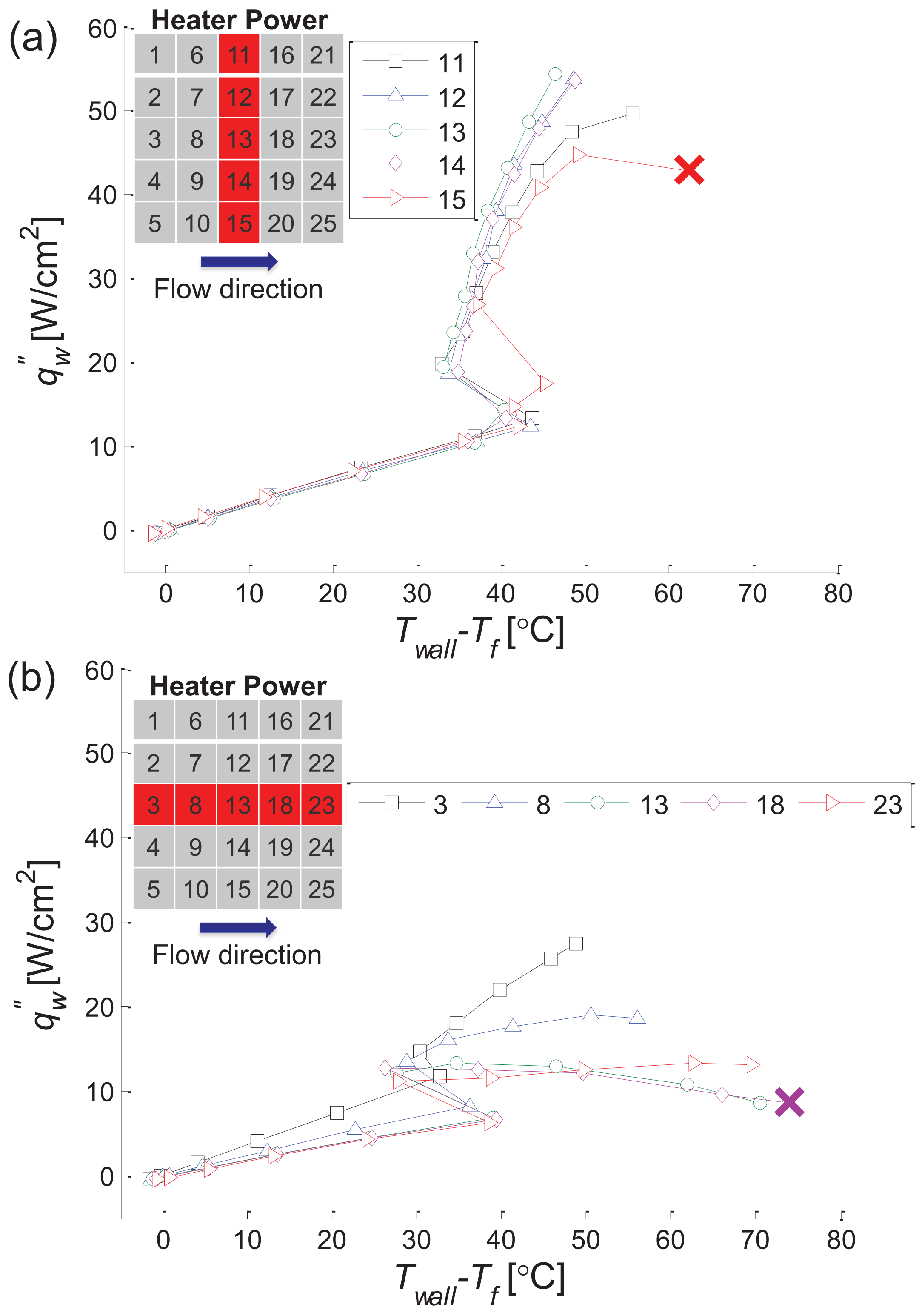
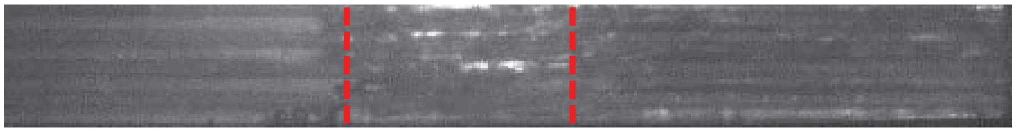
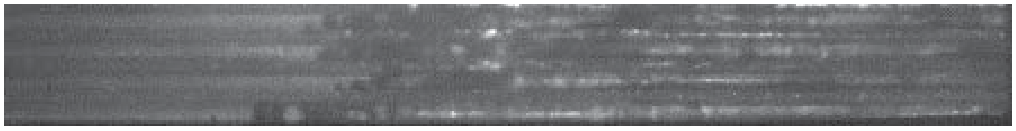


Figure 8.4. Heat flux transferred to the fluid plotted against the wall excess temperature for (a) a central transverse hotspot and (b) a central streamwise hotspot. “X” indicates the location of CHF.

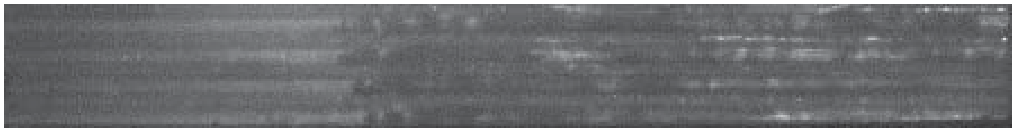


(a) t = 0



(b) t = 44.9 ms

(c) t = 50.5 ms



(d) t = 58.5 ms

(e) t = 82.4 ms

Figure 8.5. High-speed images recorded at 8000 frames per second for a central transverse hotspot at CHF. Flow goes from left to right; the hotspot is indicated by the red dashed lines. Vapor expands rapidly in the bottommost channel (maximum upstream distance at t = 50.5 ms) at CHF before the heater power is cut off.

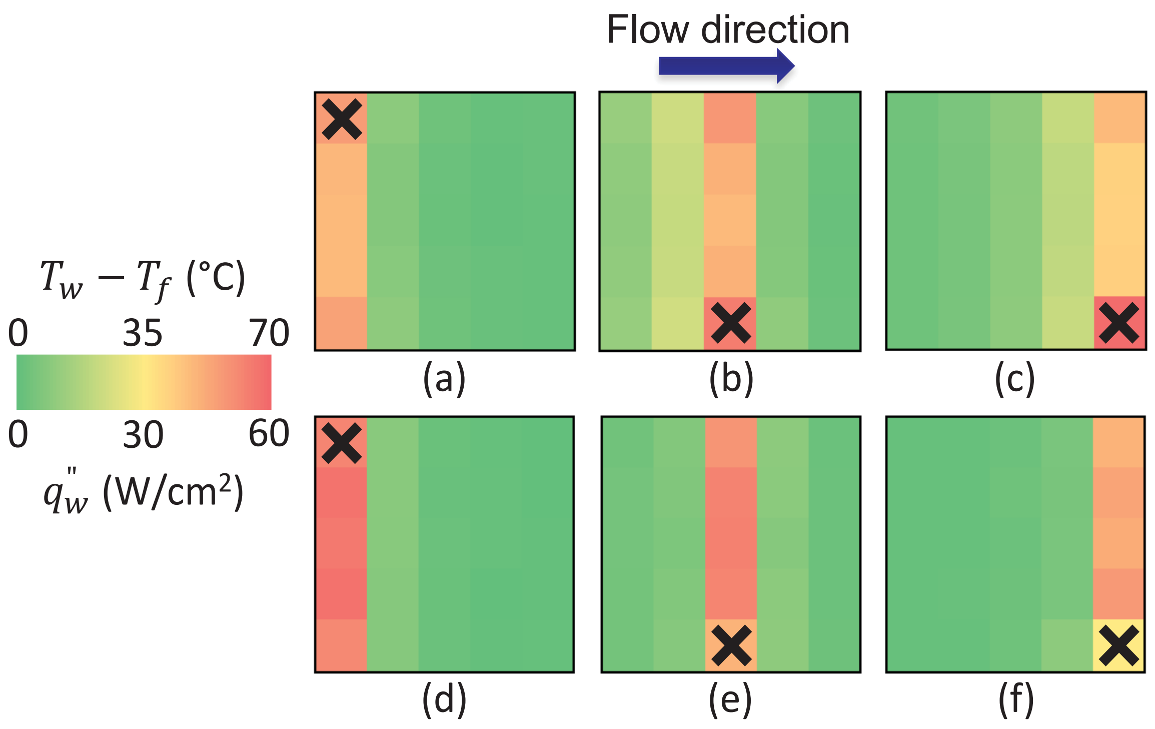


Figure 8.6. Wall excess temperature map for (a) inlet transverse, (b) central transverse, and (c) outlet transverse hotspot heating cases at CHF. Heat flux map for (d) inlet transverse, (e) central transverse, and (f) outlet transverse hotspot heating cases at CHF.

CHAPTER 9. CONCLUSIONS AND RECOMMENDATIONS

The main conclusions of this thesis are summarized in this chapter and recommendations for future work are provided.

9.1 Conclusions

In this work, two-phase flow in a microchannel heat sink was investigated. Experimental and numerical investigations of an impedance-based sensor were performed to measure the void fraction in air-water adiabatic flow in a square microchannel. Both crosswise and streamwise electrode configurations were examined and the void fraction was estimated using image analysis of high-speed videos. Additionally, an experimental study of the effects of non-uniform heating on a microchannel heat sink was performed. Several canonical hotspot and non-uniform peak heating cases were tested and the local wall temperatures, heat fluxes, and heat transfer coefficients were obtained. High-speed videos of boiling in the heat sink were also captured. A simple computational model was developed to predict the thermal performance of a microchannel heat sink exposed to non-uniform heating profiles and the predictions were compared to the obtained experimental data. Finally, an experimental study was performed to determine the effects of hotspots on the location and magnitude of the critical heat flux in a microchannel heat sink. Major findings of this thesis include:

1. The calculated time-averaged void fraction shows reasonable agreement with those predicted by the homogeneous equilibrium and drift-flux models.
2. The impedance void fraction meter measurement techniques can be used to study non-adiabatic and boiling flows with similar crosswise electrode geometry as long as changes in electrical properties of the fluid with temperature are taken into account.
3. The relationship between the void fraction and measured impedance is non-linear for all cases tested.
4. The shape and distribution of voids had no significant effect on simulated impedance for voids modeled in parallel.
5. A clear dependence on the fluid electrical conductivity was observed and an optimal range between 100 and 175 μS/cm was found for electrodes placed in a streamwise configuration.
6. For non-uniform heating in microchannel heat sinks, experimental results show that even with a very thin substrate, significant lateral conduction occurs in the base.
7. For a central streamwise hotspot, the maximum sustainable total power input achieved is reduced by 26.6% compared to a central transverse hotspot.
8. For a transverse hotspot located at the inlet, although the maximum sustainable total power input is similar to that of a central transverse hotspot case, the local maximum heat flux is increased by 35.7% as a result of significantly reduced upstream heat spreading.

9. The same total power input distributed in different locations and configurations across the heat sink can cause significantly different limits on the maximum heat fluxes and wall temperatures that can be supported.
10. For a non-uniform transverse peak-heating profile, an increase in the heating nonuniformity results in significant boiling at the location of peak heat input, whereas no boiling occurs under uniform heating conditions.
11. Taking into account flow maldistribution improved the match between the computational model and experimental data; large amounts of flow maldistribution cannot be ignored.
12. Of all the cases tested, a central streamwise hotspot is the worst case; it yielded the lowest critical heat flux and largest maximum wall excess temperature at CHF.
13. Active heaters present along the entire flow length of the heat sink create flow maldistribution among the channels and lowers the critical heat flux.
14. As a transverse hotspot is moved from the inlet to the outlet, CHF decreases and the maximum wall excess temperature increases.
15. A rapid vapor expansion in one or more channels above the hotspot leads to CHF.
16. The location of CHF depends on the heating configuration and only occurs at the location of the hotspot, and not necessarily at the outlet.

9.2 Suggestions for Future Work

Potential future work following that described in this thesis include:

1. The development of a critical heat flux correlation for non-uniform heating in a microchannel heat sink. The correlations for CHF found in the literature were all developed under uniform heating conditions. The experimental results showed a large difference in the local heat flux between uniform and non-uniform cases; current models cannot accurately predict the magnitude or the location of CHF. Once a correlation is developed, it can be implemented in the computational model for improved results.
2. Perform experiments to measure the pressure drop in each channel of a microchannel heat sink in order to better calculate flow maldistribution. Both the experiments and computational model show that when analyzing the thermal performance of a microchannel heat sink significant flow maldistribution cannot be ignored. Although the flow non-uniformities can be estimated using image analysis of high-speed videos, a more robust model based on local pressure drop is desired both for accuracy and for ease of measurement.
3. Incorporate improved heat transfer coefficent correlations into the computational model for improved results. The correlations for heat transfer coefficient used in the computational model were all derived for uniform heating conditions. An improved correlation either found in the literature or developed from experimental data presented in this thesis that takes into account the nuances of non-uniform heating will greatly improve the results of the computational model.

LIST OF REFERENCES

LIST OF REFERENCES

[1] A. Serizawa, Z. Feng, Z. and Kawara, 2002, “Two-phase flow in microchannels, Experimental Thermal Fluid Sciences, 26, pp. 703-714.

[2] X. Yang, J. Schlegel, Y. Liu, S. Paranjape, T. Hibiki and M. Ishii, 2012, “Measurement and modeling of two-phase flow parameters in scaled 8x8 BWR rod bundle,” International Journal of Heat and Fluid Flow, 34, pp. 85-97.

[3] E.S. Rosa, B.F. Flora and M.A.S.F. Souza, 2012, “Design and performance prediction of an impedance void meter applied to the petroleum industry,” Measurement Science and Technology, 23, pp. 1-14.

[4] D. Liu, P.S. Lee and S.V. Garimella, 2005, “Prediction of the onset of nucleate boiling in microchannel flow,” International Journal of Heat and Mass Transfer, 48, pp. 5134-5149.

[5] P.S. Lee and S.V. Garimella, 2008, “Saturated flow boiling heat transfer and pressure drop in silicon microchannel arrays,” International Journal of Heat and Mass Transfer, 51, pp. 789-806.

[6] S.S. Bertsch, E.A. Groll and S.V. Garimella, 2009, “Effects of heat flux, mass flux, vapor quality, and saturation temperature on flow boiling heat transfer in microchannels,” International Journal of Multiphase Flow, 35, pp. 142-154.

[7] R. Revellin and J.R. Thome, 2008, “A theoretical model for the prediction of the critical heat flux in heated microchannels,” International Journal of Heat and Mass Transfer, 51, pp. 1216-1225.

[8] A. Kosar, 2009, “A model to predict saturated critical heat flux in minichannels and microchannels,” International Journal of Thermal Sciences, 48, pp. 261-270.

[9] T. Harirchian and S.V. Garimella, 2008, “Microchannel size effects on local flow boiling heat transfer to a dielectric fluid,” International Journal of Heat and Mass Transfer, 51, pp. 3724-3735.

[10] T. Harirchian and S.V. Garimella, 2009, "Effects of channel dimension, heat flux and mass flux on flow boiling regimes in microchannels," International Journal of Multiphase Flow, 35, pp. 349-362.

[11] T. Harirchian and S.V. Garimella, 2010, "A comprehensive flow regime map for microchannel flow boiling with quantitative transition criteria," International Journal of Heat and Mass Transfer, 53, pp. 2694-2702.

[12] G.F. Hewitt, 1983, "Two-phase flow and its applications: past, present and future," Heat Transfer Engineering, 4, pp. 67-79.

[13] C.B. Sobhan and S.V. Garimella, 2001, "A comparative analysis of studies on heat transfer and fluid flow in microchannels," Microscale Thermophysics Engineering, 5, pp. 293-311.

[14] S.V. Garimella and C.B. Sobhan, 2003, "Transport in microchannels – a critical review," Annual Review of Heat Transfer, 13, pp. 1-50.

[15] S.S. Bertsch, E.A. Groll and S.V. Garimella, 2008, "Review and comparative analysis of studies on saturated flow boiling in small channels," Nanoscale Microscale Thermophysical Engineering, 12, pp. 187-227.

[16] A. Kawahara, M. Sadatomi, and K. Kumagae, 2006, "Effects of gas-liquid inlet/mixing conditions on two-phase flow in microchannels," Progress in Multiphase Flow Research, 1, pp. 197-203.

[17] A. Kawahara, M. Sadatomi, K. Nie, and H. Matsuo, 2009, "Experimental study on bubble velocity, void fraction and pressure drop for gas-liquid two-phase flow in a circular microchannel," International Journal of Heat and Fluid Flow, 30, pp. 831-841.

[18] M. Kawaji, A. Kawahara, K. Mori, M. Sadatomi, and K. Kumagae, 2006, "Gas-liquid two-phase flow in microchannels: the effects of gas-liquid injection methods," in Proceedings of the 18th National and Sevelth ISHMT-ASME Heat Transfer.

[19] J.C. Asali, T.J. Hanratty and P. Andreussi, 1985, "Interfacial drag and film height in vertical annular flow," AIChE Journal, 31, pp. 895-902.

[20] P. Andreussi, A. Di Donfrancesco and M. Messia, 1988, "An impedance method for the measurement of liquid hold up in two-phase flow," International Journal of Multiphase Flow, 14, pp. 777-785.

[21] N.A. Tsochatzidis, T.D. Karapantios, M.V. Kostoglou and A.J. Karabelas, 1992, "A conductance method for measuring liquid fraction in pipes and packed beds," International Journal of Multiphase Flow, 5, pp. 653-667.

[22] M. Fossa, 1998, “Design and performance of a conductance probe for measuring the liquid fraction in two-phase gas-liquid flows,” Journal of Flow Measurement and Instrumentation, 9, pp. 103-109.

[23] Y. Mi, M. Ishii, and L.H. Tsoukalas, 1998, “Vertical two-phase flow identification using advanced instrumentation and neural networks,” Nuclear Engineering and Design, 184, pp. 409-420.

[24] M.W.E. Coney, 1973, “The theory and application of conductance probes for the measurement of liquid film thickness in two-phase flow,” Journal of Physics E (Scientific Instruments), 6, pp. 903-911.

[25] M. Wang, W. Yin and N. Holliday, 2002, “A highly adaptive electrical impedance sensing system for flow measurement,” Measurement Science and Technology, 13, pp. 1884-1889.

[26] M.J. Da Silva, E. Schleicher and U. Hampel, 2007, “Capacitance wire-mesh sensor for fast measurement of phase fraction distributions,” Measurement Science and Technology, 18, pp. 2245-2251.

[27] H. Caniere, B. Bauwens, C. T’Joen and M. De Paepe, 2010, “Mapping of horizontal refrigerant two-phase flow patterns based on clustering of capacitive sensor signals,” International Journal of Heat and Mass Transfer, 53, pp. 5298-5307.

[28] Z. Huang, B. Wang and H. Li, 2003, “Application of electrical capacitance tomography to the void fraction measurement of two-phase flow,” IEEE Transactions on Instrumentation and Measurement, 52, pp. 7-12.

[29] O. Jones, Jr. and N. Zuber, 1975, “The interrelation between void fraction fluctuations and flow patterns in two-phase flow,” International Journal of Multiphase Flow, 2, pp. 273-306.

[30] N.K. Tutu, 1982, “Pressure fluctuations and flow pattern recognition in vertical two phase gas-liquid flows,” International Journal of Multiphase Flow, 8, pp. 443-447.

[31] G. Matsui, 1984, “Identification of flow regimes in vertical gas-liquid two-phase flow using differential pressure fluctuations,” International Journal of Multiphase Flow, 10, pp. 711-720.

[32] G. Costigan and P.B. Whalley, 1997, “Slug flow regime identification from dynamic void fraction measurements in vertical air-water flows,” International Journal of Multiphase Flow, 23, pp. 263-282.

[33] J.E. Julia, Y. Liu, S. Paranjape and M. Ishii, 2008, “Local flow regimes analysis in vertical upward two-phase flow,” Nuclear Engineering Design, 238, pp. 156-169.

[34] E.A. Hammer, T. Dyakowski and G.A. Johansen, 2006, Multiphase Flow Handbook, Editor: C.T. Crowe, CRC Press, Boca Raton, FL.

[35] J.F. Tullius, R. Vajtai and Y. Bayazitoglu, 2011, "A review of cooling in microchannels," Heat Transfer Engineering, 32, pp. 527-541.

[36] S. Kandlikar, 2012, "History, advances, and challenges in liquid flow and flow boiling heat transfer in microchannels: a critical review," Journal of Heat Transfer, 134.

[37] S.V. Garimella and T. Harirchian, 2013, Encyclopedia of Thermal Packaging, Volume 1: Microchannel Heat Sinks for Electronics Cooling, World Scientific, Singapore.

[38] E. Costa-Patry, J. Olivier, B. Michel and J.R. Thome, 2011, "Two-phase flow of refrigerants in 85 μm-wide multi-microchannels: part II – heat transfer with 35 local heaters," International Journal of Heat and Fluid Flow, 32, pp. 464-476.

[39] T.Y. Liu, P.L. Li, C.W. Liu and C. Gau, 2011, "Boiling flow characteristics in microchannels with very hydrophobic surface to super-hydrophilic surface," International Journal of Heat and Mass Transfer, 54, pp. 126-134.

[40] H.F. Hamann, A. Weger, J.A. Lacey, Z. Hu, P. Bose, E. Cohen, and J. Wakil, 2007, "Hotspot-limited microprocessors: direct temperature and power distribution measurements," Journal of Solid-State Circuits, 42, pp. 56-65.

[41] J.M. Koo, L. Jiang, A. Bari, L. Zhang, E. Wang, T.W. Kenny, J.G. Santiago and K.E. Goodson, 2002, "Convective boiling in microchannel heat sinks with spatially-varying heat generation," in Proceedings of the ASME 8th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electroic Systems (ITHERM), San Diego, CA, pp. 341-346.

[42] E.S. Cho, J.W. Choi, J.S. Yoon, and M.S. Kim, 2010, "Modeling and simulation on the mass flow distribution in microchannel heat sinks with non-uniform heat flux conditions," International Journal of Heat and Mass Transfer, 53, pp. 1341-1348.

[43] R.K. Sarangi, A. Bhattacharya, and R.S. Prasher, 2009, "Numerical modeling of boiling heat transfer in microchannels," Applied Thermal Engineering, 29, pp. 300-309.

[44] R. Revellin, J.M Quiben, J. Bonjour and J.R. Thome, 2008, "Effect of local hot spots on the maximum dissipation rates during flow boiling in a microchannel," IEEE Transactions on Components and Packaging Technologies, 31, pp. 407-416.

[45] D. Bogojevic, K. Sefiane, A.J. Walton, H. Lin, G. Cummins, D.B.R. Kenning and T.G. Karayiannis, 2009, “Experimental investigation of non-uniform heating on flow boiling instabilities in a microchannels based heat sink,” in Proceedings of the ASME 7th International Conference on Nanochannels, Microchannels and Minichannels (ICNMM), Pohang, South Korea, 82121.

[46] E. Costa-Patry, 2011, “Cooling high heat flux micro-electronic systems using refrigerants in high aspect ratio multi-microchannel evaporators,” PhD Thesis, Ecole Polytechnique Federale De Lausanne.

[47] E.S. Cho, J.W. Choi, J.S. Yoon, and M.S. Kim, 2010, “Experimental study on microchannel heat sinks considering mass flow distribution with non-uniform heat flux conditions,” International Journal of Heat and Mass Transfer, 53, pp. 2159-2168.

[48] T. Alam, P.S. Lee, C.R. Yap, and L. Jin, 2013, “A comparative study of flow boiling heat transfer and pressure drop characteristics in microgap and microchannel heat sink and an evaluation of microgap heat sink for hotspot mitigation,” International Journal of Heat and Mass Transfer, 58, pp. 335-347.

[49] J.L. Miler, R. Flynn, G. Refai-Ahmed, M. Touzelbaev, M. David, J. Steinbrenner, and K.E. Goodson, 2009, “Effects of transient heating on two-phase flow response in microchannel heat exchangers,” in Proceedings of the ASME 2009 InterPACK Conference, San Francisco, CA, USA, InterPACK2009-89325.

[50] A.P. Roday and M.K. Jensen, 2009, “A review of the critical heat flux condition in mini- and microchannels,” Journal of Mechanical Science and Technology, 23, pp. 2529-2547.

[51] C.B. Tibirica and G. Ribatski, 2013, “Flow boiling in micro-scale channels – Synthesized literature review,” International Journal of Refrigeration, 36, pp. 301-324.

[52] L. Wojtan, R. Revellin, and J.R. Thome, 2006, “Investigation of saturated critical heat flux in a single, uniformly heated microchannel,” Experimental Thermal and Fluid Science, 30, pp. 765-774.

[53] D. Del Col and S. Bortolin, 2012, “Investigation of dryout during flow boiling in a single microchannel under non-uniform axial heat flux,” International Journal of Thermal Sciences, 57, pp. 25-36.

[54] W. Qu and I. Mudawar, 2004, “Measurement and correlation of critical heat flux in two-phase micro-channel heat sinks,” International Journal of Heat and Mass Transfer, 47, pp. 2045-2059.

[55] T. Chen and S.V. Garimella, 2012, "A study of critical heat flux during flow boiling in microchannel heat sinks," Journal of Heat Transfer, 134, 0011504.

[56] A. Kosar, Y. Peles, A.E. Bergles, and G.S. Cole, 2009, "Experimental investigation of critical heat flux in microchannels for flow-field probes," in ASME Seventh International Conference on Nanochannels, Microchannels, and Minichannels, Pohang, South Korea, June 22-24, Paper No. ICNMM2009-82214.

[57] M.B. Bowers and I. Mudawar, 1994, "High flux boiling in low flow rate, low pressure drop mini-channel and micro-channel heat sinks," International Journal of Heat and Mass Transfer, 37, pp. 321-332.

[58] Y. Katto and H. Ohno, 1984, "An improved version of the generalized correlation of critical heat flux for the forced convective boiling in uniformly heated vertical tubes," International Journal of Heat and Mass Transfer, 27, pp. 1641-1648.

[59] W. Zhang, T. Hibiki, K. Mishima, and Y. Mi, 2006, "Correlation of critical heat flux for flow boiling of water in mini-channels," International Journal of Heat and Mass Transfer, 49, pp. 1058-1072.

[60] R. Revellin, K. Mishima, and J.R. Thome, 2009, "Status of prediction methods for critical heat fluxes in mini and microchannels," International Journal of Heat and Fluid Flow, 30, pp. 983-992.

[61] C.B. Tibirica, H.O.M. Felcar, and G. Ribatski, 2008, "An analysis of experimental data and prediction methods for critical heat fluxes in micro-scale channels," in 5th European Thermal-Science Conference, Eindhoven, Netherlands, May 18-22.

[62] D.D. Hall and I. Mudawar, 2000, "Critical heat flux (CHF) for water flow in tubes – II. Subcooled CHF correlations," International Journal of Heat and Mass Transfer, 43, pp. 2605-2640.

[63] M.M. Shah, 1987, "Improved general correlation for critical heat flux during upflow in uniformly heated vertical tubes," International Journal of Heat and Fluid Flow, 8, pp. 326-335.

[64] R. Revellin and J.R. Thome, 2009, "Critical heat flux during flow boiling in microchannels: A parametric study," Heat Transfer Engineering, 30, pp. 556-563.

[65] S.G. Kandlikar, 2010, "A scale analysis based theoretical force balance model for critical heat flux (CHF) during saturated flow boiling in microchannels and minichannels," Journal of Heat Transfer, 132, 081501.

[66] E. Costa-Patry and J.R. Thome, 2012, "On-chip cooling of hot-spots with a copper micro-evaporator," in 28th Annual IEEE Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), San Jose, CA, March 18-22.

[67] C.K. Liu, S.J. Yang, Y.L. Chao, K.Y. Liou, and C.C. Wang, 2013, "Effect of non-uniform heating on the performance of the microchannel heat sinks," International Communications in Heat and Mass Transfer, 43, pp. 57-62.

[68] J. Yang, D. Groeneveld, L. Leung, S. Cheng, and M. El Nakla, 2006, "An experimental and analytical study of the effect of axial power profile on CHF," Nuclear Engineering and Design, 236, pp. 1384-1395.

[69] E. Rosal, J. Cermak, L. Tong, J. Caterline, S. Kokolis, and B. Matzner, 1974, "High pressure rod bundle DNB data with axially non-uniform heat flux," Nuclear Engineering and Design, 31, pp. 1-20.

[70] A. Olekhnovitch, J. Sun, and A. Teyssedou, 2008, "A complex but accurate correlation for predicting critical heat flux in a round tube for low and medium pressures under circumferentially non-uniform heating conditions," International Journal of Heat and Mass Transfer, 51, pp. 2041-2054.

[71] S. Paranjape, S.N. Ritchey, and S.V. Garimella, 2011, "Impedance-based void fraction measurement and flow regime identification in microchannel flows," in Pacific Rim Technical Conference & Exposition on Packaging and Integration of Electronic and Photonic Systems (InterPACK), Portland, OR, July 6-8.

[72] S. Paranjape, S.N. Ritchey, and S.V. Garimella, 2012, "Impedance-based void fraction measurement and flow regime identification in microchannel flows under adiabatic conditions," International Journal of Multiphase Flow, 42, pp. 175-183.

[73] S. Tumanski, 2006, Principles of Electrical Measurement, CRC Press, Taylor & Francis, USA.

[74] The Mathworks, Inc., 2009, MATLAB version 2009b.

[75] P. Soille, 2003, Morphological Image Process, 2nd Ed., Springer-Verlag, Germany.

[76] J.C. Maxwell, 1873, A Treatise on Electricity and Magnetism, 3rd Ed., Clarendon Press, Oxford, England.

[77] N. Zuber and J.A. Findlay, 1965, "Average volumetric concentration in two-phase flow systems," Journal of Heat Transfer, 87, pp. 453-468.

[78] A.A. Armand, 1946, "The resistance during the movement of a two-phase system in horizontal pipes," Izv. Vses. Teplotekh., Inst., 1, pp. 16-23 (AERE-Lib/Trans 828).

[79] M.I. Ali, M. Sadatomi, and M. Kawaji, 1993, "Two-phase flow in narrow channels between flat plates," Can. Journal of Chemical Engineering, 71, pp. 657-666.

[80] K. Mishima and T. Hibiki, 1996, “Some characteristics of air-water two-phase flow in small diameter vertical tubes,” International Journal of Multiphase Flow, 22, pp. 703-712.

[81] A.W. Bowman and A. Azzalini, 1997, Applied Smoothing Techniques for Data Analysis, Oxford University press, New York.

[82] T. Kohonen, 1997, Self-Organizing Maps, 2nd Ed., Springer-Verlag, Berlin.

[83] ANSYS, Inc., 2009, ANSYS Fluent, Release 12.0.

[84] P. Valiorgue, S.N. Ritchey, J.A. Weibel and S.V. Garimella, 2014, “Design of a non-intrusive electrical impedance-based void fraction sensor for microchannel two-phase flows,” Measurement Science and Technology, 25, 095301.

[85] Fluent, Inc., 2007, Gambit version 2.4.6.

[86] S.V. Patankar, 1980, “Numerical Heat Transfer and Fluid Flow, Hemisphere Publishing, Washington, D.C.

[87] S.N. Ritchey, J.A. Weibel and S.V. Garimella, 2013, “Effects of non-uniform heating on two-phase flow through microchannels,” In Proceedings of the ASME International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems (InterPACK), Burlingame, CA, 73058.

[88] S.N. Ritchey, J.A. Weibel, and S.V. Garimella, 2014, “Local measurement of flow boiling heat transfer in an array of non-uniformly heated microchannels,” International Journal of Heat and Mass Transfer, 71, pp. 206-216.

[89] T. Harirchian, 2010, “Two-phase flow and heat transfer in microchannels,” PhD Thesis, Purdue University.

[90] T. Chen and S.V. Garimella, 2006, “Effect of dissolved air on subcooled flow boiling of a dielectric coolant in a microchannel heat sink,” Journal of Electronic Packaging, 128, pp. 398-404.

[91] T. Chen and S.V. Garimella, 2006, “Measurements and high-speed visualization of flow boiling of a dielectric fluid in a silicon microchannel heat sink,” International Journal of Multiphase Flow, 32, pp. 957-971.

[92] J. Taylor, 1997, An Introduction to Error Analysis, 2nd Ed., University Science Books.

[93] R.S. Patel, T. Harirchian and S.V. Garimella, 2011, “Dependence of flow boiling heat transfer coefficient on location and vapor quality in a microchannel heat sink,” in Proceedings of the Pacific Rim Technical Conference & Exposition on Packaging and Integration of Electronic and Photonic Systems (InterPACK), Portland, OR, 52089.

[94] P.S. Lee and S.V. Garimella, 2006, “Thermally developing flow and heat transfer in rectangular microchannels of different aspect ratios,” International Journal of Heat and Mass Transfer, 49, pp. 3060-3067.

[95] S.S. Bertsch, E.A. Groll, and S.V. Garimella, 2009, “A composite heat transfer correlation for saturated flow boiling in small channels,” International Journal of Heat and Mass Transfer, 52, pp. 2110-2118.

[96] M.G. Cooper, 1984, “Heat flow rates in saturated nucleate pool boiling – a wide-ranging examination using reduced properties,” Advanced Heat Transfer, 16, pp. 157-239.

[97] H. Hausen, 1943, “Darstellung des Warmeuberganges in Rohren durch verallgemeinerte Potenzbeziehungen,” Z. VDI Beiheft Verfahrenstechnik, 4, pp. 91-102.

[98] 3M, 2002, “3M Novec Engineering Fluid HFE-7100 for Heat Transfer,” 3M, St. Paul, MN, pp. 1-8.

[99] B.J. Jones, P.S. Lee, and S.V. Garimella, 2008, “Infrared micro-particle image velocimetry measurements and predictions of flow distribution in a microchannel heat sink,” International Journal of Heat and Mass Transfer, 51, pp. 1877-1887.

[100] A.M. Khounsary, D. Chojnowski, and L. Assoufid, 1997, “Thermal contact resistance across a copper-silicon interface,” in Optical Science, Engineering and Instrumentation, International Society for Optics and Photonics, pp. 45-51.

APPENDICES

Appendix A MATLAB Script for Image Processing

A MATLAB script was developed to analyze high-speed videos as discussed in Section 3.2.1 to determine the void fraction. The following script reads in each frame of a movie file, determines the boundaries of the gas regions, and calculates the void fraction. It is split up into many functions; Table A.1 displays the function number, name, description, and page number where it can be found.

Table A.1. A list of all of the functions of the image analysis script.

Function Number	Function Name	Description	Page Number
1	voidfrac.m	The main program	171
2	annpoints.m	Organizes detected edge points for annular flow	176
3	annvoid.m	Calculates void fraction data for annular flow	177
4	backg.m	Generates a background image and subtracts it for all frames	178
5	contbound.m	Fills in points to obtain a closed object	179
6	edgedetann.m	Detects bubbles edges in annular flow	179
7	edgedetect.m	Detects bubbles edges	180
8	fit_ellipse.m	Fits an ellipse to the detected edge points in bubbly flow	181
9	GetBgImage.m	Obtains a user specified background image and subtracts it for all frames	184
10	identify.m	Identifies objects in an image	185
11	inputin.m	Obtains information from the user	187
12	joinpoints.m	Find a neighboring point to connect two segments	188
13	movieplot1.m	Capture images to create a movie	189
14	outputann.m	Calculate the output for annular flow	190
15	outputbub.m	Calculate the output	190

Table A.1. Continued.

Function Number	Function Name	Description	Page Number
16	plotdataann.m	Plot the output for annular flow	191
17	plotdatabub.m	Plot the output	191
18	points.m	Obtain points above a line of symmetry	192
19	readrotate.m	Read in and rotate frames from a movie	192
20	rotatebub.m	Rotate detected edge points in a bubble	193
21	slugvoid.m	Calculate void fraction data for slug flow	194
22	symmetry.m	Find a line of symmetry for an object	195
23	trap.m	Calculate a volume rotation using the trapezoid rule	196

Function 1. voidfrac.m:

```
clear all;
close all;
tic;
[filename,width,depth,regime,xfrac,StartFrame,EndFrame,OutFileName,Angl
eRotate,CropUpRow,CropDnRow,CropLCol,CropRCol,NFrameBG,BGFileName,BGFra
meNumber,ImAdjustLow,ImAdjustHigh,ImAdjustLowAnn,ImAdjustHighAnn,CannyL
ow,CannyHigh,CannyWidth,CannyLowAnn,CannyHighAnn,CannyWidthAnn,findback
g]=inputin();   %input info from user
disp('reading movie...');
[frames,movs,movcolor]=readrotate(filename,StartFrame,EndFrame,AngleRot
ate,CropUpRow,CropDnRow,CropLCol,CropRCol);       %read in movie frames
%findbackg=input('User supplied background? yes=1, no=0 ');
if findbackg    %user supplied background
     BGUsr=GetBgImage(BGFileName,BGFrameNumber,AngleRotate,CropUpRow,Cro
pDnRow,CropLCol,CropRCol);
    [bg,frames2]=backg(frames,movs,NFrameBG,BGUsr);
else            %need to find background
    if regime==1
        BGUsr=0;
        [bg,frames2]=backg(frames,movs,NFrameBG,BGUsr);      %subtract
background from frames
    else
        bg=255*ones(size(frames{1}),'uint8');
        frames2=frames;
    end
end
fln=strrep(filename,'.avi','');                            %test name
%name of log file
filen=strcat('logfile_',fln,'_2_',num2str(StartFrame),'to',num2str(EndF
rame),'.txt');
file1=fopen(filen,'w');                                    %create log file
fprintf(file1,'Log File for test %s\n',fln);
fprintf(file1,'Start frame: %4.0f\n',StartFrame);
fprintf(file1,'End frame: %4.0f\n',EndFrame);
if regime==1
    fprintf(file1,'Flow regime: bubbly\n');
elseif regime==2
    fprintf(file1,'Flow regime: slug\n');
else
    fprintf(file1,'Flow regime: churn/annular\n');
end
fprintf(file1,'Channel dimensions:%4.0f x%4.0f square
microns\n',width,depth);
s=size(frames{1});
fprintf(file1,'Image dimensions:%4.0f x%4.0f square
pixels\n',s(1),s(2));
fprintf(file1,'------------------------------------------------\n');
%preallocate variables
voidf=zeros(1,movs(2));
Avoid=zeros(1,movs(2));
SAtot=zeros(1,movs(2));
BW=cell(1,movs(2));
coords=cell(movs(2),1);
xsall=cell(movs(2),1);
ysall=cell(movs(2),1);
yn3all=cell(movs(2),1);
xn2all=cell(movs(2),1);
mall=cell(movs(2),1);
ball=cell(movs(2),1);
allframes(movs(2))=struct('cdata',[],'colormap',[]);
disp('running calculations');
for n=1:movs(2)
    disp([num2str(100*n/movs(2)) '% complete']);    %display percent
complete
    if frames2{n}==255         %if frame is completely white (no bubbles)
        disp('skiped frame');
        fprintf(file1,'Skipped frame %i, same as background\n',n);
        s3=size(frames2{n});
        if regime~=1 && regime~=2
            BW{n}=zeros(s3(1)+2*(CannyWidthAnn+1),s3(2));
            BW{n}=logical(BW{n});
            [allframes(n)]=knovieplot1(frames{n},coords{n},xline,0,[s3(1
)+2*(CannyWidthAnn+1),s3(2)],WBbox);
        else
            BW{n}=zeros(s3(1)+2*(CannyWidth+1),s3(2)+2*(CannyWidth+1));
            BW{n}=logical(BW{n});
            [allframes(n)]=movieplot1(frames{n},coords{n},xline,0,s3+2*
(CannyWidth+1),WBbox);
        end
        continue;
    end
    if regime==1 || regime==2
        [BW{n},WBbox]=edgedetect(frames2{n},ImAdjustLow,ImAdjustHigh,Ca
nnyLow,CannyHigh,CannyWidth);    %detect bubble edges (bubbly, slug)
    else
```

```
        [BW{n},WBbox]=edgedetann(frames2{n},ImAdjustLowAnn,ImAdjustHigh
Ann,CannyLowAnn,CannyHighAnn,CannyWidthAnn); %detect bubble edges
(churn, annular)
    end
    %identify bubble coordinates
    [bound,bound2,dpixels,xline,s,ss,wratio,linebub]=identify(BW{n},WBbo
x,width,depth,xfrac,regime);
    coords{n}=bound2;
    for i=1:s(1)
        if linebub
            fprintf(file1,'Skipped bubble %i in frame %i, bubble is a
line\n',i,n);
        end
    end
    %optional movie with edges drawn on top
    [allframes(n)]=movieplot1(frames{n},bound2,xline,s,ss,WBbox);
    %[colormap2,allframes{n}]=movieplot2(frames2{n},movcolor,bound);
    if regime==1                %calculations for bubbly flow
        %preallocate variables
        Abub=zeros(1,s(1));
        Vtot=zeros(1,s(1));
        SAt=zeros(1,s(1));
        mall{n}=cell(s(1),1);
        ball{n}=cell(s(1),1);
        xsall{n}=cell(s(1),1);
        ysall{n}=cell(s(1),1);
        yn3all{n}=cell(s(1),1);
        xn2all{n}=cell(s(1),1);
        for i=1:s(1)            %calculations for each bubble in frame
            [Abub(i),m,b,empty]=symmetry(bound2{i},xline,ss);  %find
line of symmetry (3D assumption)
            mall{n}{i}=m;
            ball{n}{i}=b;
            if empty==1
                fprintf(file1,'Skipped bubble %i in frame %i, no
bubble\n',i,n);
                continue;
            elseif empty==2
                fprintf(file1,'Skipped bubble %i in frame %i, out of
frame\n',i,n);
                %continue;
            end
            [xs,ys]=points(bound2{i},m,b);  %extract useful points
            xsall{n}{i}=xs;
            ysall{n}{i}=ys;
            if length(xs)<2      %too many of these means it is picking
                disp('skip 2');  %up a lot of noise, try changing the
                fprintf(file1,'Skipped bubble %i in frame %i, too
small\n',i,n);
                continue         %thresholds in edgedetect
            end
            [xn2,yn3]=rotatebub(xs,ys,m,b); %rotate points for easier
calculations
            yn3all{n}{i}=yn3;
            xn2all{n}{i}=xn2;
            [Vtot(i),SAt(i),cylinder]=trap(xn2,yn3,dpixels); %calculate
volume and surface area
            if cylinder==1
                fprintf(file1,'Bubble %i in frame %i represented as a
cylinder\n',i,n);
            end
            Vbox=(ss(1)-2*WBbox)*(ss(2)-2*WBbox)*dpixels;
            if Vtot(i)>Vbox
                fprintf(file1,'Bubble %i in frame %i bigger than
channel\n',i,n);
            end
        end
        %calculate frame totals
        [voidf(n),Avoid(n),SAtot(n)]=outputbub(Vtot,Abub,ss,dpixels,SAt
,wratio,WBbox);
    elseif regime==2            %calculations for slug flow
        [aa]=annpoints(s,ss,bound2);        %extract useful points
        if isempty(aa{1}) || isempty(aa{2})
            Vtot=0;
            Abub=0;
            SAt=0;
        else
            %calculate volume and surface area
            [Vtot,Abub,SAt]=slugvoid(aa,ss,xline);
        end
        %calculate frame totals
        [voidf(n),Avoid(n),SAtot(n)]=outputann(Vtot,Abub,ss,dpixels,SAt
,wratio,regime,WBbox);
    else                        %calculations for annular/churn flow
        [aa]=annpoints(s,ss,bound2);        %extract useful points
        if isempty(aa{1}) || isempty(aa{2})
            Vtot=0;
            Abub=0;
            SAt=0;
        else
            %calculate volume and surface area
            [Vtot,Abub,SAt]=annvoid(aa,ss,xline,dpixels);
        end
        %calculate frame totals
        [voidf(n),Avoid(n),SAtot(n)]=outputann(Vtot,Abub,ss,dpixels,SAt
,wratio,regime,WBbox);
    end
end
if regime==1
    plotdatabub(Avoid,voidf,SAtot);  %plot data by frame number (bubbly)
end
if regime==2 || regime==3
    %plot data by frame number (slug, churn, annular)
    plotdataann(Avoid,voidf,SAtot);
end
time=toc;
disp(['This movie took ' num2str(time) ' seconds to run.']);
fprintf(file1,'-------------------------------------------------\n');
fprintf(file1,'Time averaged void fraction: %4.2f\n',mean(voidf));
fprintf(file1,'Time averaged area void fraction: %4.2f\n',mean(Avoid));
fprintf(file1,'-------------------------------------------------\n');
fprintf(file1,'This movie took %4.2f seconds to run.\n',time);
c=clock;
fprintf(file1,'This program ran at %i:%02.0f:%02.0f
on %i/%i/%i.',c(4),c(5),c(6),c(2),c(3),c(1));
fclose(file1);          %close log file
save(OutFileName)
OutFileName2=strrep(OutFileName,'.mat','');
OutFileName2=strcat(OutFileName2,'.avi');
disp('Finished!');
%end of program
```

Function 2. annpoints.m:

```
%organize points for annular flow
function [aa]=annpoints(s,ss,bound2)
%assume bubble is a cylinder
a=cell(2,s(1));                          %define size of cell
for i=1:s(1)                             %number of objects detected
    temp=bound2{i}(:,1);                 %put x values in column 1
    bound2{i}(:,1)=bound2{i}(:,2);       %put y values in column 2
    bound2{i}(:,2)=temp;
    s2=size(bound2{i});
    for j=1:s2(1)
            a{1,i}(j,1)=bound2{i}(j,1); %add x value to top
            a{1,i}(j,2)=bound2{i}(j,2); %add y value to top
            a{2,i}(j,1)=bound2{i}(j,1); %add x value to bottom
            a{2,i}(j,2)=bound2{i}(j,2); %add y value to bottom
    end
    a{1,i}=unique(a{1,i},'rows'); %eliminate repeated values for top
    a{2,i}=unique(a{2,i},'rows'); %eliminate repeated values for bottom
end
aa=cell(2,1);
for i=1:s(1)
    aa{1}=[aa{1};a{1,i}];    %create single matrix of top x,y values
    aa{2}=[aa{2};a{2,i}];    %create single matrix of bottom x,y values
end
end
```

Function 3. annvoid.m:

```
%calculate various void fraction data for annular flow
function [Vtot3,Abub3,SAt]=annvoid(aa,ss,xline,dpixels)
s2=max(size(aa{1}),size(aa{2}));        %number of unique points
Vtot3=zeros(1,ss(2));
SAt=zeros(1,ss(2));
k=0;
for i=1:ss(2)                                 %sweep across x axis
                                              %a1 is top points
                                              %a2 is bottom points
    a1=find(aa{1}(:,1)==i,s2(1));        %find top x values equal to i
    a2=find(aa{2}(:,1)==i,s2(1));        %find bottom x values equal to i
    if isempty(a1) || isempty(a2)
        continue                     %if no points found, move to next set
    end
    %calculation of prism volume (length*width*height)
    Vtot3(i)=(i-k)*(max(aa{1}(a1,2))-min(aa{2}(a2,2)))*(dpixels-
ss(1)+max(aa{1}(a1,2))-min(aa{2}(a2,2)));
    %ave void volume
    SAt(i)=2*(i-k)*(max(aa{1}(a1,2))-min(aa{2}(a2,2)))+2*(i-
k)*(dpixels-ss(1)+max(aa{1}(a1,2))-min(aa{2}(a2,2)));
    %surface area of void
    k=i;
end
a1=find(aa{1}(:,1)==xline);      %top x value bubble points on xline
a2=find(aa{2}(:,1)==xline);      %bottom x value bubble points on xline
Abub3=0;
if ~isempty(a1) && ~isempty(a2)
    d=max(aa{1}(a1,2))-min(aa{2}(a2,2));  %calculate ave width of
bubble on xline
    Abub3=d*(dpixels-ss(1)+d);             %calculate the ave area of
the bubble on xline
end
end
```

Function 4. backg.m:

```
%obtain a background image and subtract it from all frames uses concept
% that background pixels are the most common pixel in a series of
% consecutive frames, this assumption does not work as well for annular
% flow or regimes with a high density of bubbles
function [bg,frames2]=backg(frames,movs,NFrameBG,BGUsr)
s=size(frames{1});       %image size
row=s(1);
col=s(2);
if ~BGUsr
    if NFrameBG<=0
        disp('Need greater than 0 frames to obtain a background image');
    end
    n=NFrameBG;          %number of frames needed to get background image
    fr=zeros(row,col,n);     %dummy variable so we don't overwrite the
original image
    for i=1:n
        fr(:,:,i)=frames{i};
    end
    bg=zeros(row,col);
    for i=1:row
        for j=1:col
            x=double(reshape(fr(i,j,:),1,n));    %obtain all values of
one pixel over many frames
            dx=diff(x); %create vector pixel differences between frames
            y=zeros((n-3),1);
            for k=1:(n-3)
                y(k)=sum(abs(dx(k:(k+2))));      %sum differences across
4 frames
            end
            ind=find(y==min(y));                 %find least changing
(smallest difference)
            bg(i,j)=floor(mean(x(ind:(ind+3)))); %background pixel is
equal to value
        end
    end
    bg=uint8(bg);
else
    bg=BGUsr;
end
frames2=frames;
for n=1:movs(2)
    for i=1:row
        for j=1:col
            temp1=double(frames{n}(i,j));    %frame pixel value
            temp2=double(bg(i,j));           %background pixel value
            if abs(temp1-temp2)<=10          %if frame pixel is within
10 of background value
                frames2{n}(i,j)=255;      %rewrite frame pixel to white
            end
        end
    end
end
end
```

Function 5. contbound.m:

```
function [jcvbound]=contbound(cvbound)
jcvbound(1,:)=cvbound(1,:);
for i=1:length(cvbound(:,1))-1
    x1=cvbound(i,1);
    y1=cvbound(i,2);
    x2=cvbound(i+1,1);
    y2=cvbound(i+1,2);
    [joinedmatrix]=joinpoints(x1,y1,x2,y2);
    jcvbound=[jcvbound; joinedmatrix(2:end,:)];
end
x1=cvbound(end,1);
y1=cvbound(end,2);
x2=cvbound(1,1);
y2=cvbound(1,2);
[joinedmatrix]=joinpoints(x1,y1,x2,y2);
jcvbound=[jcvbound; joinedmatrix(2:end,:)];
end
```

Function 6. edgedetann.m:

```
%detect edges in image
%annular flow
function
[BW,WBbox]=edgedetann(I,ImAdjustLowAnn,ImAdjustHighAnn,CannyLowAnn,Cann
yHighAnn,CannyWidthAnn)
%detect edges of bubbles
s=size(I);
WBbox=uint8(CannyWidthAnn)+1;   %bounding box width = width of canny
algorithm +1 pixel
WBbox=double(WBbox);
I2(1:(s(1)+2*WBbox),1:s(2))=255;
I2((WBbox+1):(s(1)+WBbox),1:s(2))=I;
I2=uint8(I2);
%change threshhold values to correspond to pure liquid images
%have pixels at 50% and above changed to 100% (white)
%have pixels at 25% and below changed to 0% (black)
%apply same threshold values to all frames
J=imadjust(I2,[ImAdjustLowAnn ImAdjustHighAnn],[1 0]);  %adjust image
and make it negative
BW=edge(J,'canny',[CannyLowAnn CannyHighAnn],CannyWidthAnn);    %detect
edges in image
end
```

Function 7. edgedetect.m:

```
%detect edges in image
function
[BW,WBbox]=edgedetect(I,ImAdjustLow,ImAdjustHigh,CannyLow,CannyHigh,Can
nyWidth)
%detect edges of bubbles
s=size(I);
WBbox=uint8(CannyWidth)+1;      %bounding box width = width of canny
algorithm +1 pixel
WBbox=double(WBbox);
I2(1:(s(1)+2*WBbox),1:(s(2)+2*WBbox))=255;
I2((WBbox+1):(s(1)+WBbox),(WBbox+1):(s(2)+WBbox))=I;
I2=uint8(I2);
%change threshhold values to correspond to pure liquid images
%have pixels at 70% and above changed to 100% (white)
%have pixels at 30% and below changed to 0% (black)
%apply same threshold values to all frames
J=imadjust(I2,[ImAdjustLow ImAdjustHigh],[1 0]);    %adjust images and
makes a negative
BW=edge(J,'canny',[CannyLow CannyHigh],CannyWidth); %detect edges in
image
end
```

Function 8. fit_ellipse.m:

```
function ellipse_t = fit_ellipse( x,y,axis_handle )
% finds the best fit to an ellipse for the given set of points.
% initialize
orientation_tolerance = 1e-3;
% empty warning stack
warning( '' );
% prepare vectors, must be column vectors
x = x(:);
y = y(:);
% remove bias of the ellipse - to make matrix inversion more accurate.
(will be added later on).
mean_x = mean(x);
mean_y = mean(y);
x = x-mean_x;
y = y-mean_y;
% the estimation for the conic equation of the ellipse
X = [x.^2, x.*y, y.^2, x, y ];
if X==0
    ellipse_t.phi=[];
    return
end
con=cond(X);
if con>1e4
    ellipse_t.phi=[];
    return
end
a = sum(X)/(X'*X);
% check for warnings
if ~isempty( lastwarn )
    disp( 'stopped because of a warning regarding matrix inversion' );
    ellipse_t = struct( ...
        'a',[],...
        'b',[],...
        'phi',[],...
        'X0',[],...
        'Y0',[],...
        'X0_in',[],...
        'Y0_in',[],...
        'long_axis',[],...
        'short_axis',[],...
        'status','');
    return
end
% extract parameters from the conic equation
[a,b,c,d,e] = deal( a(1),a(2),a(3),a(4),a(5) );
% remove the orientation from the ellipse
if ( min(abs(b/a),abs(b/c)) > orientation_tolerance )
    orientation_rad = 1/2 * atan( b/(c-a) );
    cos_phi = cos( orientation_rad );
    sin_phi = sin( orientation_rad );
    [a,b,c,d,e] = deal(...
        a*cos_phi^2 - b*cos_phi*sin_phi + c*sin_phi^2,...
        0,...
```

```
            a*sin_phi^2 + b*cos_phi*sin_phi + c*cos_phi^2,...
            d*cos_phi - e*sin_phi,...
            d*sin_phi + e*cos_phi );
        [mean_x,mean_y] = deal( ...
            cos_phi*mean_x - sin_phi*mean_y,...
            sin_phi*mean_x + cos_phi*mean_y );
    else
        orientation_rad = 0;
        cos_phi = cos( orientation_rad );
        sin_phi = sin( orientation_rad );
    end
    % check if conic equation represents an ellipse
    test = a*c;
    status = '';
    % if we found an ellipse return it's data
    if (test>0)
        % make sure coefficients are positive as required
        if (a<0), [a,c,d,e] = deal( -a,-c,-d,-e ); end
        % final ellipse parameters
        X0          = mean_x - d/2/a;
        Y0          = mean_y - e/2/c;
        F           = 1 + (d^2)/(4*a) + (e^2)/(4*c);
        [a,b]       = deal( sqrt( F/a ),sqrt( F/c ) );
        long_axis   = 2*max(a,b);
        short_axis  = 2*min(a,b);
        % rotate the axes backwards to find the center point of the
    original TILTED ellipse
        R           = [ cos_phi sin_phi; -sin_phi cos_phi ];
        P_in        = R * [X0;Y0];
        X0_in       = P_in(1);
        Y0_in       = P_in(2);
        % pack ellipse into a structure
        ellipse_t = struct( ...
            'a',a,...
            'b',b,...
            'phi',orientation_rad,...
            'X0',X0,...
            'Y0',Y0,...
            'X0_in',X0_in,...
            'Y0_in',Y0_in,...
            'long_axis',long_axis,...
            'short_axis',short_axis,...
            'status','' );
    else
        % report an empty structure
        ellipse_t = struct( ...
            'a',[],...
            'b',[],...
            'phi',[],...
            'X0',[],...
            'Y0',[],...
            'X0_in',[],...
            'Y0_in',[],...
            'long_axis',[],...
            'short_axis',[],...
            'status',status );
    end
    % check if we need to plot an ellipse with it's axes.
    if (nargin>2) && ~isempty( axis_handle ) && (test>0)
        % rotation matrix to rotate the axes with respect to an angle phi
        R = [ cos_phi sin_phi; -sin_phi cos_phi ];
        % the axes
        ver_line        = [ [X0 X0]; Y0+b*[-1 1] ];
        horz_line       = [ X0+a*[-1 1]; [Y0 Y0] ];
        new_ver_line    = R*ver_line;
        new_horz_line   = R*horz_line;
        % the ellipse
        theta_r         = linspace(0,2*pi);
        ellipse_x_r     = X0 + a*cos( theta_r );
        ellipse_y_r     = Y0 + b*sin( theta_r );
        rotated_ellipse = R * [ellipse_x_r;ellipse_y_r];
        % draw
        hold_state = get( axis_handle,'NextPlot' );
        set( axis_handle,'NextPlot','add' );
        plot( new_ver_line(1,:),new_ver_line(2,:),'r' );
        plot( new_horz_line(1,:),new_horz_line(2,:),'r' );
        plot( rotated_ellipse(1,:),rotated_ellipse(2,:),'r' );
        set( axis_handle,'NextPlot',hold_state );
    end
```

Function 9. GetBgImage.m:

```
function
[bg]=GetBgImage(BGFileName,BGFrameNumber,AngleRotate,CropUpRow,CropDnRo
w,CropLCol,CropRCol)
mov=aviread(BGFileName,BGFrameNumber);
frames(:,:,:)=mov(1).cdata;
framesr=imrotate(frames,AngleRotate);
bg=framesr(CropUpRow:CropDnRow,CropLCol:CropRCol,:);
end
```

Function 10. identify.m:

```
%identify objects in image
function
[bound,bound2,dpixels,xline,s,ss,wratio,linebub]=identify(BW,WBbox,widt
h,depth,xfrac,regime)
%identify objects in image
BW2=imfill(BW,'holes');           %fill in holes in image
bound=bwboundaries(BW2);          %find boundaries of shapes in image
                                  %bound{i}(:,1)=y values
                                  %bound{i}(:,2)=x values
s=size(bound);                    %determines number of shapes, use s(1)
ss=size(BW2);                     %determines size of image, use ss(1)
if s(1)>1             %if more than 2 objects, see if inside one another
    for i=1:s(1)
        if isempty(bound{i})
            continue;
        end
     msk1=poly2mask(bound{i}(:,1),bound{i}(:,2),ss(2),ss(1)); %object 1
        for j=1:s(1)
            if i==j                     %use different objects
                continue;
            end
            if isempty(bound{j})
                continue;
            end
     msk2=poly2mask(bound{j}(:,1),bound{j}(:,2),ss(2),ss(1)); %object 2
            a=msk1(msk2);
            inside=0;
            if ~isempty(a)
               inside=all(a); %determine if object 2 is inside object 1
            end
            if inside                %if inside
                bound{j}=[];         %delete coordinates of inside object
            end
        end
    end
end
s=size(bound);
if regime==1 || regime==2
    for i=1:s(1)
        bound{i}=bound{i}-WBbox;
    end
else
    for i=1:s(1)
        bound{i}(:,1)=bound{i}(:,1)-WBbox;
    end
end
% Convex Hull routine here to modify bound.
cvbound = cell(1,s(1)); %Allocate cell array for Convex Hull Boundaries
jcvbound=cell(1,s(1));
linebub=cell(1,s(1));
if regime==1
    for BoundIndex = 1:s(1)
```

```
        collinear8 = @(varargin) rank(cat(1,varargin{:}) -
circshift(cat(1,varargin{:}),1)) == 1;
        if ~collinear8(bound{BoundIndex})
            CVIndex =
convhull(bound{BoundIndex}(:,2),bound{BoundIndex}(:,1)); %obtain
convhull indices
            cvbound{BoundIndex}(:,:)=bound{BoundIndex}(CVIndex,:);%only
use convhull points
            jcvbound{BoundIndex}=contbound(cvbound{BoundIndex});  %fill
in remaining to get a closed object
            linebub=false;
        else
            jcvbound{BoundIndex}=[];
            linebub=true;
        end
    end
end
bound2=cell(1,s(1));
if regime==1
    for i=1:s(1)
        bound2{i}(:,1)=(ss(1)-2*WBbox)-jcvbound{i}(:,1); %correct y
values so image isn't flipped
        bound2{i}(:,2)=jcvbound{i}(:,2);    %add x values to new matrix
    end
else
    for i=1:s(1)
        bound2{i}(:,1)=(ss(1)-2*WBbox)-bound{i}(:,1);
        bound2{i}(:,2)=bound{i}(:,2);
    end
end
bound2=bound2';                             %transpose cell
wratio=(ss(1)-2*WBbox)/width; %ratio of pixels to microns in channel
width
dpixels=wratio*depth;                       %pixels in channel depth
if regime==1 || regime==2
    xline=floor((ss(2)-2*WBbox)*xfrac+WBbox);   %vertical plane for
cross sectional area void fraction
else
    xline=floor((ss(2)-2*WBbox-1)*xfrac+WBbox);
end
if xline==0
    xline=1;
end
end
```

Function 11. inputin.m:

```
%input information from the user
function
[filename,width,depth,regime,xfrac,StartFrame,EndFrame,OutFileName,Angl
eRotate,CropUpRow,CropDnRow,CropLCol,CropRCol,NFrameBG,BGFileName,BGFra
meNumber,ImAdjustLow,ImAdjustHigh,ImAdjustLowAnn,ImAdjustHighAnn,CannyL
ow,CannyHigh,CannyWidth,CannyLowAnn,CannyHighAnn,CannyWidthAnn,findback
g]=inputin()
%collect channel dimensions and video information
RunName='/home/citadel/b/shared/Sidharth/ONR/TwoPhaseAirWater/TestCell2
/TestSet6/Videos/TC2TS6_C001S0';
movnum='222';
ExtName='avi';                %video file type
filename=[RunName movnum '/TC2TS6_C001S0' movnum '.' ExtName];
StartFrame = 1;               %first frame
EndFrame = 50;                %last frame
OutFileName=['/home/citadel/b/shared/Sidharth/ONR/TwoPhaseAirWater/Test
Cell2/TestSet6/ProcVideos/TC2TS6_C0001S0' movnum '_' num2str(StartFrame)
'to' num2str(EndFrame) '.mat'];     %save file name
AngleRotate=0;             %frame rotation angle
CropUpRow=361;                 %top pixel
CropDnRow=913;                %bottom pixel
CropLCol=1;                 %left pixel
CropRCol=512;                 %right pixel
NFrameBG=40;            %number of frames to use to find background image
BGFileName='/home/citadel/b/willi319/ONRMicroChannel/TwoPhaseAirWater/T
estCell1/TestSet2/Videos/TC1TS2_C001S0015/TC1TS2_C001S0015.avi';  %file
name of background image
BGFrameNumber=1;              %frame number of background image
findbackg=0;                  %user supplied background? yes=1, no=0
ImAdjustLow=0.3;              %lower threshold for image adjusting
ImAdjustHigh=0.7;             %upper threshold for image adjusting
ImAdjustLowAnn=0.25;          %lower threshold for image adjusting,
annular flow
ImAdjustHighAnn=0.5;          %upper threshold for image adjusting,
annular flow
CannyLow=0.25;                %lower threshold for canny edge detection
CannyHigh=0.85;               %upper threshold for canny edge detection
CannyWidth=2;                 %sigma value for canny edge detection
CannyLowAnn=0.1;              %lower threshold for canny edge detection,
annular flow
CannyHighAnn=0.8;             %upper threshold for canny edge detection,
annular flow
CannyWidthAnn=1.5;            %sigma value for canny edge detection,
annular flow
width=780;                    %channel width in microns
depth=780;                    %channel depth in microns
regime=1;                     %flow regime, 1 for bubbly flow, 2 for slug
flow, 3 for annular/churn flow
xfrac=0;                      %location for area void fraction
end
```

Function 12. joinpoints.m:

```
function [joinedmatrix]=joinpoints(x1,y1,x2,y2)
joinedmatrix=[x1 y1];
%isneighbor
if abs(x1-x2)<=1 && abs(y1-y2)<=1    %they are neighboring points
    joinedmatrix=[joinedmatrix; x2 y2];
else
    if x1==x2        %vertical line
        deltay=y2-y1;
        ynext=y1+sign(deltay);
        joinedmatrix=[joinedmatrix; x1 ynext];
        while ynext~=y2
            ynext=ynext+sign(deltay);
            joinedmatrix=[joinedmatrix; x1 ynext];
        end
    else        %find slope and make a line
        slope=(y2-y1)/(x2-x1);
        if abs(slope)<1
            %xtrace
            deltax=x2-x1;
            xnext=x1+sign(deltax);
            ynext=round(slope*(xnext-x1))+y1;
            joinedmatrix=[joinedmatrix; xnext ynext];
            while xnext~=x2
                xnext=xnext+sign(deltax);
                ynext=round(slope*(xnext-x1))+y1;
                joinedmatrix=[joinedmatrix; xnext ynext];
            end
        else
            %ytrace
            deltay=y2-y1;
            ynext=y1+sign(deltay);
            xnext=round(1/slope*(ynext-y1))+x1;
            joinedmatrix=[joinedmatrix; xnext ynext];
            while ynext~=y2
                ynext=ynext+sign(deltay);
                xnext=round(1/slope*(ynext-y1))+x1;
                joinedmatrix=[joinedmatrix; xnext ynext];
            end
        end
    end
end
end
```

Function 13. movieplot1.m:

```
%capture images for movie
%can capture rotated and cropped images
%can capture images with objects plotted on top
%objects plotted on top after convexhull applied
function [allframes]=movieplot1(I,bound,xline,s,ss,WBbox)
%plot objects in image, one at a time
if ~isempty(bound)          %if bound is not empty
    for i=1:s(1)
        bound{i}(:,1)=(ss(1)-2*WBbox)-bound{i}(:,1);  %correct y values
so image isn't flipped
    end
end
imshow(I);                              %display original image
%allframes=getframe;                     %turn on if you want rotated and
cropped images only
hold on;                  %plot multiple shapes on same graph over image
axis equal;               %make axes equal in value so graph is easy to read
for i=1:s(1)
    b=mod(i,7);
    if b==0
        plot(bound{i}(:,2),bound{i}(:,1),'r');  %plot shape in red
    elseif b==1
        plot(bound{i}(:,2),bound{i}(:,1),'b');  %plot shape in blue
    elseif b==2
        plot(bound{i}(:,2),bound{i}(:,1),'g');  %plot shape in green
    elseif b==3
        plot(bound{i}(:,2),bound{i}(:,1),'y');  %plot shape in yellow
    elseif b==4
        plot(bound{i}(:,2),bound{i}(:,1),'m');  %plot shape in magenta
    else
        plot(bound{i}(:,2),bound{i}(:,1),'c');  %plot shape in cyan
    end
end
plot([xline,xline],[0,ss(1)],'--r');
pause(0.1);
allframes=getframe;         %turn on if you want images with objects
plotted on top
hold off;
end
```

Function 14. outputann.m:

```
%calculate output from annular flow calculations
function
[voidf3,Avoid3,SAtot]=outputann(Vtot3,Abub3,ss,dpixels,SAt,wratio,regim
e,WBbox)
if regime==2
    Vbox=(ss(1)-2*WBbox)*(ss(2)-2*WBbox)*dpixels;           %volume of
channel, slug flow [pixels^3]
else
    Vbox=(ss(1)-2*WBbox)*ss(2)*dpixels;                 %volume of
channel, annular flow [pixels^3]
end
%calculation output
Vvoid3=sum(Vtot3);                %volume of all voids [pixels^3], [ave]
voidf3=Vvoid3/Vbox*100;           %void fraction, [ave]
%area void fraction output
area=dpixels*(ss(1)-2*WBbox);           %cross sectional area [pixels^2]
Avoid3=Abub3/area*100;                  %ave possible area void fraction
SAtot=sum(SAt);                  %surface area of voids [pixels^2]
SAtot=SAtot/Vvoid3;              %surface area to volume ratio [1/pixels]
end
```

Function 15. outputbub.m:

```
%calculate output from bubbly/slug flow calculations
function
[voidf3,Avoid,SAtot]=outputbub(Vtot3,Abub,ss,dpixels,SAt,wratio,WBbox)
Vbox=(ss(1)-2*WBbox)*(ss(2)-2*WBbox)*dpixels;   %volume of channel
[pixels^3]
%trapezoid rule output
Vvoid3=sum(Vtot3);          %volume of all voids [pixels^3], [trapezoid]
voidf3=Vvoid3/Vbox*100;     %void fraction, [trapezoid]
%area void fraction output
Atot=sum(Abub);                  %total void at cross section [pixels^2]
area=dpixels*(ss(1)-2*WBbox);    %cross sectional area [pixels^2]
Avoid=Atot/area*100;             %area void fraction
SAtot=sum(SAt);                  %surface area of all voids [pixels^2]
SAtot=SAtot/Vbox;                %surface area to volume ratio [1/pixels]
end
```

Function 16. plotdataann.m:

```
%plot output data for annular flow
function []=plotdataann(Avoid1,voidf,SAtot)
figure
plot(Avoid1,'-ob');
xlabel('Frame Number','fontsize',16);
ylabel('Area void fraction [max] (%)','fontsize',16);
figure
plot(voidf,'-ob');
xlabel('Frame Number','fontsize',16);
ylabel('Volume void fraction [max] (%)','fontsize',16);
figure
plot(SAtot,'-ob');
xlabel('Frame Number','fontsize',16);
ylabel('Surface area concentration [1/pixels]','fontsize',16);
end
```

Function 17. plotdatabub.m:

```
%plot output data for bubbly flow
function []=plotdatabub(Avoid,voidf3,SAtot)
figure
plot(Avoid,'-ob');
xlabel('Frame Number','fontsize',16);
ylabel('Area void fraction (%)','fontsize',16);
figure
plot(voidf3,'-ob');
xlabel('Frame Number','fontsize',16);
ylabel('Volume void fraction [trapezoid] (%)','fontsize',16);
figure
plot(SAtot,'-ob');
xlabel('Frame Number','fontsize',16);
ylabel('Surface area concentration [1/pixels]','fontsize',16);
end
```

Function 18. points.m:

```
%obtain points above line of symmetry
function [xs,ys]=points(bound2,m,b)
k=1;
xs=[];
ys=[];
s=size(bound2);           %determines number of edge points for bubble
for j=1:s(1)
    temp=m*bound2(j,2)+b; %determine if edge point is above line of
symmetry
    if bound2(j,1)>temp
        xs(k)=bound2(j,2); %x value if point is above line of symmetry
        ys(k)=bound2(j,1); %y value if point is above line of symmetry
        k=k+1;
    end
end
a=unique([xs' ys'],'rows');
if isempty(a)
    xs=0;
    ys=0;
else
    xs=a(:,1)';
    ys=a(:,2)';
end
end
```

Function 19. readrotate.m:

```
%read frames of movie and rotate images
function
[frames2,movs,movcolor]=readrotate(filename,StartFrame,EndFrame,AngleRo
tate,CropUpRow,CropDnRow,CropLCol,CropRCol)
%mov=aviread(filename,StartFrame:EndFrame);      %read in avi file
%movs=size(mov);              %number of frames
movs=[1 (EndFrame-StartFrame+1)];
frames=cell(1,movs(2));
frames2=cell(1,movs(2));
for i=1:movs(2)
    fprintf('...');
    if mod(i,20)==0
        a=100*i/movs(2);
        fprintf('%4.1f%% complete\n',a);
    end
    mov=aviread(filename,(StartFrame-1+i));
    frames{i}(:,:,:)=mov.cdata;        %pull out frame image from movie
    framesr=imrotate(frames{i},AngleRotate);       %rotate frame image
    frames2{i}=framesr(CropUpRow:CropDnRow,CropLCol:CropRCol,:);   %crop
frame image
end
fprintf('100%% complete\n');
mov=aviread(filename,StartFrame);
movcolor=mov.colormap;   %obtain movie colormap
end
```

Function 20. rotatebub.m:

```
%rotate points in bubble
function [xn2,yn3]=rotatebub(xs,ys,m,b)
s=length(xs);   %determine number of edge points above line of symmetry
xb=zeros(1,s);
yb=zeros(1,s);
xb=(m*ys+xs-m*b)/(m^2+1);  %rotation calculations
yb=(m^2*ys+m*xs+b)/(m^2+1);
xs1=min(xb);
ys1=min(yb);
if ((yb(2)-yb(1))/(xb(2)-xb(1)))<0
    ys1=max(yb);
end
xn=sqrt((ys1-yb).^2+(xs1-xb).^2);       %rotated x values
yn=sqrt((ys-yb).^2+(xs-xb).^2);         %rotated y values
a=[xn' yn'];
s=size(a);
anew=zeros(s(1),2);
for j=1:s(1)
    a1=find(a(:,1)==a(j,1),s(1));
    anew(j,1)=a(j,1);                %for x values with multiple y values
    anew(j,2)=max(a(a1,2));          %use only the maximum y value
end
anew=unique(anew,'rows');            %delete repeated pairs of points
xn2=anew(:,1)';                      %unique x values
yn3=anew(:,2)';                      %unique y values
end
```

Function 21. slugvoid.m:

```
%calculate various void fraction data for annular flow
function [Vtot3,Abub3,SAt]=slugvoid(aa,ss,xline)
s2=max(size(aa{1}),size(aa{2}));        %number of unique points
Vtot3=zeros(1,ss(2));
SAt=zeros(1,ss(2));
k=0;
for i=1:ss(2)                          %sweep across x axis
                                       %a1 is top points
                                       %a2 is bottom points
    a1=find(aa{1}(:,1)==i,s2(1));      %find top x values equal to i
    a2=find(aa{2}(:,1)==i,s2(1));      %find bottom x values equal to i
    if isempty(a1) || isempty(a2)
        continue                  %if no points found, move to next set
    end
    Vtot3(i)=(i-k)*pi/4*(max(aa{1}(a1,2))-min(aa{2}(a2,2)))^2;    %ave
void fraction
    SAt(i)=(i-k)*pi*(max(aa{1}(a1,2))-min(aa{2}(a2,2)));   %surface
area of void
    k=i;
end
a1=find(aa{1}(:,1)==xline);      %top x value bubble points on xline
a2=find(aa{2}(:,1)==xline);      %bottom x value bubble points on xline
Abub3=0;
if isempty(a1)==0 && isempty(a2)==0
    d=max(aa{1}(a1,2))-min(aa{2}(a2,2));  %calculate ave width of
bubble on xline
    Abub3=pi/4*d^2;  %calculate the ave area of the bubble on xline
end
end
```

Function 22. symmetry.m:

```
%find best line of symmetry for object
function [Abub,m,b,empty]=symmetry(bound2,xline,ss)
%determine best line of symmetry for bubble
bound2=unique(bound2,'rows');        %delete repeated pairs of points
x=bound2(:,2);                          %x values for bubble
y=bound2(:,1);                          %y values for bubble
ellipse_t=fit_ellipse(x,y); %use best fit ellipse program to determine
                            %best line of symmetry
a=find(bound2(:,2)==xline); %find bubble points on xline to use for
area void fraction
Abub=0;
if ~isempty(a)
    d=max(bound2(a,1))-min(bound2(a,1));%calculate diameter of bubble
on xline
    Abub=pi/4*d^2;          %calculate the area of the bubble on xline
end
empty=0;
if isempty(ellipse_t.phi)
    disp('skip 1');
    empty=1;
    Abub=0;
    m=0;
    b=0;
    return
end
x0=ellipse_t.X0_in;             %x coordinate of center of ellipse
y0=ellipse_t.Y0_in;             %y coordinate of center of ellipse
alpha=ellipse_t.phi;            %orientation angle of ellipse
if abs(x0)>2*ss(2) || abs(y0)>2*ss(1)
    empty=2;
    Abub=0;
    m=0;
    b=0;
    return
end
m=-tan(alpha);                  %slope of line of symmetry
b=y0+tan(alpha)*x0;             %y intercept of best line of symmetry
end
```

Function 23. trap.m:

```
%volume rotation using trapezoid rule
function [Vtot,SAt,cylinder]=trap(x,y,dpixels)
s=length(x);
V=zeros(1,s-1);
SA=zeros(1,s-1);
ff=max(y);
cylinder=false;
if 2*ff>dpixels              %determine if bubble should be calculated
    cylinder=true;           %as a cylinder
end

if ~cylinder;    %bubble height smaller than channel depth, spheroid
    for j=1:(s-1)%trapezoid rule and pappus theorem for volume and area
        V(j)=pi/3*abs(x(j+1)-x(j))*((y(j))^2+y(j)*y(j+1)+(y(j+1))^2);
        SA(j)=pi*(y(j)+y(j+1))*sqrt((y(j)-y(j+1))^2+(x(j+1)-x(j))^2);
    end
else             %bubble height larger than channel depth, cylinder
    for j=1:(s-1)        %trapezoid rule times height
        V(j)=abs(x(j+1)-x(j))*(y(j)+y(j+1))*dpixels;
        SA(j)=2*abs(x(j+1)-x(j))*(y(j)+y(j+1))+2*dpixels*sqrt((y(j)-
y(j+1))^2+(x(j+1)-x(j))^2);
    end
end
Vtot=sum(V);        %bubble volume [trapezoid rule] [pixels^3]
SAt=sum(SA);        %bubble surface area [pixels^2]
end
```

Appendix B MATLAB Script for Numerical Simulation of Electrical Impedance

A MATLAB script was developed to simulate the response of a miniature impedance-based void fraction meter as discussed in Section 4.1.2. The following script generates a 3D domain based on high-speed videos and solves for the impedance across two electrodes. It is split up into several functions; Table B.1 displays the function number, name, a description, and the page number where it is found. Several functions are omitted due to their repetitive nature and length. These are run using the data generated from the image processing scripts.

Table B.1. A list of all of the functions for numerical simulations.

Function Number	Function Name	Description	Page Number
24	gambitjou.m	Generates a 3D mesh from images for bubbly flow	197
25	gambitjou2.m	Generates a 3D mesh from images for slug flow	201
26	gambitjou3.m	Generates a 3D mesh from images for annular flow	204
27	lap3dsolver.m	3D Laplace solver to find the resistance	207
28	lap3dsolver2.m	3D Laplace solver to find the capacitance	212
29	tdma.m	TDMA solver	217
30	tdmaline.m	Line-by-line TDMA solver	218

Function 24. gambitjou.m:

```
%use for bubbly flow
%run using data generated from voidfrac.m
k=logical(false((ss(1)-2*WBbox),(ss(2)-2*WBbox),dpixels));
siz=size(xsall{frame});               %number of bubbles, use siz(1)
for j=1:siz(1)                        %loop for each bubble
    m=mall{frame}{j};                 %slope of line of symmetry
    b=ball{frame}{j};                 %y intercept of line of symmetry
    x=xsall{frame}{j};                %x coordinates of bubble
    y=ysall{frame}{j};                  %y coordinates of bubble
    xb=(m*y+x-m*b)/(m^2+1);             %x coordinate of rotation
    yb=(m^2*y+m*x+b)/(m^2+1);           %y coordinate of rotation
    yn3=sqrt((y-yb).^2+(x-xb).^2);  %magnitude of point from line of
symmetry
    si=size(x);                         %number of points, use si(2)
    minx=min(x);                        %leftmost point
    maxx=max(x);                        %rightmost point
    p1=[0 minx-1 m*(minx-1)+b];         %point on line of rotation
    p2=[0 maxx+1 m*(maxx+1)+b];         %point on line of rotation
    p=rand(1,3);                        %random point in 3D
    r=cross(p-p1,p2-p1);           %line perpendicular to line of rotation
    s=cross(r,p2-p1);              %line perpendicular to line of rotation
    r=r/sqrt(dot(r,r));                 %normalize vector
    s=s/sqrt(dot(s,s));                 %normalize vector
    tempx=zeros(si(2),1);
    tempy=zeros(si(2),1);
    tempz=zeros(si(2),1);
    for i=1:si(2)
        theta=0:(2*pi/180):(2*pi);          %angles of rotation
        for n=1:180
            %calculate point in 3D space
            tempz=yn3(i)*cos(theta(n))*r(1)+yn3(i)*sin(theta(n))*s(1);
%z coordinate
            tempx=xb(i)+yn3(i)*cos(theta(n))*r(2)+yn3(i)*sin(theta(n))*
s(2); %x coordinate
            tempy=yb(i)+yn3(i)*cos(theta(n))*r(3)+yn3(i)*sin(theta(n))*
s(3); %y coordinate
            %translate into cell coordinates
            tempy=(ss(1)-WBbox)-round(tempy); %nearest cell in y
direction
            tempz=round(tempz+dpixels/2); %nearest cell in z direction
            tempx=round(tempx);           %nearest cell in x direction
            if tempx<1
                tempx=1;
            end
            if tempx>(ss(2)-2*WBbox)
                tempx=ss(2)-2*WBbox;
            end
            if tempy<1
                tempy=1;
            end
            if tempy>(ss(1)-2*WBbox)
                tempy=ss(1)-2*WBbox;
            end
            k(tempy,tempx,tempz)=true;
        end
    end
end
%close boundaries and fill holes
se=ones(5);
se([1 1 end end],[1 end 1 end])=0; %create morphological structuring
element
k4=logical(false((ss(1)-2*WBbox),(ss(2)-
2*WBbox),dpixels)); %preallocate
```

```
for i=1:dpixels                         %loop over each z plane
    k2=imdilate(k(:,:,i),se);           %dilate image to form closed object
    k3=imfill(k2,'holes');              %fill objects
    k4(:,:,i)=imerode(k3,se);%erode image so boundary is back to normal
end
clear k2 k3 k
%k4 is 3D logical matrix describing bubble areas
%1 indicates part of bubble, 0 is background
%downsample to make calculations faster
%transform cube of 8 voxels into one new voxel
N=size(k4);
ax1=mod(N(1),2);
ax2=mod(N(2),2);
ax3=mod(N(3),2);
k5=logical(false(floor(N(1)/2)+ax1,floor(N(2)/2)+ax2,floor(N(3)/2)+ax3)
);
for i=1:2:(N(1)-ax1)
    for j=1:2:(N(2)-ax2)
        for k=1:2:(N(3)-ax3)
            k5(round(i/2),round(j/2),round(k/2))=round(mean([k4(i,j,k)
k4(i+1,j,k) k4(i,j+1,k) k4(i+1,j+1,k) k4(i,j,k+1) k4(i+1,j,k+1)
k4(i+1,j+1,k+1) k4(i,j+1,k+1)]));
        end
    end
end
if ax1
    i=N(1);
    for j=1:2:(N(2)-ax2)
        for k=1:2:(N(3)-ax3)
            k5(floor(N(1)/2)+ax1,round(j/2),round(k/2))=round(mean([k4(
i,j,k) k4(i,j+1,k) k4(i,j,k+1) k4(i,j+1,k+1)]));
        end
    end
end
if ax2
    j=N(2);
    for i=1:2:(N(1)-ax1)
        for k=1:2:(N(3)-ax3)
            k5(round(i/2),floor(N(2)/2)+ax2,round(k/2))=round(mean([k4(
i,j,k) k4(i+1,j,k) k4(i,j,k+1) k4(i+1,j,k+1)]));
        end
    end
end
if ax3
    k=N(3);
    for i=1:2:(N(1)-ax1)
        for j=1:2:(N(2)-ax2)
            k5(round(i/2),round(j/2),floor(N(3)/2)+ax3)=round(mean([k4(
i,j,k) k4(i+1,j,k) k4(i,j+1,k) k4(i+1,j+1,k)]));
        end
    end
end
if ax1 && ax2
    i=N(1);
    j=N(2);
    for k=1:2:(N(3)-ax3)
        k5(floor(N(1)/2)+ax1,floor(N(2)/2)+ax2,round(k/2))=round(mean([
k4(i,j,k) k4(i,j,k+1)]));
    end
end
if ax1 && ax3
    i=N(1);
    k=N(3);
    for j=1:2:(N(2)-ax2)
        k5(floor(N(1)/2)+ax1,round(j/2),floor(N(3)/2)+ax3)=round(mean([
k4(i,j,k) k4(i,j+1,k)]));
    end
end
if ax2 && ax3
    j=N(2);
    k=N(3);
    for i=1:2:(N(1)-ax1)
        k5(round(i/2),floor(N(2)/2)+ax2,floor(N(3)/2)+ax3)=round(mean([
k4(i,j,k) k4(i+1,j,k)]));
    end
end
if ax1 && ax2 && ax3
     k5(floor(N(1)/2)+ax1,floor(N(2)/2)+ax2,floor(N(3)/2)+ax3)=k4(N(1),N
(2),N(3));
end
```

Function 25. gambitjou2.m:

```
%use for slug flow
%run using data generated from voidfrac.m
k=logical(false((ss(1)-2*WBbox),(ss(2)-2*WBbox),dpixels));
siz=size(coords{frame});
temp=[];
for i=1:siz(1)
    temp=[temp;coords{frame}{i}];   %put all points into one matrix
end
x=temp(:,2);                        %x coordinates of bubble
y=temp(:,1);                        %y coordinates of bubble
si=size(x);
minx=min(x);                        %leftmost point
maxx=max(x);                        %rightmost point
for j=0:(ss(2)-2*WBbox)+1           %loop for each x point
    a1=find(x==j);
    if isempty(a1)
        continue
    end
    ymin=min(y(a1));
    ymax=max(y(a1));
    xb=j;
    yb=(ymin+ymax)/2;
    p1=[0 minx-1 yb];     %point on line of rotation
    p2=[0 maxx+1 yb];     %point on line of rotation
    p=rand(1,3);                    %random point in 3D
    r=cross(p-p1,p2-p1);       %line perpendicular to line of rotation
    s=cross(r,p2-p1);          %line perpendicular to line of rotation
    r=r/sqrt(dot(r,r));             %normalize vector
    s=s/sqrt(dot(s,s));             %normalize vector
    yn3=abs(y(a1)-yb);
    si=size(yn3);
    tempx=zeros(si(1),1);
    tempy=zeros(si(1),1);
    tempz=zeros(si(1),1);
    for i=1:si(1)
        theta=0:(2*pi/500):(2*pi);                 %angles of rotation
        for n=1:500
            %calculate point in 3D space
            tempz=yn3(i)*cos(theta(n))*r(1)+yn3(i)*sin(theta(n))*s(1);
%z coordinate
            tempx=xb; %x coordinate
            tempy=yb+yn3(i)*cos(theta(n))*r(3)+yn3(i)*sin(theta(n))*s(3
); %y coordinate
            %translate into cell coordinates
            tempy=(ss(1)-WBbox)-round(tempy); %nearest cell in y
direction
            tempz=round(tempz+dpixels/2); %nearest cell in z direction
            tempx=round(tempx);           %nearest cell in x direction
            if tempx<1
                tempx=1;
            end
            if tempx>(ss(2)-2*WBbox)
                tempx=ss(2)-2*WBbox;
            end
            if tempy<1
                tempy=1;
            end
            if tempy>(ss(1)-2*WBbox)
                tempy=ss(1)-2*WBbox;
            end
            if tempz<1
                tempz=1;
            end
            if tempz>dpixels
                tempz=dpixels;
            end
            k(tempy,tempx,tempz)=true;
        end
    end
end
%close boundaries and fill holes
se=ones(5);
se([1 1 end end],[1 end 1 end])=0; %create morphological structuring
element
k4=logical(false((ss(1)-2*WBbox),(ss(2)-
2*WBbox),dpixels)); %preallocate
for i=1:dpixels                     %loop over each z plane
    k2=imdilate(k(:,:,i),se);       %dilate image to form closed object
    k3=imfill(k2,'holes');          %fill objects
    k4(:,:,i)=imerode(k3,se);%erode image so boundary is back to normal
end
clear k2 k3 k
%k4 is 3D logical matrix describing bubble areas
%1 indicates part of bubble, 0 is background
%downsample to make calculations faster
%transform cube of 8 voxels into one new voxel
N=size(k4);
ax1=mod(N(1),2);
ax2=mod(N(2),2);
ax3=mod(N(3),2);
k5=logical(false(floor(N(1)/2)+ax1,floor(N(2)/2)+ax2,floor(N(3)/2)+ax3)
);
for i=1:2:(N(1)-ax1)
    for j=1:2:(N(2)-ax2)
        for k=1:2:(N(3)-ax3)
            k5(round(i/2),round(j/2),round(k/2))=round(mean([k4(i,j,k)
k4(i+1,j,k) k4(i,j+1,k) k4(i+1,j+1,k) k4(i,j,k+1) k4(i+1,j,k+1)
k4(i+1,j+1,k+1) k4(i,j+1,k+1)]));
        end
    end
end
if ax1
    i=N(1);
    for j=1:2:(N(2)-ax2)
        for k=1:2:(N(3)-ax3)
            k5(floor(N(1)/2)+ax1,round(j/2),round(k/2))=round(mean([k4(
i,j,k) k4(i,j+1,k) k4(i,j,k+1) k4(i,j+1,k+1)]));
        end
        end
    end
    if ax2
        j=N(2);
        for i=1:2:(N(1)-ax1)
            for k=1:2:(N(3)-ax3)
                k5(round(i/2),floor(N(2)/2)+ax2,round(k/2))=round(mean([k4(
i,j,k) k4(i+1,j,k) k4(i,j,k+1) k4(i+1,j,k+1)]));
            end
        end
    end
    if ax3
        k=N(3);
        for i=1:2:(N(1)-ax1)
            for j=1:2:(N(2)-ax2)
                k5(round(i/2),round(j/2),floor(N(3)/2)+ax3)=round(mean([k4(
i,j,k) k4(i+1,j,k) k4(i,j+1,k) k4(i+1,j+1,k)]));
            end
        end
    end
    if ax1 && ax2
        i=N(1);
        j=N(2);
        for k=1:2:(N(3)-ax3)
            k5(floor(N(1)/2)+ax1,floor(N(2)/2)+ax2,round(k/2))=round(mean([
k4(i,j,k) k4(i,j,k+1)]));
        end
    end
    if ax1 && ax3
        i=N(1);
        k=N(3);
        for j=1:2:(N(2)-ax2)
            k5(floor(N(1)/2)+ax1,round(j/2),floor(N(3)/2)+ax3)=round(mean([
k4(i,j,k) k4(i,j+1,k)]));
        end
    end
    if ax2 && ax3
        j=N(2);
        k=N(3);
        for i=1:2:(N(1)-ax1)
            k5(round(i/2),floor(N(2)/2)+ax2,floor(N(3)/2)+ax3)=round(mean([
k4(i,j,k) k4(i+1,j,k)]));
        end
    end
    if ax1 && ax2 && ax3
         k5(floor(N(1)/2)+ax1,floor(N(2)/2)+ax2,floor(N(3)/2)+ax3)=k4(N(1),N
(2),N(3));
    end
```

Function 26. gambitjou3.m:

```
%use for annular flow
%run using data generated from voidfrac.m
k=logical(false((ss(1)-2*WBbox),(ss(2)-2*WBbox),dpixels));
siz=size(coords{frame});
temp=[];
for i=1:siz(1)
    temp=[temp;coords{frame}{i}];   %put all points into one matrix
end
x=temp(:,2);                        %x coordinates of bubble
y=temp(:,1);                        %y coordinates of bubble
si=size(x);
minx=min(x);                        %leftmost point
maxx=max(x);                        %rightmost point
for j=0:(ss(2)-2*WBbox)+1           %loop for each x point
    a1=find(x==j);
    if isempty(a1)
        continue
    end
    ymin=min(y(a1));
    ymax=max(y(a1));
    xb=j;
    yb=(ymin+ymax)/2;
    yhat=(ymax-ymin)/2;
    zhat=(dpixels-(ss(1)-2*WBbox)+ymax-ymin)/2;
    xhat=xb;
    %calculate point in 3D space
    tempx=round(xhat);              %nearest cell in x direction
    tempy1=round((ss(1)-2*WBbox)/2-yhat);   %upper point
    tempz1=round(dpixels/2-zhat);           %front point
    tempy2=round((ss(1)-2*WBbox)/2+yhat);   %lower point
    tempz2=round(dpixels/2+zhat);           %back point
    %translate into cell coordinates
    if tempx<1
        tempx=1;
    end
    if tempx>(ss(2)-2*WBbox)
        tempx=ss(2)-2*WBbox;
    end
    if tempy1<2
        tempy1=2;
    end
    if tempy2>(ss(1)-2*WBbox-1)
        tempy2=ss(1)-2*WBbox-1;
    end
    if tempz2>(dpixels-1)
        tempz2=dpixels-1;
    end
    if tempz1<2
        tempz1=2;
    end
    k(tempy1:tempy2,tempx,tempz1:tempz2)=true;
end
k4=k;
```

```
clear k2 k3 k
%k4 is 3D logical matrix describing bubble areas
%1 indicates part of bubble, 0 is background
%downsample to make calculations faster
%transform cube of 8 voxels into one new voxel
N=size(k4);
ax1=mod(N(1),2);
ax2=mod(N(2),2);
ax3=mod(N(3),2);
k5=logical(false(floor(N(1)/2)+ax1,floor(N(2)/2)+ax2,floor(N(3)/2)+ax3)
);
for i=1:2:(N(1)-ax1)
    for j=1:2:(N(2)-ax2)
        for k=1:2:(N(3)-ax3)
            k5(round(i/2),round(j/2),round(k/2))=round(mean([k4(i,j,k)
k4(i+1,j,k) k4(i,j+1,k) k4(i+1,j+1,k) k4(i,j,k+1) k4(i+1,j,k+1)
k4(i+1,j+1,k+1) k4(i,j+1,k+1)]));
        end
    end
end
if ax1
    i=N(1);
    for j=1:2:(N(2)-ax2)
        for k=1:2:(N(3)-ax3)
            k5(floor(N(1)/2)+ax1,round(j/2),round(k/2))=round(mean([k4(
i,j,k) k4(i,j+1,k) k4(i,j,k+1) k4(i,j+1,k+1)]));
        end
    end
end
if ax2
    j=N(2);
    for i=1:2:(N(1)-ax1)
        for k=1:2:(N(3)-ax3)
            k5(round(i/2),floor(N(2)/2)+ax2,round(k/2))=round(mean([k4(
i,j,k) k4(i+1,j,k) k4(i,j,k+1) k4(i+1,j,k+1)]));
        end
    end
end
if ax3
    k=N(3);
    for i=1:2:(N(1)-ax1)
        for j=1:2:(N(2)-ax2)
            k5(round(i/2),round(j/2),floor(N(3)/2)+ax3)=round(mean([k4(
i,j,k) k4(i+1,j,k) k4(i,j+1,k) k4(i+1,j+1,k)]));
        end
    end
end
if ax1 && ax2
    i=N(1);
    j=N(2);
    for k=1:2:(N(3)-ax3)
        k5(floor(N(1)/2)+ax1,floor(N(2)/2)+ax2,round(k/2))=round(mean([
k4(i,j,k) k4(i,j,k+1)]));
    end
end
```

```
if ax1 && ax3
    i=N(1);
    k=N(3);
    for j=1:2:(N(2)-ax2)
        k5(floor(N(1)/2)+ax1,round(j/2),floor(N(3)/2)+ax3)=round(mean([
k4(i,j,k) k4(i,j+1,k)]));
    end
end
if ax2 && ax3
    j=N(2);
    k=N(3);
    for i=1:2:(N(1)-ax1)
        k5(round(i/2),floor(N(2)/2)+ax2,floor(N(3)/2)+ax3)=round(mean([
k4(i,j,k) k4(i+1,j,k)]));
    end
end
if ax1 && ax2 && ax3
     k5(floor(N(1)/2)+ax1,floor(N(2)/2)+ax2,floor(N(3)/2)+ax3)=k4(N(1),N
(2),N(3));
end
```

Function 27. lap3dsolver.m:

```
%solve 3D laplace equation for two phase flow in a microchannel
%run after gambitjou.m
%________===Ve1===________
%|                       |
%|   bb             bb   |
%|   bb             bb   |
%|_______         _______|
%         ===Ve2===
%== are electrodes
%bb are bubbles
%3D logical matrix k5 contains bubble cell locations
%a is coefficient of ith jth kth term, Vp, current point
%b is coefficient of i+1 jth kth term, Vr, right point
%c is coefficient of i-1 jth kth term, Vl, left point
%d is coefficient of ith j+1 kth term, Vm, bottom point
%e is coefficient of ith j-1 kth term, Vt, top point
%f is coefficient of ith jth k+1 term, Vb, back point
%g is coefficient of ith jth k-1 term, Vf, front point
%h is source term
%V is voltage for every cell
%all voltage outputs have units
len=(ss(2)-2*WBbox)/wratio/1000;  %calculate channel length [mm]
dx=2*len/(ss(2)-2*WBbox);   %delta x in left and right regions [mm]
dx2=dx;
dy=dx;      %delta y in top and bottom regions [mm]
dy2=dy;
dz=dx;      %delta z in front and back regions [mm]
dz2=dz;
if ax1
    dx2=dx/2;
end
if ax2
    dy2=dy/2;
end
if ax3
    dz2=dz/2;
end
k1=.01;     %electrical conductivity of water region [1/ohm m]
k2=2.5e-14; %electrical conductivity of bubble region [1/ohm m]
Ve1=5;      %voltage of electrode 1
Ve2=1;      %voltage of electrode 2
N=size(k5); %dimensions of mesh
            %N(1) is number of rows (y)
            %N(2) is number of columns (x)
            %N(3) is number of planes (z)
tic;
[a,b,c,d,e,f,g,h]=initmat(dx,dx2,dy,dy2,dz,dz2,k1,k2,Ve1,Ve2,N,k5,ax1,a
x2,ax3);
disp('matrices initialized');
toc;
tic;
counter=0;  %set counter to zeros
err=0.9;    %error from one iteration to the next
```

```
V=abs((Ve2-Ve1)/2+Ve1)*ones(N(1),N(2),N(3));    %initial guess
while err>0.00001
    one=V;
    for k=1:N(3)    %planes first
        a1=reshape(a(:,:,k),N(1),N(2));
        b1=reshape(b(:,:,k),N(1),N(2));
        c1=reshape(c(:,:,k),N(1),N(2));
        d1=reshape(d(:,:,k),N(1),N(2));
        e1=reshape(e(:,:,k),N(1),N(2));
        f1=reshape(f(:,:,k),N(1),N(2));
        g1=reshape(g(:,:,k),N(1),N(2));
        if k==1
       h1=reshape(h(:,:,k),N(1),N(2))+f1.*reshape(V(:,:,k+1),N(1),N(2));
        elseif k==N(3)
            h1=reshape(h(:,:,k),N(1),N(2))+g1.*reshape(V(:,:,k-
1),N(1),N(2));
        else
            h1=reshape(h(:,:,k),N(1),N(2))+f1.*reshape(V(:,:,k+1),N(1),
N(2))+g1.*reshape(V(:,:,k-1),N(1),N(2));
        end
        V1=reshape(V(:,:,k),N(1),N(2));
        V(:,:,k)=tdmaline(a1,b1,c1,d1,e1,h1,V1);
    end
    for j=1:N(2)    %columns second
        a1=reshape(a(:,j,:),N(1),N(3));
        b1=reshape(b(:,j,:),N(1),N(3));
        c1=reshape(c(:,j,:),N(1),N(3));
        d1=reshape(d(:,j,:),N(1),N(3));
        e1=reshape(e(:,j,:),N(1),N(3));
        f1=reshape(f(:,j,:),N(1),N(3));
        g1=reshape(g(:,j,:),N(1),N(3));
        if j==1
       h1=reshape(h(:,j,:),N(1),N(3))+b1.*reshape(V(:,j+1,:),N(1),N(3));
        elseif j==N(2)
            h1=reshape(h(:,j,:),N(1),N(3))+c1.*reshape(V(:,j-
1,:),N(1),N(3));
        else
            h1=reshape(h(:,j,:),N(1),N(3))+b1.*reshape(V(:,j+1,:),N(1),
N(3))+c1.*reshape(V(:,j-1,:),N(1),N(3));
        end
        V1=reshape(V(:,j,:),N(1),N(3));
        V(:,j,:)=tdmaline(a1,f1,g1,d1,e1,h1,V1);
    end
    for i=1:N(1)    %rows third
        a1=reshape(a(i,:,:),N(2),N(3));
        b1=reshape(b(i,:,:),N(2),N(3));
        c1=reshape(c(i,:,:),N(2),N(3));
        d1=reshape(d(i,:,:),N(2),N(3));
        e1=reshape(e(i,:,:),N(2),N(3));
        f1=reshape(f(i,:,:),N(2),N(3));
        g1=reshape(g(i,:,:),N(2),N(3));
        if i==1
       h1=reshape(h(i,:,:),N(2),N(3))+d1.*reshape(V(i+1,:,:),N(2),N(3));
        elseif i==N(1)
```

```
            h1=reshape(h(i,:,:),N(2),N(3))+e1.*reshape(V(i-
1,:,:),N(2),N(3));
        else
            h1=reshape(h(i,:,:),N(2),N(3))+d1.*reshape(V(i+1,:,:),N(2),
N(3))+e1.*reshape(V(i-1,:,:),N(2),N(3));
        end
        V1=reshape(V(i,:,:),N(2),N(3));
        V(i,:,:)=tdmaline(a1,f1,g1,b1,c1,h1,V1);
    end
    for k=N(3):-1:1 %backwards planes fourth
        a1=reshape(a(:,:,k),N(1),N(2));
        b1=reshape(b(:,:,k),N(1),N(2));
        c1=reshape(c(:,:,k),N(1),N(2));
        d1=reshape(d(:,:,k),N(1),N(2));
        e1=reshape(e(:,:,k),N(1),N(2));
        f1=reshape(f(:,:,k),N(1),N(2));
        g1=reshape(g(:,:,k),N(1),N(2));
        if k==1
       h1=reshape(h(:,:,k),N(1),N(2))+f1.*reshape(V(:,:,k+1),N(1),N(2));
        elseif k==N(3)
            h1=reshape(h(:,:,k),N(1),N(2))+g1.*reshape(V(:,:,k-
1),N(1),N(2));
        else
            h1=reshape(h(:,:,k),N(1),N(2))+f1.*reshape(V(:,:,k+1),N(1),
N(2))+g1.*reshape(V(:,:,k-1),N(1),N(2));
        end
        V1=reshape(V(:,:,k),N(1),N(2));
        V(:,:,k)=tdmaline(a1,b1,c1,d1,e1,h1,V1);
    end
    for j=N(2):-1:1 %backwards columns fifth
        a1=reshape(a(:,j,:),N(1),N(3));
        b1=reshape(b(:,j,:),N(1),N(3));
        c1=reshape(c(:,j,:),N(1),N(3));
        d1=reshape(d(:,j,:),N(1),N(3));
        e1=reshape(e(:,j,:),N(1),N(3));
        f1=reshape(f(:,j,:),N(1),N(3));
        g1=reshape(g(:,j,:),N(1),N(3));
        if j==1
       h1=reshape(h(:,j,:),N(1),N(3))+b1.*reshape(V(:,j+1,:),N(1),N(3));
        elseif j==N(2)
            h1=reshape(h(:,j,:),N(1),N(3))+c1.*reshape(V(:,j-
1,:),N(1),N(3));
        else
            h1=reshape(h(:,j,:),N(1),N(3))+b1.*reshape(V(:,j+1,:),N(1),
N(3))+c1.*reshape(V(:,j-1,:),N(1),N(3));
        end
        V1=reshape(V(:,j,:),N(1),N(3));
        V(:,j,:)=tdmaline(a1,f1,g1,d1,e1,h1,V1);
    end
    for i=N(1):-1:1 %backwards rows sixth
        a1=reshape(a(i,:,:),N(2),N(3));
        b1=reshape(b(i,:,:),N(2),N(3));
        c1=reshape(c(i,:,:),N(2),N(3));
        d1=reshape(d(i,:,:),N(2),N(3));
        e1=reshape(e(i,:,:),N(2),N(3));
        f1=reshape(f(i,:,:),N(2),N(3));
        g1=reshape(g(i,:,:),N(2),N(3));
        if i==1
       h1=reshape(h(i,:,:),N(2),N(3))+d1.*reshape(V(i+1,:,:),N(2),N(3));
        elseif i==N(1)
            h1=reshape(h(i,:,:),N(2),N(3))+e1.*reshape(V(i-
1,:,:),N(2),N(3));
        else
            h1=reshape(h(i,:,:),N(2),N(3))+d1.*reshape(V(i+1,:,:),N(2),
N(3))+e1.*reshape(V(i-1,:,:),N(2),N(3));
        end
        V1=reshape(V(i,:,:),N(2),N(3));
        V(i,:,:)=tdmaline(a1,f1,g1,b1,c1,h1,V1);
    end
    two=V;
    err=max(max(max(abs((one-two)./one))));
    counter=counter+1;
    if ~mod(counter,10)
        disp(sprintf('still going... iteration: %i ave V: %2.3f
err: %1.5f',counter,mean(mean(mean(V))),err));
    end
    if counter>=1000
        disp(sprintf('over 1000 outer iterations, error is %1.5f',err));
        err=0;
    end
    if err>5
        disp(sprintf('diverging, error is %1.5f',err));
        err=0;
    end
    one=mean(mean(mean(V)));
    if one>2*Ve1
        disp(sprintf('V values too high, error is %1.5f',err));
        err=0;
    end
end
toc;
clear one two a b c d e f g h a1 b1 c1 d1 e1 f1 g1 h1
xx=ones(N(1),N(2));
for i=1:N(1)
    xx(i,:)=1:N(2);
end
yy=ones(N(1),N(2));
for j=1:N(2)
    yy(:,j)=1:N(1);
end
figure
contour(xx,yy,V(:,:,floor(N(3)/2)));    %plot contour of voltages
title('Voltage contour map','fontsize',16);
figure
surf(xx,yy,V(:,:,floor(N(3)/2)));       %plot surface of voltages
title('Voltage profile','fontsize',16);
%voltage to resistance calculation
elec1=round(N(2)/3);    %cell number for left side of electrode
elec2=round(2*N(2)/3);  %cell number for right side of electrode
I=zeros((elec2-elec1+1),N(3));
```

```
for j=elec1:elec2
    for k=1:(N(3)-1)
        i=1;
        I((j-elec1+1),k)=(Ve1-V(i,j,k))/dy*dx*dz*(k5(i,j,k)*k2+(1-
k5(i,j,k))*k1)*2;
    end
    I((j-elec1+1),N(3))=(Ve1-
V(1,elec2,N(3)))/dy*dx*dz2*(k5(1,elec2,N(3))*k2+(1-
k5(1,elec2,N(3)))*k1)*2;
end
current=sum(sum(I))/1000;
resistance(frame)=(Ve1-Ve2)/current;
save(OutFileName)
```

Function 28. lap3dsolver2.m:

```
%solve 3D laplace equation for two phase flow in a microchannel
%run after gambitjou.m
%________===Ve1===________
%|                       |
%|   bb             bb   |
%|   bb             bb   |
%|_______         _______|
%        ===Ve2===
%== are electrodes
%bb are bubbles
%3D logical matrix k4 contains bubble cell locations
%a is coefficient of ith jth kth term, Vp, current point
%b is coefficient of i+1 jth kth term, Vr, right point
%c is coefficient of i-1 jth kth term, Vl, left point
%d is coefficient of ith j+1 kth term, Vm, bottom point
%e is coefficient of ith j-1 kth term, Vt, top point
%f is coefficient of ith jth k+1 term, Vb, back point
%g is coefficient of ith jth k-1 term, Vf, front point
%h is source term
%V is voltage for every cell
%all voltage outputs have units
len=(ss(2)-2*WBbox)/wratio/1000;  %calculate channel length [mm]
dx=2*len/(ss(2)-2*WBbox);   %delta x in left and right regions [mm]
dx2=dx;
dy=dx;      %delta y in top and bottom regions [mm]
dy2=dy;
dz=dx;      %delta z in front and back regions [mm]
dz2=dz;
if ax1
    dx2=dx/2;
end
if ax2
    dy2=dy/2;
end
if ax3
    dz2=dz/2;
end
k1=80;      %dielectric constant of water region [-]
k2=1;       %dielectric constant of bubble region [-]
Ve1=5;      %voltage of electrode 1
Ve2=1;      %voltage of electrode 2
N=size(k5); %dimensions of mesh
            %N(1) is number of rows (y)
            %N(2) is number of columns (x)
            %N(3) is number of planes (z)
tic;
[a,b,c,d,e,f,g,h]=initmat(dx,dx2,dy,dy2,dz,dz2,k1,k2,Ve1,Ve2,N,k5,ax1,a
x2,ax3);
disp('matrices initialized');
toc;
tic;
counter=0;  %set counter to zeros
err=0.9;    %error from one iteration to the next
```

```
V=abs((Ve2-Ve1)/2+Ve1)*ones(N(1),N(2),N(3));    %initial guess
while err>0.00001
    one=V;
    for k=1:N(3)    %planes first
        a1=reshape(a(:,:,k),N(1),N(2));
        b1=reshape(b(:,:,k),N(1),N(2));
        c1=reshape(c(:,:,k),N(1),N(2));
        d1=reshape(d(:,:,k),N(1),N(2));
        e1=reshape(e(:,:,k),N(1),N(2));
        f1=reshape(f(:,:,k),N(1),N(2));
        g1=reshape(g(:,:,k),N(1),N(2));
        if k==1
       h1=reshape(h(:,:,k),N(1),N(2))+f1.*reshape(V(:,:,k+1),N(1),N(2));
        elseif k==N(3)
            h1=reshape(h(:,:,k),N(1),N(2))+g1.*reshape(V(:,:,k-
1),N(1),N(2));
        else
            h1=reshape(h(:,:,k),N(1),N(2))+f1.*reshape(V(:,:,k+1),N(1),
N(2))+g1.*reshape(V(:,:,k-1),N(1),N(2));
        end
        V1=reshape(V(:,:,k),N(1),N(2));
        V(:,:,k)=tdmaline(a1,b1,c1,d1,e1,h1,V1);
    end
    for j=1:N(2)    %columns second
        a1=reshape(a(:,j,:),N(1),N(3));
        b1=reshape(b(:,j,:),N(1),N(3));
        c1=reshape(c(:,j,:),N(1),N(3));
        d1=reshape(d(:,j,:),N(1),N(3));
        e1=reshape(e(:,j,:),N(1),N(3));
        f1=reshape(f(:,j,:),N(1),N(3));
        g1=reshape(g(:,j,:),N(1),N(3));
        if j==1
       h1=reshape(h(:,j,:),N(1),N(3))+b1.*reshape(V(:,j+1,:),N(1),N(3));
        elseif j==N(2)
            h1=reshape(h(:,j,:),N(1),N(3))+c1.*reshape(V(:,j-
1,:),N(1),N(3));
        else
            h1=reshape(h(:,j,:),N(1),N(3))+b1.*reshape(V(:,j+1,:),N(1),
N(3))+c1.*reshape(V(:,j-1,:),N(1),N(3));
        end
        V1=reshape(V(:,j,:),N(1),N(3));
        V(:,j,:)=tdmaline(a1,f1,g1,d1,e1,h1,V1);
    end
    for i=1:N(1)    %rows third
        a1=reshape(a(i,:,:),N(2),N(3));
        b1=reshape(b(i,:,:),N(2),N(3));
        c1=reshape(c(i,:,:),N(2),N(3));
        d1=reshape(d(i,:,:),N(2),N(3));
        e1=reshape(e(i,:,:),N(2),N(3));
        f1=reshape(f(i,:,:),N(2),N(3));
        g1=reshape(g(i,:,:),N(2),N(3));
        if i==1
       h1=reshape(h(i,:,:),N(2),N(3))+d1.*reshape(V(i+1,:,:),N(2),N(3));
        elseif i==N(1)
```

```
            h1=reshape(h(i,:,:),N(2),N(3))+e1.*reshape(V(i-
1,:,:),N(2),N(3));
        else
            h1=reshape(h(i,:,:),N(2),N(3))+d1.*reshape(V(i+1,:,:),N(2),
N(3))+e1.*reshape(V(i-1,:,:),N(2),N(3));
        end
        V1=reshape(V(i,:,:),N(2),N(3));
        V(i,:,:)=tdmaline(a1,f1,g1,b1,c1,h1,V1);
    end
    for k=N(3):-1:1 %backwards planes fourth
        a1=reshape(a(:,:,k),N(1),N(2));
        b1=reshape(b(:,:,k),N(1),N(2));
        c1=reshape(c(:,:,k),N(1),N(2));
        d1=reshape(d(:,:,k),N(1),N(2));
        e1=reshape(e(:,:,k),N(1),N(2));
        f1=reshape(f(:,:,k),N(1),N(2));
        g1=reshape(g(:,:,k),N(1),N(2));
        if k==1
       h1=reshape(h(:,:,k),N(1),N(2))+f1.*reshape(V(:,:,k+1),N(1),N(2));
        elseif k==N(3)
            h1=reshape(h(:,:,k),N(1),N(2))+g1.*reshape(V(:,:,k-
1),N(1),N(2));
        else
            h1=reshape(h(:,:,k),N(1),N(2))+f1.*reshape(V(:,:,k+1),N(1),
N(2))+g1.*reshape(V(:,:,k-1),N(1),N(2));
        end
        V1=reshape(V(:,:,k),N(1),N(2));
        V(:,:,k)=tdmaline(a1,b1,c1,d1,e1,h1,V1);
    end
    for j=N(2):-1:1 %backwards columns fifth
        a1=reshape(a(:,j,:),N(1),N(3));
        b1=reshape(b(:,j,:),N(1),N(3));
        c1=reshape(c(:,j,:),N(1),N(3));
        d1=reshape(d(:,j,:),N(1),N(3));
        e1=reshape(e(:,j,:),N(1),N(3));
        f1=reshape(f(:,j,:),N(1),N(3));
        g1=reshape(g(:,j,:),N(1),N(3));
        if j==1
       h1=reshape(h(:,j,:),N(1),N(3))+b1.*reshape(V(:,j+1,:),N(1),N(3));
        elseif j==N(2)
            h1=reshape(h(:,j,:),N(1),N(3))+c1.*reshape(V(:,j-
1,:),N(1),N(3));
        else
            h1=reshape(h(:,j,:),N(1),N(3))+b1.*reshape(V(:,j+1,:),N(1),
N(3))+c1.*reshape(V(:,j-1,:),N(1),N(3));
        end
        V1=reshape(V(:,j,:),N(1),N(3));
        V(:,j,:)=tdmaline(a1,f1,g1,d1,e1,h1,V1);
    end
    for i=N(1):-1:1 %backwards rows sixth
        a1=reshape(a(i,:,:),N(2),N(3));
        b1=reshape(b(i,:,:),N(2),N(3));
        c1=reshape(c(i,:,:),N(2),N(3));
        d1=reshape(d(i,:,:),N(2),N(3));
        e1=reshape(e(i,:,:),N(2),N(3));
        f1=reshape(f(i,:,:),N(2),N(3));
        g1=reshape(g(i,:,:),N(2),N(3));
        if i==1
       h1=reshape(h(i,:,:),N(2),N(3))+d1.*reshape(V(i+1,:,:),N(2),N(3));
        elseif i==N(1)
            h1=reshape(h(i,:,:),N(2),N(3))+e1.*reshape(V(i-
1,:,:),N(2),N(3));
        else
            h1=reshape(h(i,:,:),N(2),N(3))+d1.*reshape(V(i+1,:,:),N(2),
N(3))+e1.*reshape(V(i-1,:,:),N(2),N(3));
        end
        V1=reshape(V(i,:,:),N(2),N(3));
        V(i,:,:)=tdmaline(a1,f1,g1,b1,c1,h1,V1);
    end
    two=V;
    err=max(max(max(abs((one-two)./one))));
    counter=counter+1;
    if ~mod(counter,10)
        disp(sprintf('still going... iteration: %i ave V: %2.3f
err: %1.5f',counter,mean(mean(mean(V))),err));
    end
    if counter>=1000
        disp(sprintf('over 1000 outer iterations, error is %1.5f',err));
        err=0;
    end
    if err>5
        disp(sprintf('diverging, error is %1.5f',err));
        err=0;
    end
    one=mean(mean(mean(V)));
    if one>2*Ve1
        disp(sprintf('V values too high, error is %1.5f',err));
        err=0;
    end
end
toc;
clear one two a b c d e f g h a1 b1 c1 d1 e1 f1 g1 h1
xx=ones(N(1),N(2));
for i=1:N(1)
    xx(i,:)=1:N(2);
end
yy=ones(N(1),N(2));
for j=1:N(2)
    yy(:,j)=1:N(1);
end
figure
contour(xx,yy,V(:,:,floor(N(3)/2)));    %plot contour of voltages
title('Voltage contour map','fontsize',16);
figure
surf(xx,yy,V(:,:,floor(N(3)/2)));       %plot surface of voltages
title('Voltage profile','fontsize',16);
%voltage to capacitance calculation
elec1=round(N(2)/3);    %cell number for left side of electrode
elec2=round(2*N(2)/3);  %cell number for right side of electrode
Q=zeros((elec2-elec1+1),N(3));
```

```
for j=elec1:elec2
    for k=1:(N(3)-1)
        i=1;
        Q((j-elec1+1),k)=(Ve1-V(i,j,k))/dy*dx*dz*(k5(i,j,k)*k2+(1-
k5(i,j,k))*k1)*2;
    end
    Q((j-elec1+1),N(3))=(Ve1-
V(1,elec2,N(3)))/dy*dx*dz2*(k5(1,elec2,N(3))*k2+(1-
k5(1,elec2,N(3)))*k1)*2;
end
charge=sum(sum(Q))*8.854e-12/1000;
capacitance(frame)=charge/(Ve1-Ve2);
save(OutFileName)
```

Function 29. tdma.m:

```
%TDMA solver
%equation at ith grid point:
% a_i * phi_i = b_i * phi_i+1 + c_i * phi_i-1 + d_i
%tridiagonal matrix formed from all grid points:
% [ a -b 0  0  ]       [ d ]
% [-c a  -b 0  ] phi = [ d ]
% [ 0 -c a  -b ]       [ d ]
% [ 0 0  -c a  ]       [ d ]
%a is coefficient of ith term
%b is coefficient of i+1 term
%c is coefficient of i-1 term
%d is source term
%function outputs phi
function [phi] = tdma(a,b,c,d)
N=length(a);    %number of grid points
P=zeros(1,N);
Q=zeros(1,N);
P(1)=b(1)/a(1);
Q(1)=d(1)/a(1);
for i=2:N
    P(i)=b(i)/(a(i)-c(i)*P(i-1));
    Q(i)=(d(i)+c(i)*Q(i-1))/(a(i)-c(i)*P(i-1));
end
phi(N)=Q(N);
for i=N-1:-1:1
    phi(i)=P(i)*phi(i+1)+Q(i);
end
end
```

Function 30. tdmaline.m:

```
%line by line TDMA solver
%equation at ith grid point:
% a_i,j * phi_i,j = b_i,j * phi_i+1,j + c_i,j * phi_i-1,j + d_i,j *
% phi_i,j+1 + e_i,j * phi_i,j-1 + f
%tridiagonal matrix formed from all grid points:
% [ a -b 0  0  ]       [ d ]
% [-c a  -b 0  ] phi = [ d ]
% [ 0 -c a  -b ]       [ d ]
% [ 0 0  -c a  ]       [ d ]
%a is coefficient of ith jth term
%b is coefficient of i+1 jth term
%c is coefficient of i-1 jth term
%d is coefficient of ith j+1 term
%e is coefficient of ith j-1 term
%f is source term
%g is initial guesses for phi
%function outputs phi
function [phi] = tdmaline(a,b,c,d,e,f,g)
N=size(a);      %number of grid points
                %N(1) is number of rows
                %N(2) is number of columns
    for i=1:N(1)        %rows first
        a1=a(i,:);
        b1=b(i,:);
        c1=c(i,:);
        if i==1
            d1=f(i,:)+d(i,:).*g(i+1,:);
        elseif i==N(1)
            d1=f(i,:)+e(i,:).*g(i-1,:);
        else
            d1=f(i,:)+d(i,:).*g(i+1,:)+e(i,:).*g(i-1,:);
        end
        g(i,:)=tdma(a1,b1,c1,d1);   %pass to tdma solver
    end
    for j=1:N(2)        %columns second
        a1=a(:,j);
        b1=d(:,j);
        c1=e(:,j);
        if j==1
            d1=f(:,j)+b(:,j).*g(:,j+1);
        elseif j==N(2)
            d1=f(:,j)+c(:,j).*g(:,j-1);
        else
            d1=f(:,j)+b(:,j).*g(:,j+1)+c(:,j).*g(:,j-1);
        end
        g(:,j)=tdma(a1,b1,c1,d1);   %pass to tdma solver
    end
    for i=N(1):-1:1     %backward rows third
        a1=a(i,:);
        b1=b(i,:);
        c1=c(i,:);
        if i==1
            d1=f(i,:)+d(i,:).*g(i+1,:);
        elseif i==N(1)
            d1=f(i,:)+e(i,:).*g(i-1,:);
        else
            d1=f(i,:)+d(i,:).*g(i+1,:)+e(i,:).*g(i-1,:);
        end
        g(i,:)=tdma(a1,b1,c1,d1);   %pass to tdma solver
    end
    for j=N(2):-1:1     %backward columns fourth
        a1=a(:,j);
        b1=d(:,j);
        c1=e(:,j);
        if j==1
            d1=f(:,j)+b(:,j).*g(:,j+1);
        elseif j==N(2)
            d1=f(:,j)+c(:,j).*g(:,j-1);
        else
            d1=f(:,j)+b(:,j).*g(:,j+1)+c(:,j).*g(:,j-1);
        end
        g(:,j)=tdma(a1,b1,c1,d1);   %pass to tdma solver
    end
phi=g;
end
```

Appendix C MATLAB Script for Non-Uniform Data Analysis

A MATLAB script was developed to analyze the data obtained in the non-uniform heating experiments as discussed in Section 6.2. This script is a modified version of the one used by Harirchian [89] to accommodate non-uniform heating profiles. The script first reads in the data obtained during experiments. It then applies a set of correlations to find the diode temperature, heat generation, and heat loss at every node. It then calculates the pressure drop across the test section and the local saturation temperature. Next it calculates the net heat transfer between each node and the local fluid temperatures. It then calculates the local heat transfer coefficients and records all of the data in a spreadsheet.

CalculatorUpdatedConduction2.m:

```
clc
clear
Boiling = xlsread('CalibrationData', 'Boiling', 'B3:B27');
Dimensions = xlsread('CalibrationData', 'Dimensions', 'B1:B7');
Heaters = xlsread('TestData','Heaters','B2:B26');    %which heaters are
turned on
wf = Dimensions(1); %fin width m
w = Dimensions(2); %channel width m
d = Dimensions(3); %channel depth m
N = Dimensions(4); %number of channels
Width = Dimensions(5); %chip width m
Length = Dimensions(6); %chip length m
t = Dimensions(7); %total chip thickness m
tbase = t - d; %chip base thickness m
Aw = N*(w+2*d)*Length; %wetted area m^2
Af = 2*Length*d; %fin area m^2
SiliconK = 140;
hfg = 89000; %Heat of vaporization
cpf = 1100; %Specific Heat, fluid J/kg*K
Position =
[.9 .9 .9 .9 .9 .7 .7 .7 .7 .7 .5 .5 .5 .5 .5 .3 .3 .3 .3 .3 .1 .1 .1 .
1 .1];
sigma = .0062; %Fluid surface tension N/m .0062
rhof = 1600; %Fluid Density kg/m3  1600
```

```
rhog = 15.84; %Vapor Density kg/m3  15.84
uf = .00052; %Fluid viscosity .00052
ug = .00002; %Gas Viscosity .00002
A1 = 0.01465*0.0228; %Inlet manifold area m^2
A2 = 0.00165*0.01267; %Plenum Area m^2
A3 = w*d*N; %Total Channel Area m^2
A4 = 0.01465*0.00835; %Exit Manifold Area m^2
%Calibration
%Diode Calibration
DiodeV = xlsread('CalibrationData', 'DiodeCalibration', 'A3:Y8');
SizeD = size(DiodeV);
DiodeT = xlsread('CalibrationData', 'DiodeCalibration', 'A11:A16');
DiodeVAvg=zeros(1,SizeD(1));
for i1 = 1:SizeD(1)
    DiodeVAvg(i1) = sum(DiodeV(i1,:))/SizeD(2);
end
DiodeModel = polyfit(DiodeVAvg', DiodeT, 1);
%Resistor Calibration
Resistance  = xlsread('CalibrationData', 'ResistorCalibration', 'A3:Y7');
SizeR = size(Resistance);
ResistanceT = xlsread('CalibrationData', 'ResistorCalibration',
'A11:A15');
ResistorModel=zeros(3,SizeR(2));
for i1 = 1:SizeR(2)
    [ResistorModel(:,i1)]=polyfit(ResistanceT,Resistance(:,i1),2);
end
%Heat Loss
Voltage = xlsread('CalibrationData', 'HeatLossCalibration', 'A10:A14');
Current = xlsread('CalibrationData', 'HeatLossCalibration', 'B10:B14');
DiodeLossVoltage = xlsread('CalibrationData', 'HeatLossCalibration',
'A3:Y7');
LossTemperature = DiodeModel(1).*DiodeLossVoltage +
ones(size(DiodeLossVoltage))*DiodeModel(2);
SizeT = size(LossTemperature);
SizeV = size(Voltage);
TotalLoss = Voltage.*Current;
AvgLossTemp=zeros(1,SizeT(1));
for i1 = 1:SizeT(1)
    AvgLossTemp(i1) = sum(LossTemperature(i1,:))/SizeT(2);
end
LossResistance=repmat(ResistorModel(1,:),5,1).*LossTemperature.^2+repma
t(ResistorModel(2,:),5,1).*LossTemperature+repmat(ResistorModel(3,:),5,
1);
LocalHeatLoss=repmat(Voltage,1,25).^2./LossResistance;
LocalLossModel=zeros(2,SizeT(2));
for i1 = 1:SizeT(2)
    ModelTemperatureTemp=LossTemperature(:,i1);
    ModelLossTemp=LocalHeatLoss(:,i1);
   [LocalLossModel(:,i1)]=polyfit(ModelTemperatureTemp,ModelLossTemp,1);
end
TotalLossModel = polyfit(AvgLossTemp', TotalLoss, 1);
%Data Processing
TestData = xlsread('TestData', 'ExperimentalData'); %Read in
experimental data
SampleSize = size(TestData);
```

```
%Diode Temperature at each node
TestTemp=DiodeModel(1)*TestData(:,1:25)+DiodeModel(2);
%Heater resistance in each node
TestResistance=repmat(ResistorModel(1,:),SampleSize(1),1).*TestTemp.^2+
repmat(ResistorModel(2,:),SampleSize(1),1).*TestTemp+repmat(ResistorMod
el(3,:),SampleSize(1),1);
%Heat generation at each node
LocalHeatGeneration=repmat(TestData(:,26),1,25).^2./TestResistance.*(re
pmat(Heaters',SampleSize(1),1)==1) +
repmat(TestData(:,32),1,25).^2./TestResistance.*(repmat(Heaters',Sample
Size(1),1)==2);
%Heat loss at each node
TestLocalHeatLoss=repmat(LocalLossModel(1,:),SampleSize(1),1).*TestTemp
+repmat(LocalLossModel(2,:),SampleSize(1),1);
%Inlet pressure for each voltage level tested [Pa]
InletPressure=(14.697+TestData(:,27))/.00014504;
%Pressure drop for each voltage [Pa]
DP=(TestData(:,28))/.00014504;
%Exit fluid temperature [C]
ExitTemp=TestData(:,30);
%Inlet fluid temp [C]
Tin=TestData(:,29);
%Mass flow rate [mL/min]
Mdot=TestData(:,31);
%Mass flux (kg/m^2s)
G = Mdot/60*(.01)^3*rhof/A3;
%%%%Pressure drop calculations%%%%%
%Inlet manifold to plenum [Pa]
alpha2 = 0.00165/0.01267;
G2=G*A3/A2;
Kc2 = .0088*alpha2^2-.1785*alpha2+1.6027;
DP12 = (1-(A2/A1)^2+Kc2)*0.5*G2.^2/rhof;
%Plenum to Microchannel [Pa]
alpha3 = max(w/d, d/w);
Kc3 = .0088*alpha3^2-.1785*alpha3+1.6027;
DP23 = (1-(A3/A2)^2+Kc3)*0.5*G.^2/rhof;
qdotnet = sum((LocalHeatGeneration-TestLocalHeatLoss), 2);
%Microchannel to exit manifold
TsatExit=1928./(10.216-log10(InletPressure-DP))-273.15;
xe=1/hfg*(qdotnet./(G*d*w*N)-cpf*(TsatExit-Tin)); %Exit vapor quality
(overall)
for i1 = 1:(SampleSize(1))
    if(xe(i1)>1)
        xe(i1) = .999999999;
    end
    if(xe(i1)<0)
        xe(i1) = .0001;
    end
end
Xvv=(uf/ug)^.5*((1-xe)./xe).^.5*(rhog/rhof)^.5;
DP34=G.^2/rhof*(A3/A4*(A3/A4-1)).*(1-xe).^2.*(1+5./Xvv+1./Xvv.^2);
%Pressure drop across channels alone
DP3=DP-DP12-DP23-DP34;
%Local pressure at each node
```

```
LocalPressure=repmat(InletPressure,1,25)-repmat(DP12,1,25)-
repmat(DP23,1,25)-DP3*Position;
%Local saturation temperature
Tsat=1928./(10.216-log10(LocalPressure))-273.15;
TsatAve = mean(Tsat,2);
%Establish a local fluid temperature matrix with temperature at each
node for each voltage tested, taking into account single and two-phase
flow regions
Ac1= Width*(t-d)/5;
Ac2= Length*(t-d)/5;
NetConductionOut=zeros(SampleSize(1),25);
C1=Ac1*SiliconK/(Length/5);
C2=Ac2*SiliconK/(Width/5);
NetConductionOut(:,1)=C2*(TestTemp(:,1)-
TestTemp(:,2))+C1*(TestTemp(:,1)-TestTemp(:,6));
NetConductionOut(:,2:4)=C2*(2*TestTemp(:,2:4)-TestTemp(:,3:5)-
TestTemp(:,1:3))+C1*(TestTemp(:,2:4)-TestTemp(:,7:9));
NetConductionOut(:,5)=C2*(TestTemp(:,5)-
TestTemp(:,4))+C1*(TestTemp(:,5)-TestTemp(:,10));
NetConductionOut(:,6)=C2*(TestTemp(:,6)-
TestTemp(:,7))+C1*(2*TestTemp(:,6)-TestTemp(:,1)-TestTemp(:,11));
NetConductionOut(:,7:9)=C2*(2*TestTemp(:,7:9)-TestTemp(:,8:10)-
TestTemp(:,6:8))+C1*(2*TestTemp(:,7:9)-TestTemp(:,2:4)-
TestTemp(:,12:14));
NetConductionOut(:,10)=C2*(TestTemp(:,10)-
TestTemp(:,9))+C1*(2*TestTemp(:,10)-TestTemp(:,5)-TestTemp(:,15));
NetConductionOut(:,11)=C2*(TestTemp(:,11)-
TestTemp(:,12))+C1*(2*TestTemp(:,11)-TestTemp(:,6)-TestTemp(:,16));
NetConductionOut(:,12:14)=C2*(2*TestTemp(:,12:14)-TestTemp(:,13:15)-
TestTemp(:,11:13))+C1*(2*TestTemp(:,12:14)-TestTemp(:,7:9)-
TestTemp(:,17:19));
NetConductionOut(:,15)=C2*(TestTemp(:,15)-
TestTemp(:,14))+C1*(2*TestTemp(:,15)-TestTemp(:,10)-TestTemp(:,20));
NetConductionOut(:,16)=C2*(TestTemp(:,16)-
TestTemp(:,17))+C1*(2*TestTemp(:,16)-TestTemp(:,11)-TestTemp(:,21));
NetConductionOut(:,17:19)=C2*(2*TestTemp(:,17:19)-TestTemp(:,18:20)-
TestTemp(:,16:18))+C1*(2*TestTemp(:,17:19)-TestTemp(:,12:14)-
TestTemp(:,22:24));
NetConductionOut(:,20)=C2*(TestTemp(:,20)-
TestTemp(:,19))+C1*(2*TestTemp(:,20)-TestTemp(:,15)-TestTemp(:,25));
NetConductionOut(:,21)=C2*(TestTemp(:,21)-
TestTemp(:,22))+C1*(TestTemp(:,21)-TestTemp(:,16));
NetConductionOut(:,22:24)=C2*(2*TestTemp(:,22:24)-TestTemp(:,23:25)-
TestTemp(:,21:23))+C1*(TestTemp(:,22:24)-TestTemp(:,17:19));
NetConductionOut(:,25)=C2*(TestTemp(:,25)-
TestTemp(:,24))+C1*(TestTemp(:,25)-TestTemp(:,20));
NetHeatTransfer=LocalHeatGeneration-TestLocalHeatLoss-NetConductionOut;
Tf = zeros(SampleSize(1), 25);
for i1 = 1:(SampleSize(1))
    for i2 = 25:-1:1
        if i2>=21
            Tf(i1,i2)=.5*NetHeatTransfer(i1,i2)/((G(i1)*w*d*N)/5*cpf)+T
in(i1);
        else
```

```
            Tf(i1,i2)=(.5*NetHeatTransfer(i1,i2)+.5*NetHeatTransfer(i1,
i2+5))/((G(i1)*w*d*N)/5*cpf)+Tf(i1,(i2+5));
        end
        if Tf(i1,i2)>Tsat(i1,i2)
            Tf(i1,i2) = Tsat(i1,i2);
        end
    end
end
%Q" on channel walls
NetHeatFlux = NetHeatTransfer./(Aw/25);
%Iterative h calculation
h = NetHeatFlux./(TestTemp - Tf);
htest = h;
test = 1;
while(test>.01)
    m = ((2*abs(h))/(SiliconK*wf)).^.5;
    nf = tanh(m*d)./(m*d);
    no = 1.-N*Af/Aw*(1-nf);
    h = NetHeatFlux./(no.*(TestTemp - Tf));
    test = max(max(abs(h-htest)));
    htest = h;
end
BaseHeatFlux = (sum(NetHeatTransfer,2))/(Width*Length);
Re = G*(A3/N)^.5/(uf); %Reynolds Number
Bo = 9.81*(rhof-rhog)*(A3/N)/sigma; %Bond Number
Bl=NetHeatFlux./(repmat(G,1,25)*hfg); %Boiling number
%Phase change number
Npch=NetHeatFlux*(w+2*d)/(A3/N*hfg)*(rhof-
rhog)/(rhof*rhog).*repmat(Position,SampleSize(1),1)*Length./(repmat(G,1
,25)/rhof);
xLocal=zeros(SampleSize(1),25);
xLocal(:,21:25)=1/hfg*(.5*NetHeatTransfer(:,21:25)./(repmat(G,1,5)*d*w*
N/5)-cpf*(Tf(:,21:25)-repmat(Tin,1,5)));
xLocal(:,16:20)=1/hfg*(.5*NetHeatTransfer(:,16:20)./(repmat(G,1,5)*d*w*
N/5)-cpf*(Tf(:,16:20)-Tf(:,21:25)));
xLocal(:,11:15)=1/hfg*(.5*NetHeatTransfer(:,11:15)./(repmat(G,1,5)*d*w*
N/5)-cpf*(Tf(:,11:15)-Tf(:,16:20)));
xLocal(:,6:10)=1/hfg*(.5*NetHeatTransfer(:,6:10)./(repmat(G,1,5)*d*w*N/
5)-cpf*(Tf(:,6:10)-Tf(:,11:15)));
xLocal(:,1:5)=1/hfg*(.5*NetHeatTransfer(:,1:5)./(repmat(G,1,5)*d*w*N/5)
-cpf*(Tf(:,1:5)-Tf(:,6:10)));
for i1 = 1:SampleSize(1)
    for i2 = 1:25
        if(xLocal(i1,i2)>1)
            xLocal(i1,i2) = 1;
        elseif(xLocal(i1,i2)<0)
            xLocal(i1,i2)=0;
        end
    end
end
xlswrite('Results', NetHeatFlux, 'LocalNetHeatFlux')
xlswrite('Results', NetHeatTransfer, 'LocalNetHeatTransfer')
xlswrite('Results', BaseHeatFlux, 'BaseHeatFlux')
xlswrite('Results', h, 'LocalHeatTransferCoeff')
xlswrite('Results',TestTemp,'Td')
xlswrite('Results',Tf,'Tf')
xlswrite('Results', G, 'G')
xlswrite('Results', nf, 'LocalFinEfficiency')
xlswrite('Results', no, 'LocalOverallEfficiency')
xlswrite('Results', {'xe', 'Lsp'}, 'ExitQuality, Lsp', 'A1:B1')
xlswrite('Results', xe, 'ExitQuality, Lsp', 'A2')
xlswrite('Results', {'Total Drop', 'Microchannel Drop', 'DP12', 'DP23',
'DP34'}, 'Pressure', 'A1:E1')
xlswrite('Results', DP, 'Pressure', 'A2')
xlswrite('Results', DP3, 'Pressure', 'B2')
xlswrite('Results', DP12, 'Pressure', 'C2')
xlswrite('Results', DP23, 'Pressure', 'D2')
xlswrite('Results', DP34, 'Pressure', 'E2')
xlswrite('Results', {'Diode Model'; 'T=aV+b'; 'a'; 'b'}, 'Models',
'A1:A4')
xlswrite('Results', DiodeModel', 'Models', 'B3:B4')
xlswrite('Results', {'Resistor Model'; 'R=aT^2+bT+c';'a';'b';'c'},
'Models', 'A5:A9')
xlswrite('Results', ResistorModel, 'Models', 'B7')
xlswrite('Results', {'Heat Loss Model'; 'q=a*T+b'; 'a'; 'b'}, 'Models',
'A11:A14')
xlswrite('Results', LocalLossModel, 'Models', 'B13')
xlswrite('Results', {'Reynolds Number', 'Bond Number'}, 'Bo_Re',
'A1:B1')
xlswrite('Results', Re, 'Bo_Re', 'A2');
xlswrite('Results', Bo', 'Bo_Re', 'B2');
xlswrite('Results', Bl, 'Boiling Number');
xlswrite('Results', Npch, 'Phase Change Number');
```

Appendix D Non-Uniform Heating Plots

This section contains the set of graphs describing the data collected under hotspot and non-uniform peak heating conditions as described in Section 6.3. The graphs are grouped by case and contain all of the data recorded.

Case 1a: Central Transverse Hotspot

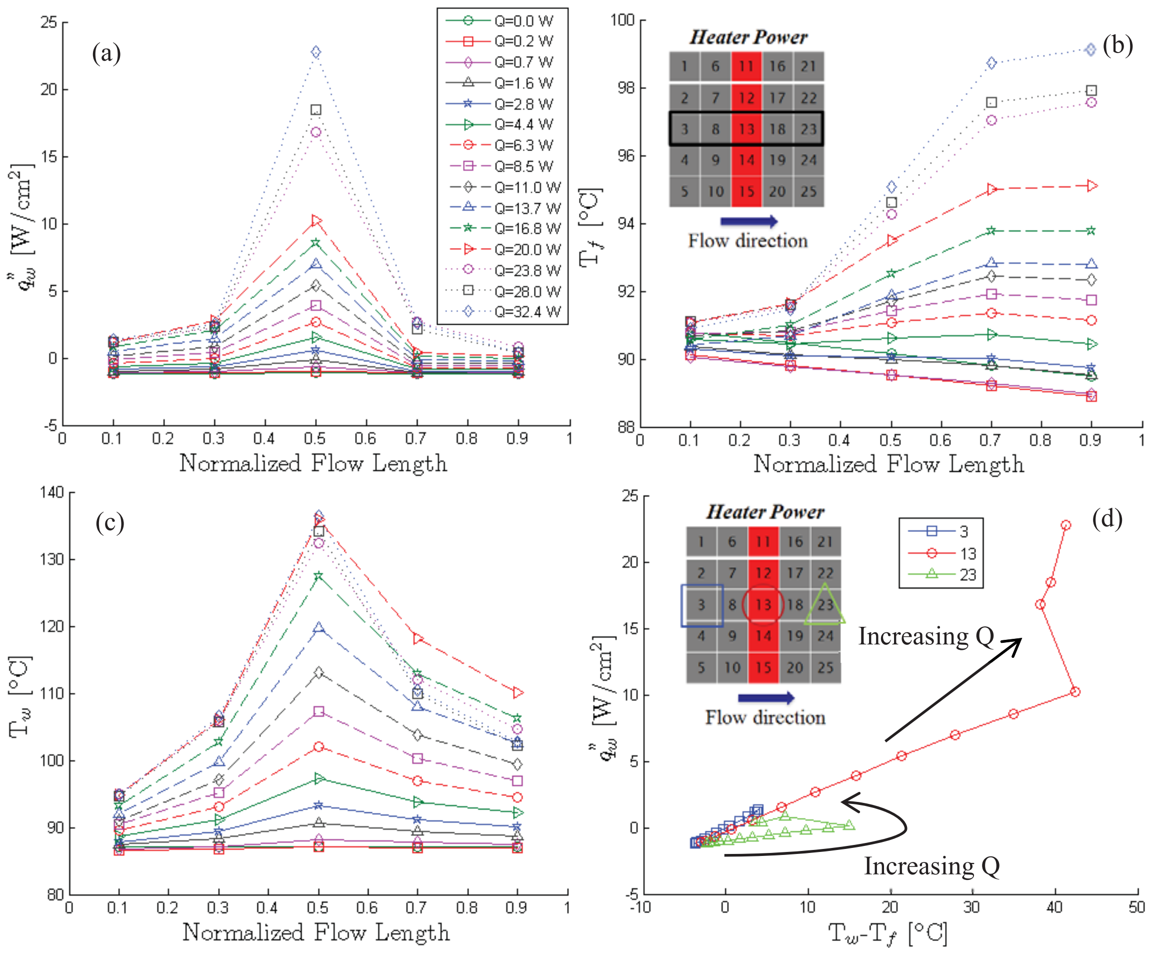


Figure D.1. (a) Local heat flux transferred to the fluid, (b) fluid temperature along the flow length, (c) wall temperature along the flow length, and (d) wall heat flux plotted against the wall excess temperature at increasing power input levels for a central transverse hotspot.

Case 1b: Central Streamwise Hotspot

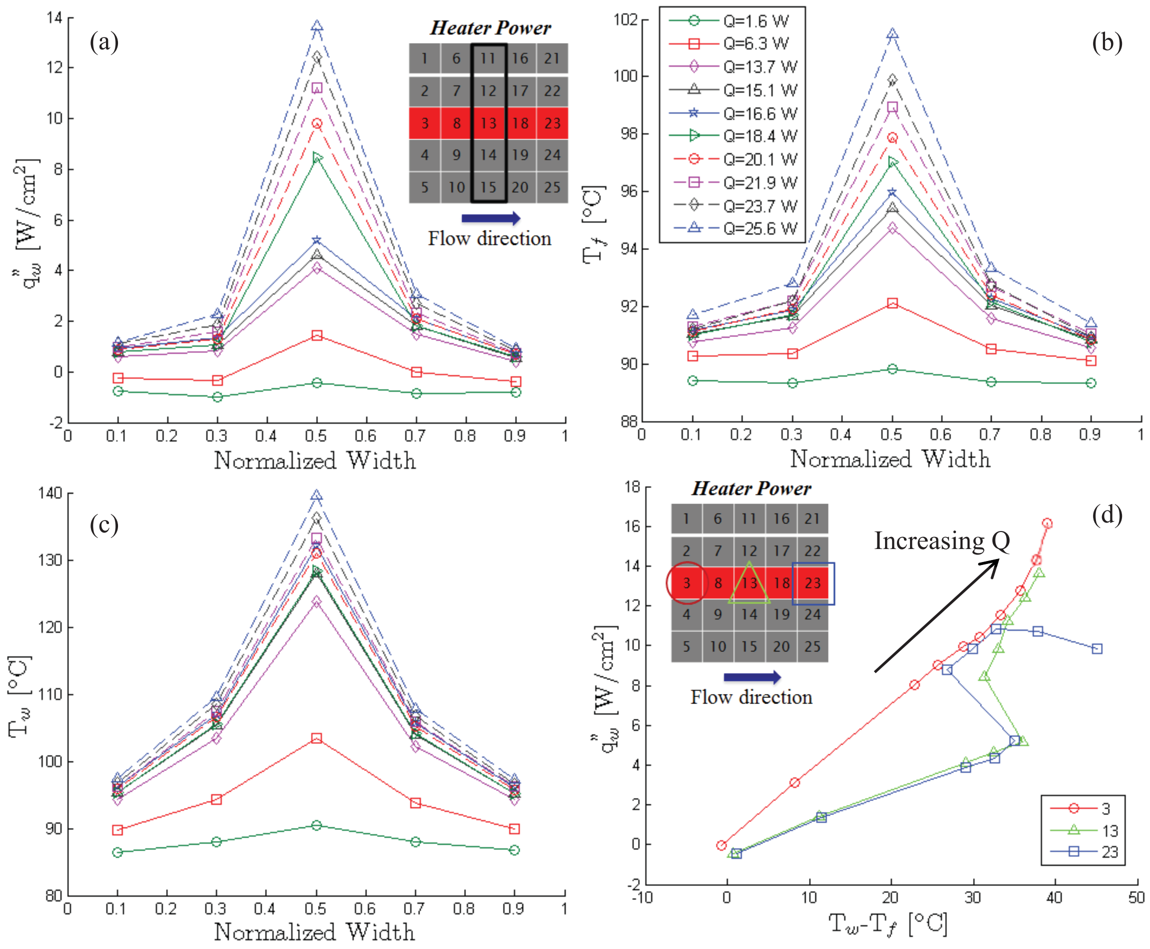


Figure D.2. (a) Local heat flux transferred to the fluid, (b) fluid temperature along the flow length, (c) wall temperature along the flow length, and (d) wall heat flux plotted against the wall excess temperature at increasing power input levels for a central streamwise hotspot.

Case 1c: Inlet Transverse Hotspot

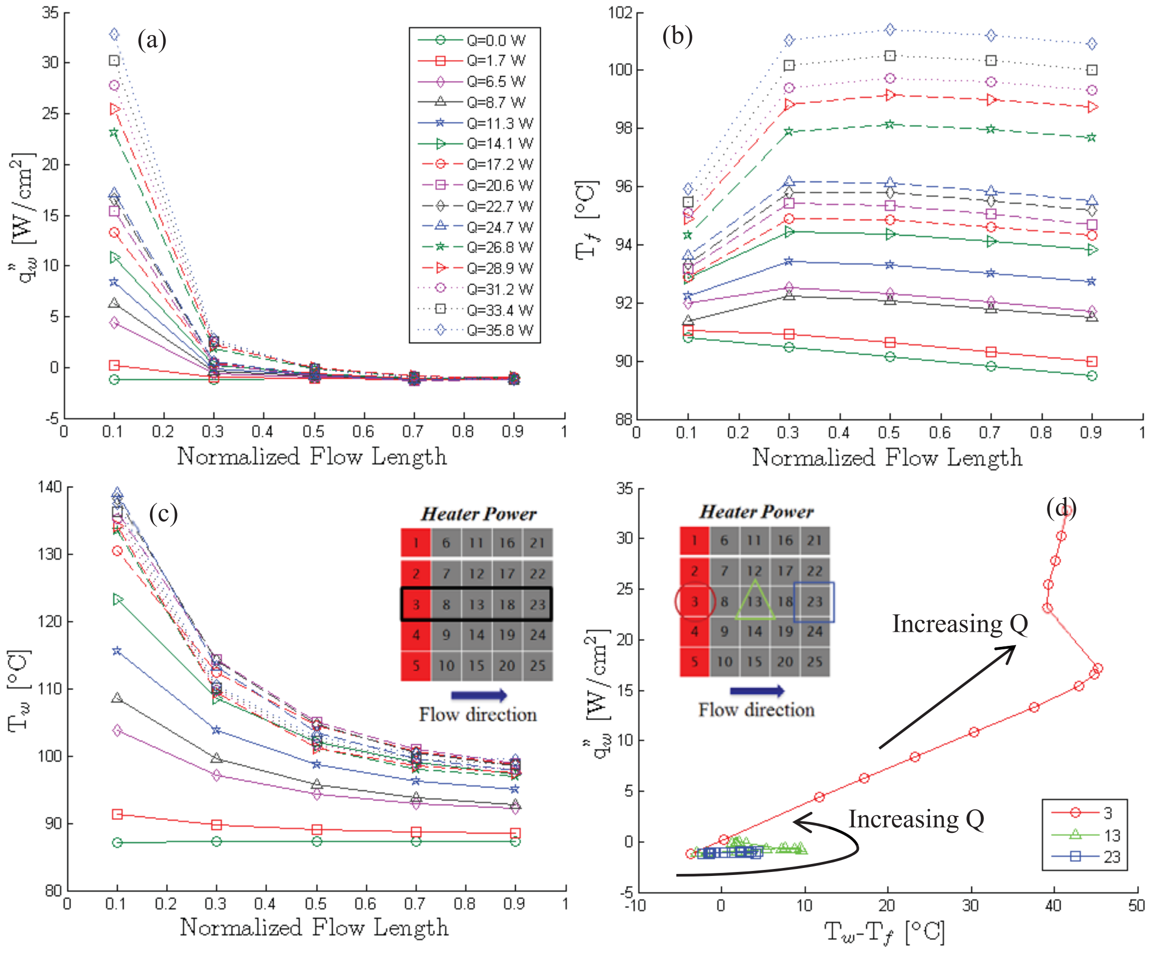


Figure D.3. (a) Local heat flux transferred to the fluid, (b) fluid temperature along the flow length, (c) wall temperature along the flow length, and (d) wall heat flux plotted against the wall excess temperature at increasing power input levels for an inlet transverse hotspot.

Case 1d: Double Transverse Hotspot

The fourth hotspot heating profile tested was Case 1d, dual transverse hotspots. Power was supplied to 5 transverse heater elements located at both the inlet and outlet of the flow stream while the remaining 15 resistors were unpowered. As the power input increases, two local heat flux maxima occur at the inlet and outlet active heater elements; the heat flux to the fluid is slightly higher at the inlet as seen along the central streamwise sensors in Figure D.4a at increasing power input levels.

As the power level increases, the wall temperature becomes highest at the hotspot locations. The outlet hotspot has a slightly higher wall temperature than the inlet hotspot due to the fluid temperature increase along the flow length. At a power input of 65 W, the inlet hotspot wall temperature is 133.3 °C and the outlet hotspot wall temperature is 133.7 °C. The minimum wall temperature occurred at the middle of the heat sink between the two hotspots (104.1 °C). This temperature is on the same order as the outlet wall temperature of Case 1a (102.5 °C), but larger than the Case 1a inlet wall temperature (94.6 °C). The wall temperatures measured across the central streamwise sensors at increasing input power levels are shown in Figure D.4c.

Boiling curves are shown in Figure D.4d for sensors 3, 13, and 23. Boiling begins at the outlet (inlet) hotspot at a total input power of 27.5 W (30.6 W) and a local heat flux of 11.3 W/cm^2 (14.4 W/cm^2) with a wall excess temperature of 30.2 °C (32.3 °C). Boiling is suppressed up to a comparatively higher heat flux at the inlet due to the larger developing-flow single-phase heat transfer coefficient that mitigates surface superheat.

Images extracted from high-speed videos (Supplementary Video 4) for the dual transverse hotspots at different power levels of 21.9 W, 33.7 W, 48.2 W, and 65.0 W are shown in Figure D.5. In this case, boiling incipience occurs at both the inlet and outlet hotspots for power levels above 21.9 W; however this boiling does not occur across all channels until the power input is increased further. At high power levels vigorous boiling occurs at the outlet hotspot, and partial dryout is observed in some channels. Vapor bubbles formed at the inlet hotspot affect the downstream flow regime and boiling incipience at the outlet hotspot. These bubbles coalesce with those formed at the downstream hotspot to form large vapor regions and partial dryout. For similar power levels and heat fluxes, this phenomenon is not seen in the other hotspot cases, and is unique to dual hot spots. Additionally, the fluid reaches the saturation temperature earlier along the flow length, causing partial dryout to occur in several channels prior to reaching the outlet hotspot at high power levels.

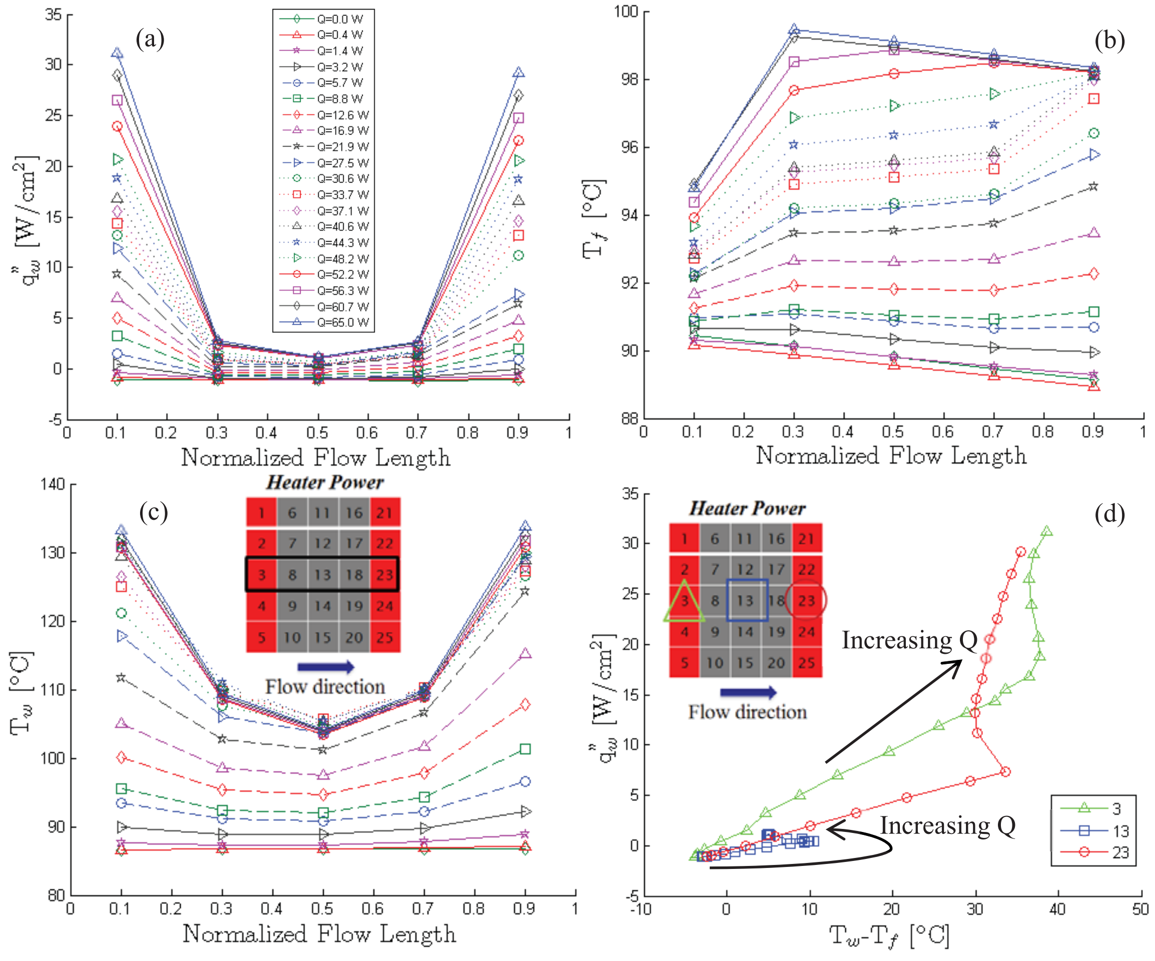


Figure D.4. (a) Local heat flux transferred to the fluid, (b) fluid temperature along the flow length, (c) wall temperature along the flow length, and (d) wall heat flux plotted against the wall excess temperature at increasing power input levels for a double transverse hotspot.

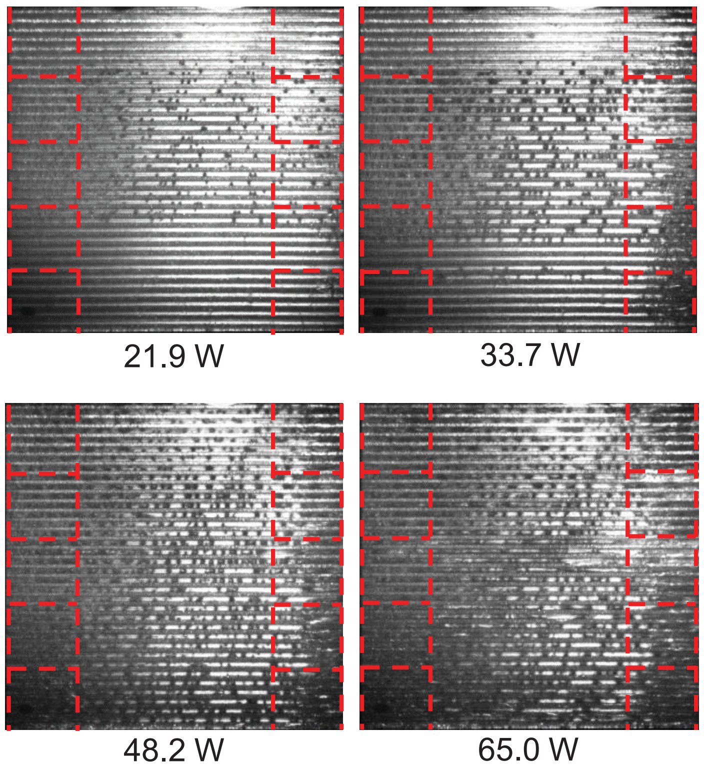


Figure D.5. Images at increasing power levels for dual transverse hotspots extracted from high-speed video. Red lines indicate the locations of the heated sensors.

Case 2a: Non-Uniform Transverse Peak

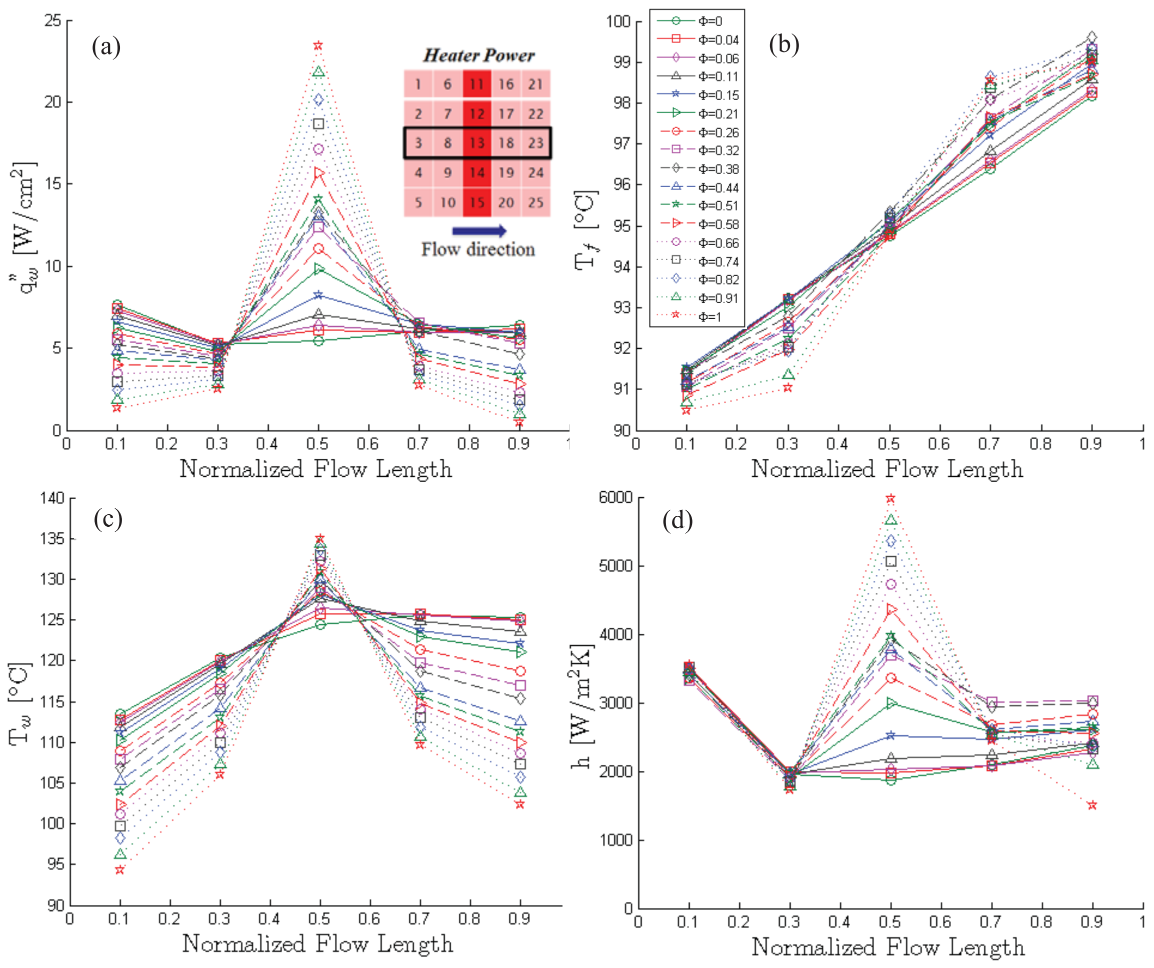


Figure D.6. (a) Local heat flux transferred to the fluid, (b) fluid temperature along the flow length, (c) wall temperature along the flow length, and (d) heat transfer coefficient along the flow length at increasing degrees of nonuniformity between the heat flux at the peak and the background heater locations for Case 2a.

Case 2b: Non-Uniform Streamwise Peak

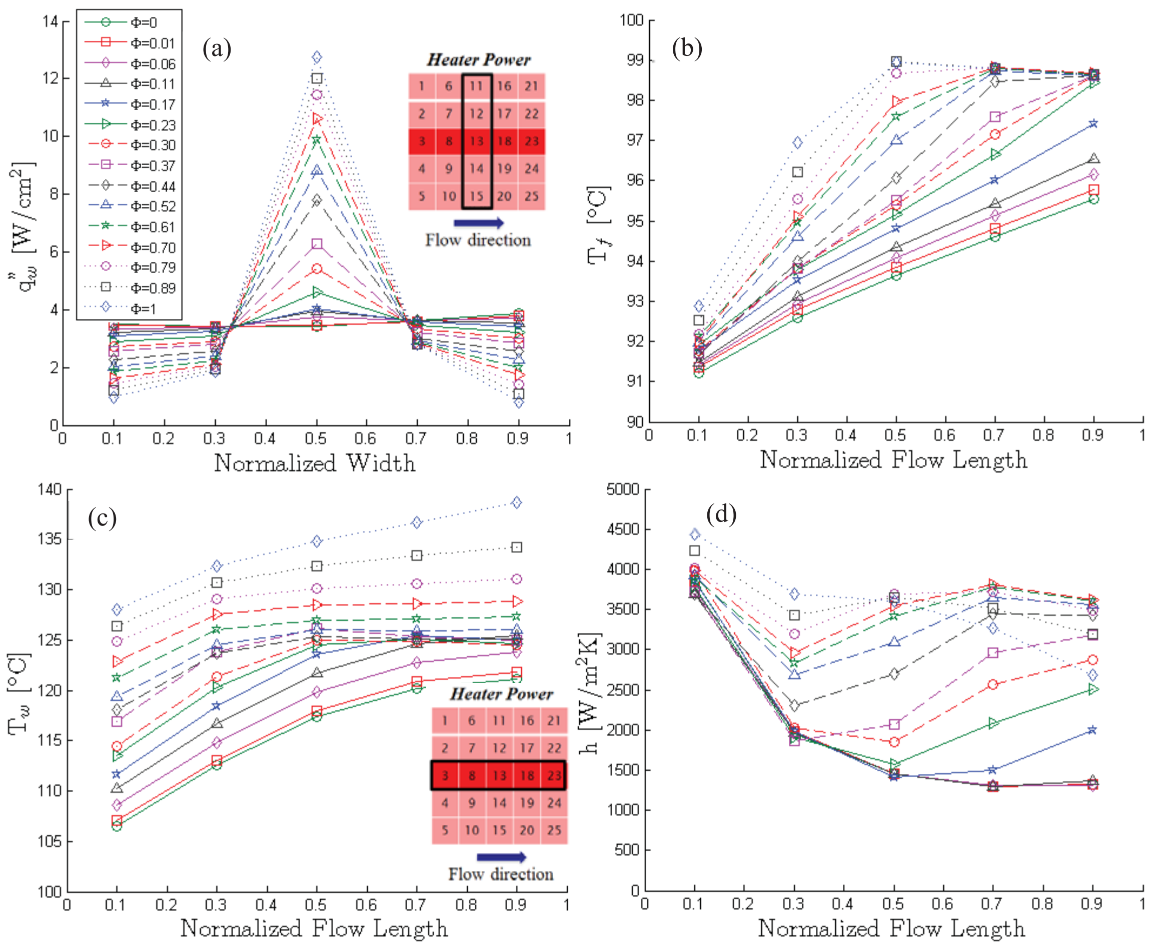


Figure D.7. (a) Local heat flux transferred to the fluid, (b) fluid temperature along the flow length, (c) wall temperature along the flow length, and (d) heat transfer coefficient along the flow length at increasing degrees of nonuniformity between the heat flux at the peak and the background heater locations for Case 2b.

Appendix E Non-Uniform Heating in a Copper Heat Sink

A second set of experiments were conducted using non-uniform heating profiles imposed on an attached copper microchannel heat sink[2]. The channels are the same dimensions as the test piece described in Section 6.1.1, but the heat sink has a thicker base. Additionally, the heat sink itself it attached to a smooth silicon thermal test chip adding a contact resistance that does not exist in the previous work.

E.1 Test Section

The microchannel test section used in this study consists of a microchannel heat sink provided by Wolverine Tube, Inc. and a thermal test chip provided by IBM, and it is shown in Figure E.1. A transparent, polycarbonate manifold cover plate seals and routes the working fluid through the microchannel heat sink. The heat sink is made of copper 110 and has a base area of 12.7 mm × 12.7 mm. The channels have a width of 259 μm and a depth of 342 μm. The heat sink is placed on top of a smooth silicon thermal test chip; no thermal interface material (TIM) is used. The thermal test chip has a base area of 21.3 mm × 21.3 mm and has a thickness of approximately 0.81 mm. A 5 × 5 array of resistance heaters and resistance temperature detectors (RTDs) is fabricated on the underside of the thermal test chip.

Since the microchannel heat sink has a smaller base area than the silicon thermal test chip, the heat sink was carefully placed in the center so that it only covered the

[2] The author would like to thank Peter Beucher of Wolverine Tube, Inc. and Katie Rivera of IBM for providing the copper microchannel heat sink and silicon thermal test chip.

middle 3 × 3 array of heater elements, as shown in Figure E.2. Silicone rubber sheets were placed on the sides of the heat sink to force fluid to flow through the channels and prevent it from bypassing the heat sink. In this configuration, fluid flows over a portion of the thermal test chip, through the microchannels, and over another portion of the thermal test chip. In this way, it is possible for the fluid to increase in temperature before entering and after exiting the channels. Since no TIM was used, the heat sink is held in place by pressure from the cover plate; the interfacial contact resistance was estimated to be 0.5 x 10^{-4} m^2K/W [100]. With the addition of an attached copper heat sink combined with a lack of a TIM, it is estimated that there is significantly more lateral conduction and heat spreading through the substrate as compared to in the test chip presented in Section 6.3.

E.2 <u>Calibration</u>

A calibration of the RTDs, the resistance heaters, and the heat loss to the test section was performed. The calibration performed for the RTDs was done in an oven at six temperatures ranging from 30 to 100 ˚C and the data is plotted is Figure E.3. A linear least squares regression of voltage to temperature was fit to each of the RTDs The calibration performed for the heaters was done in an oven at five temperatures ranging from 40 to 100 ˚C and the data is plotted in Figure E.4. A quadratic regression line was fit to each of the heaters to relate the measured temperature to the resistance. After the RTDs and resistance heaters were calibrated, the test section was assembled. In order to calibrate the heat loss from the test section, seven power levels ranging from 0 to 0.55 W were applied while there was no flow through the test section. The heat loss via natural convection and radiation on the outer surfaces of the test section is measured as the amount of power supplied to the heaters and the temperatures are measured. The heat loss data as a function of temperature is plotted in Figure E.5 with a linear fit to the data.

E.3 Experimental Procedures

The flow loop is the same as that described in Section 8.1.2. The working fluid, HFE-7100, was chosen for its relatively low boiling point (61 ˚C at atmospheric pressure). Experiments are conducted at a single mass flux of 770 kg/m^{2}s. The fluid is heated to approximately 51 ˚C at the inlet to the test chip. Both the flow rate and the inlet temperature were at maintained at a constant value throughout the test.

E.4 Data Reduction

Modifications were made to the data reduction analysis described in Section 6.2 and shown in Appendix C to account for the extra heat spreading in the test section. An additional contact resistance between the silicon thermal test chip and copper microchannel heat sink was added, as well as a separate calculation for the regions upstream and downstream of the heat sink where the fluid is in direct contact with the silicon test chip. A schematic diagram of the energy flow in the test section is shown in Figure E.6. Finally, fluid properties were updated to account for the change in fluid from FC-77 to HFE-7100.

E.5 Results and Discussion

Two cases were tested: (1) a uniform heating case where all nine heaters under the microchannel heat sink were activated and (2) a single hotspot case where only the central heater was activated. Wall temperatures, heat transfer coefficients, and wall heat fluxes along the flow direction for the uniform heating case are shown in Figure E.7. In this case, significant lateral conduction within the silicon thermal test chip was observed, as expected. The unheated regions upstream and downstream of the heat sink showed a considerable rise in the wall temperature, with a relatively small peak in the center of the heated region. The heat transfer coefficients and wall heat fluxes above the heated regions are significantly larger than those above the unheated regions, as expected.

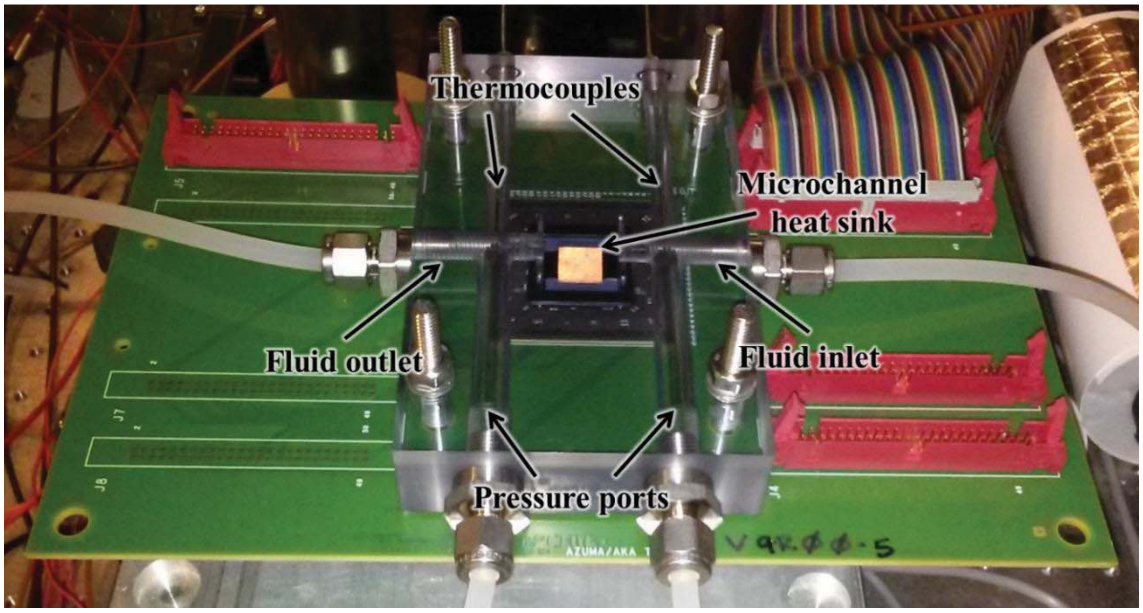
The wall temperatures, heat transfer coefficients, and wall heat fluxes along the flow direction for a single hotspot case are shown in Figure E.8. Like the uniform case, a single hotspot case displays significant heat spreading in both the silicon thermal test chip and the copper microchannel heat sink. The highest local wall temperature was measured above the active heater element, as expected; however, a significant rise in the wall temperatures at the outer parts of the chip was still observed. The highest wall temperature recorded was 97.5 ˚C for a single hotspot above the central heater for a power input level of 38.1 W; the uniform heating case produced a maximum wall temperature of 85.5 ˚C despite a total power input level of 88.7 W, more than twice that of the hotspot case.

In the uniform heating case, boiling incipience was observed at a power input level of 33.0 W. Boiling was observed in all of the channels in the heat sink. In the single hotspot case, boiling incipience was observed at a power input level of 34.6 W and a maximum wall temperature drop of approximately 3 ˚C was observed due to boiling incipience. Boiling was only observed in the region above the hotspot as seen in Figure E.9. In both heating cases, boiling begins at about the same total power input despite the large difference in the base heated area. This is due to the large amount of heat spreading within the thermal test chip and heat sink that effectively smoothed out the hotspot.

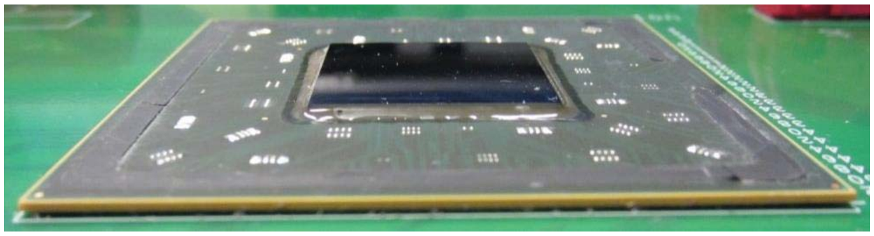
E.6 Conclusions

In this work, a copper microchannel heat sink was attached to a silicon thermal test chip and both a uniform and a single hotspot heating case were tested. The thicker heat sink base combined with a lack of a TIM between the test chip and heat sink increased the lateral conduction within the test section. This led to an increased thermal resistance between the heaters and the fluid, causing more heat to flow in the lateral direction as compared to the results seen in Section 6.3. The addition of a TIM between the silicon and copper would likely decrease heat spreading to the non-heated regions.

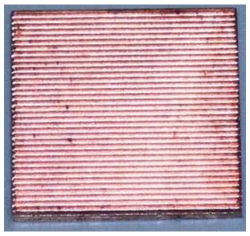
At higher power input levels, the fluid entering the test section is preheated via lateral conduction prior to entering the heat sink. If enough heat is supplied to this region, the fluid may boil prior to entering the channels and cause flow instabilities in the test section and trap vapor in the inlet manifold. It is recommended that a microchannel heat sink be placed as close to the inlet of the test section as possible to avoid this situation.



(a)



(b)



(c)

Figure E.1. (a) The assembled test section, (b) the silicon thermal test chip, and (c) the copper microchannel heat sink.

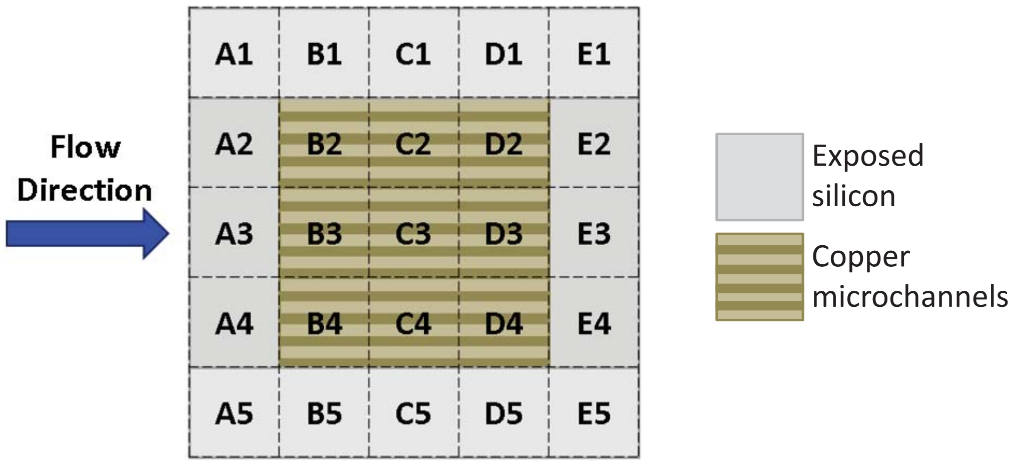


Figure E.2. A diagram of the microchannel heat sink in relation to the heater locations on the thermal test chip.

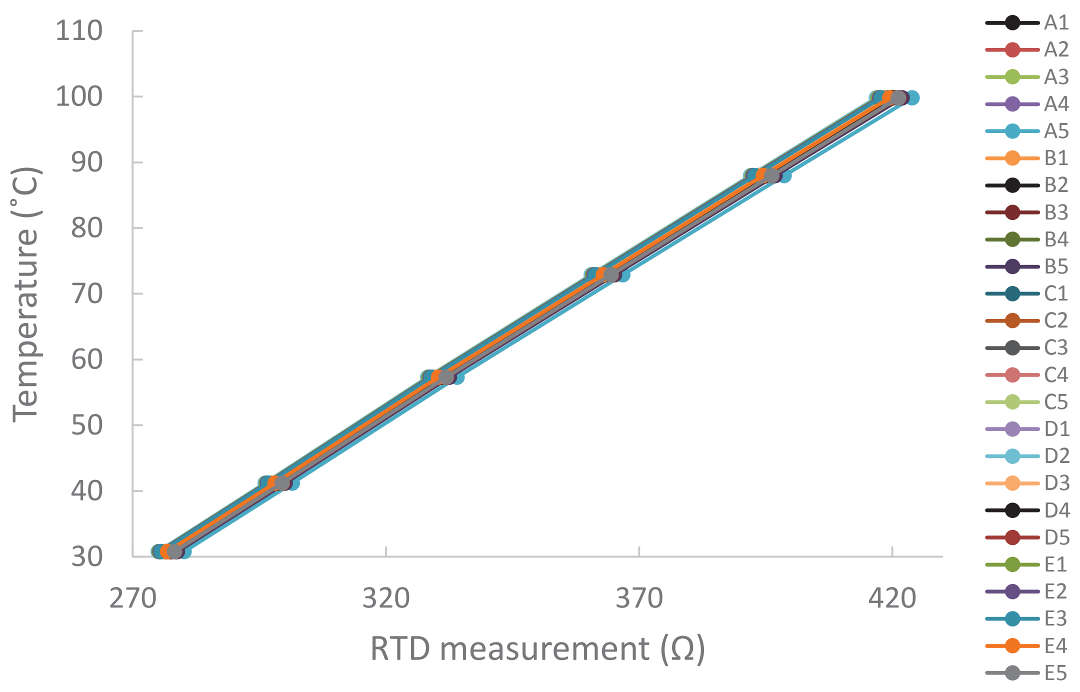


Figure E.3. Calibration lines for each RTD in the thermal test chip.

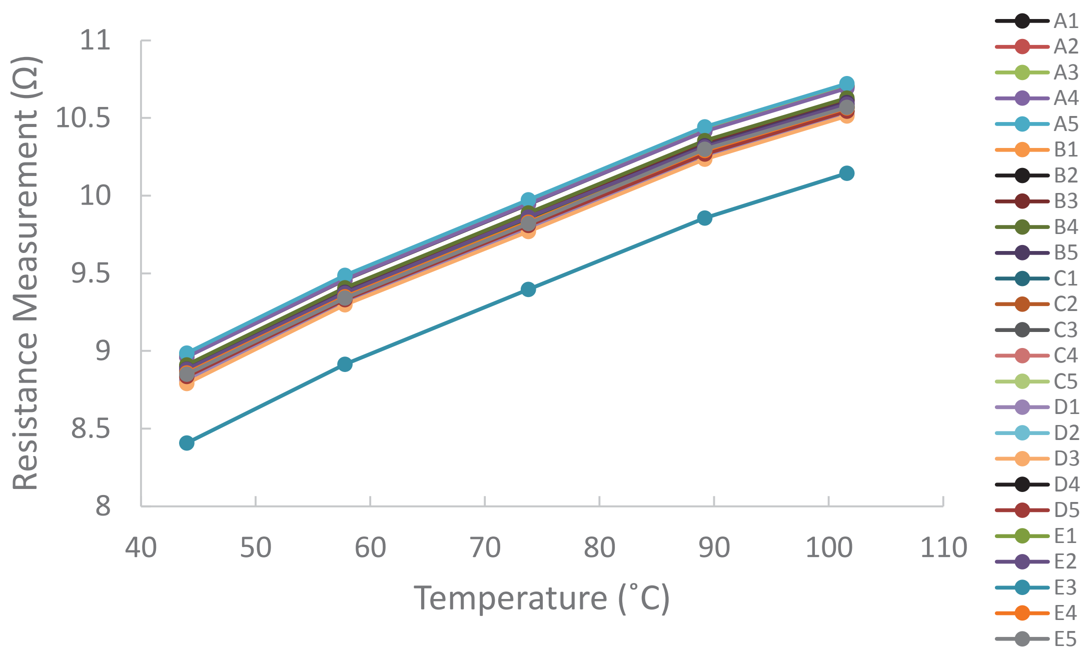


Figure E.4. Calibration lines for each heater element in the thermal test chip.

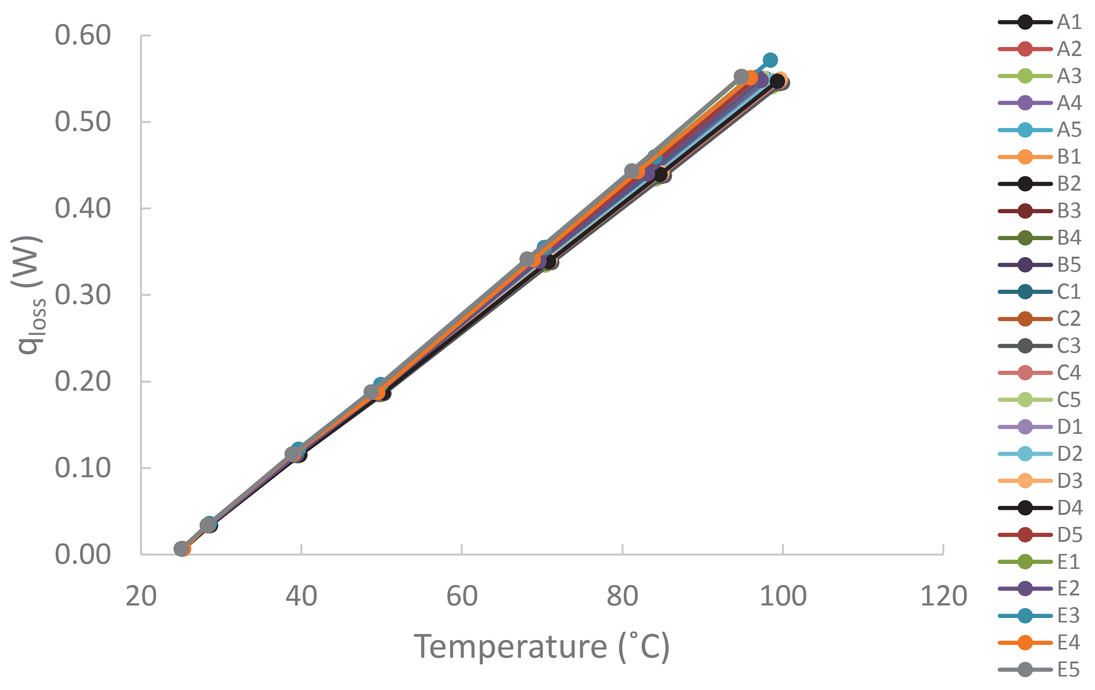


Figure E.5. Calibration lines for the heat loss in the assembled test section.

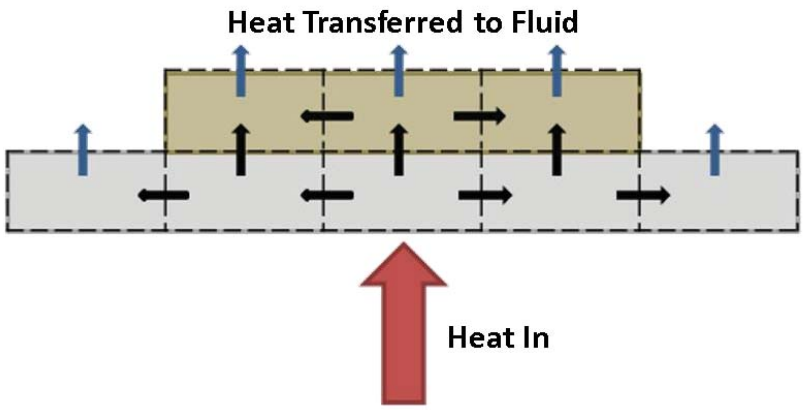


Figure E.6. A diagram of the flow of heat through a cross section of the test section.

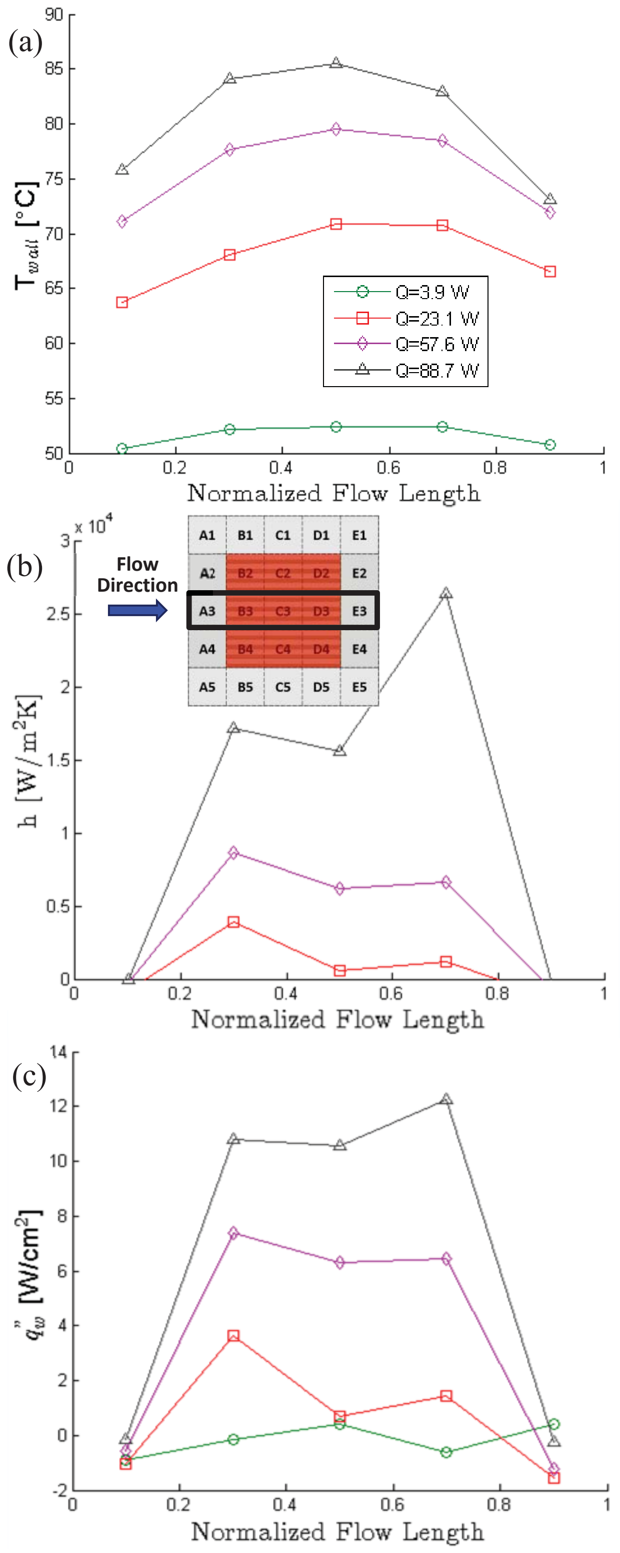


Figure E.7. (a) The local wall temperature, (b) heat transfer coefficient, and (c) heat flux transferred to the fluid over the flow length for a uniform heating case.

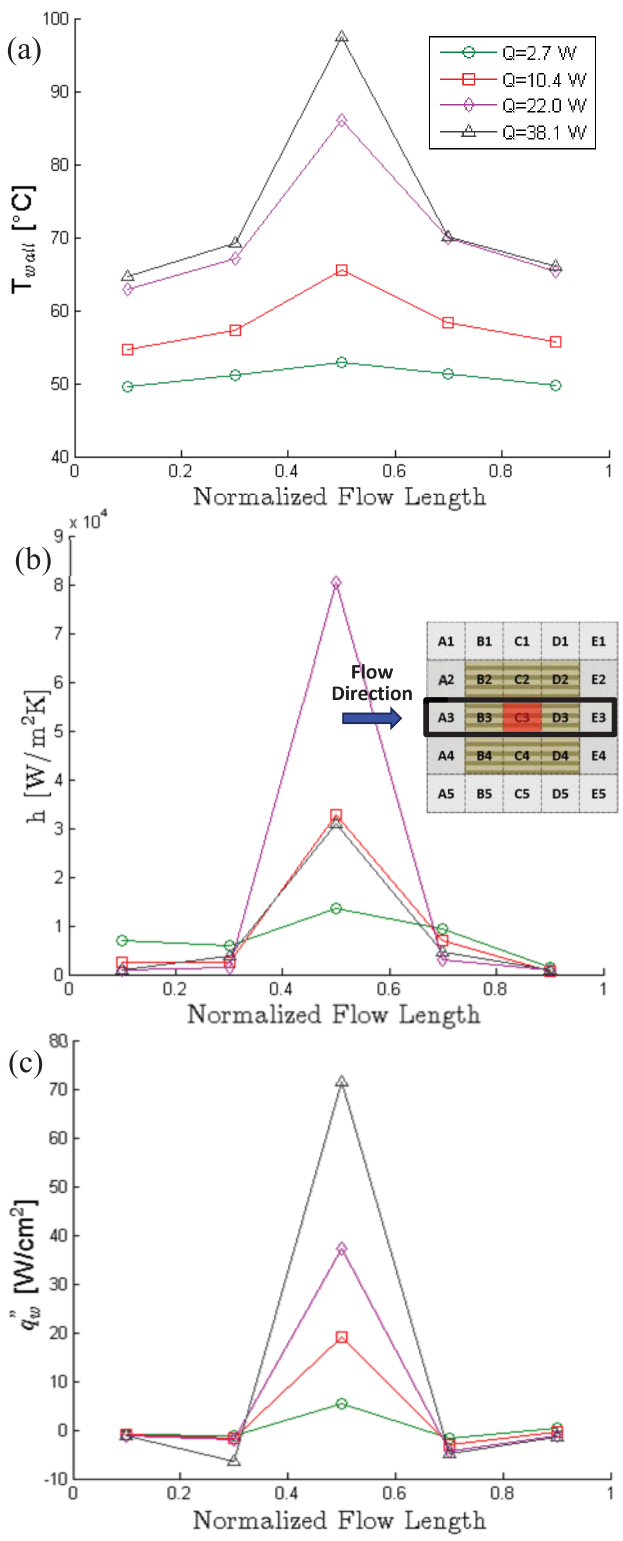


Figure E.8. (a) The local wall temperature, (b) heat transfer coefficient, and (c) heat flux transferred to the fluid over the flow length for a single hotspot case.

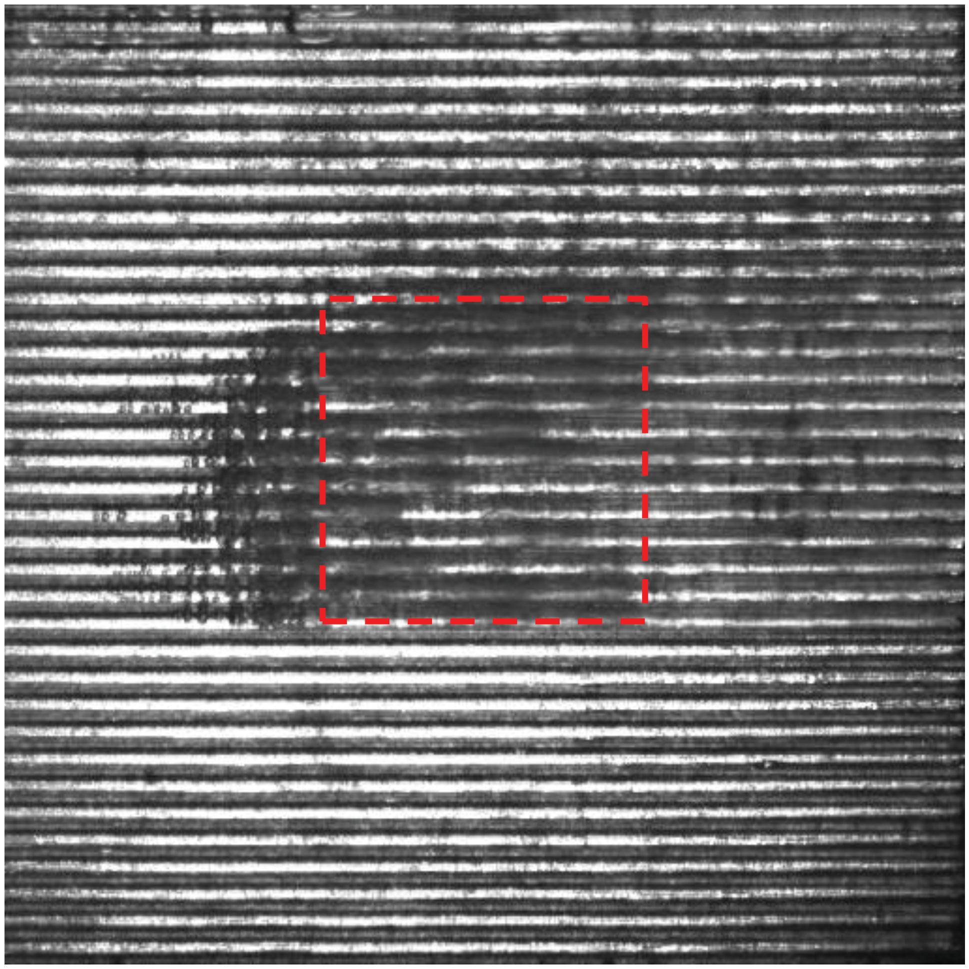


Figure E.9. Image taken at 38.1 W for a single hotspot. The red dashed lines indicate the location of the hotspot.

Appendix F Critical Heat Flux Plots

This section contains a set of graphs describing the critical heat flux data collected using various hotspot heating conditions as described in Section 8.2. The graphs contain all of the data recorded for each case.

Uniform Heating

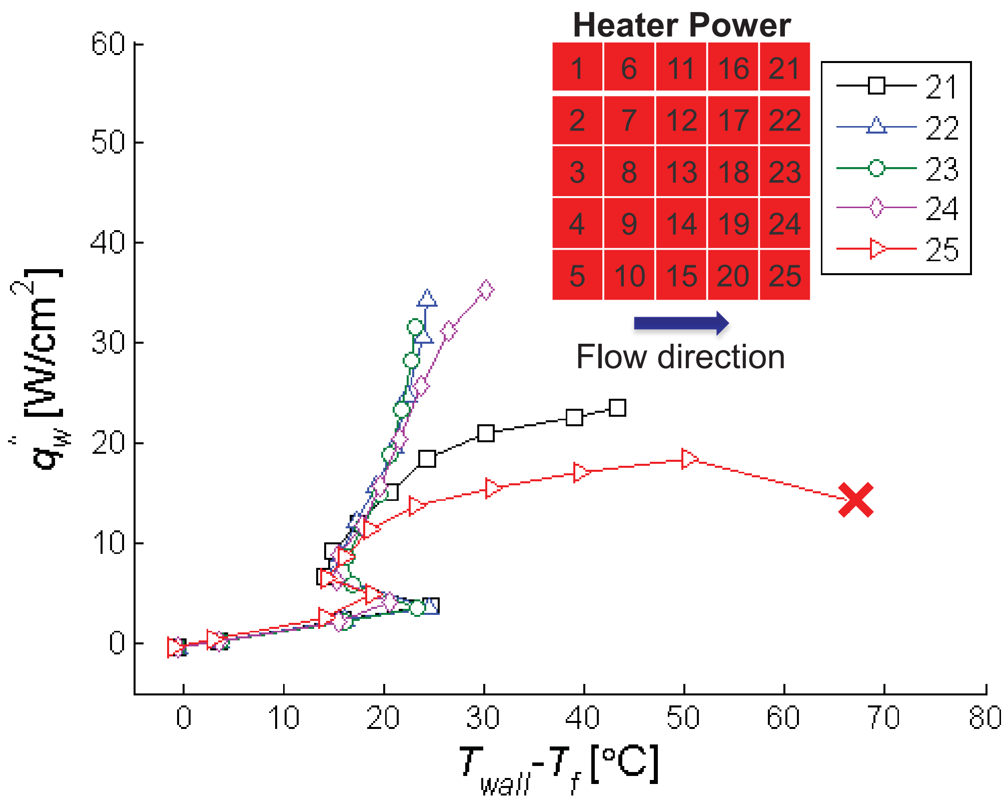


Figure F.1. Heat flux transferred to the fluid plotted against the wall excess temperature for a uniform heating profile. “X” indicates the location of CHF.

Inlet Transverse Hotspot

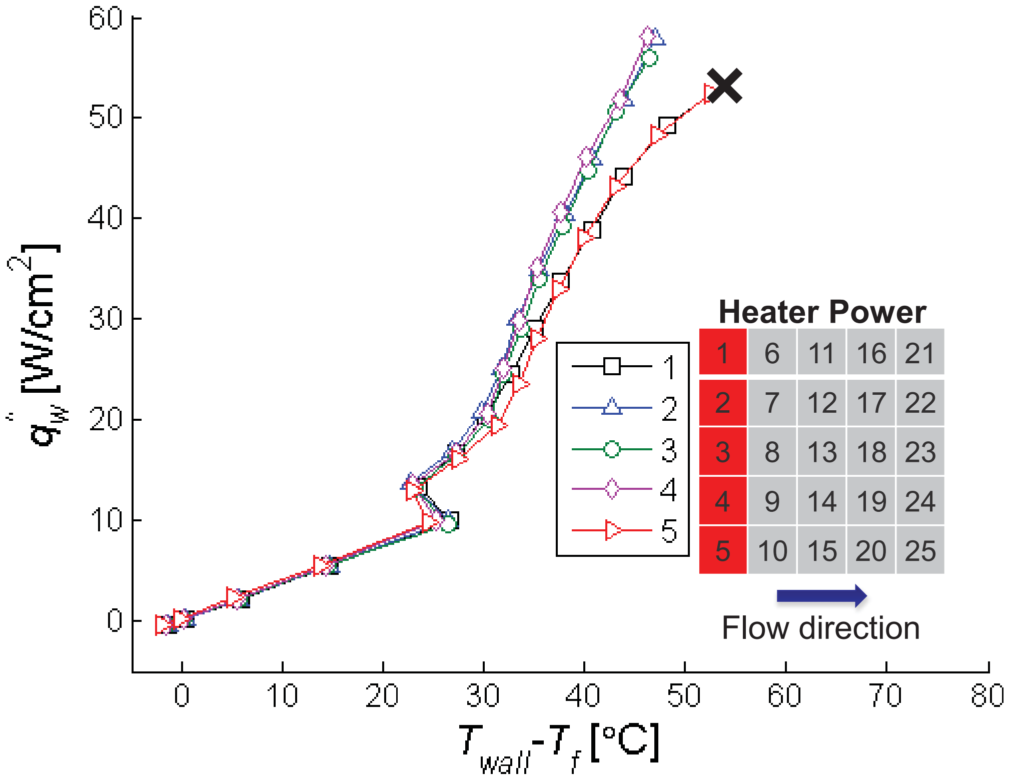


Figure F.2. Heat flux transferred to the fluid plotted against the wall excess temperature for an inlet transverse hotspot. "X" indicates the location of CHF.

Central Transverse Hotspot

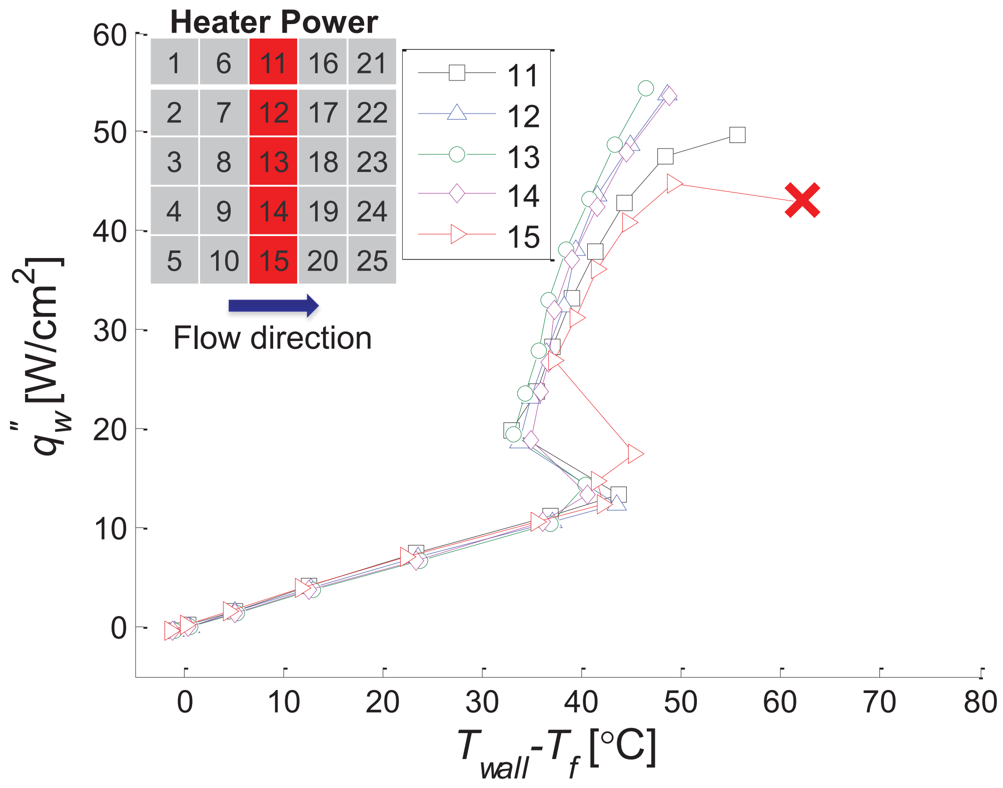


Figure F.3. Heat flux transferred to the fluid plotted against the wall excess temperature for a central transverse hotspot. “X” indicates the location of CHF.

Outlet Transverse Hotspot

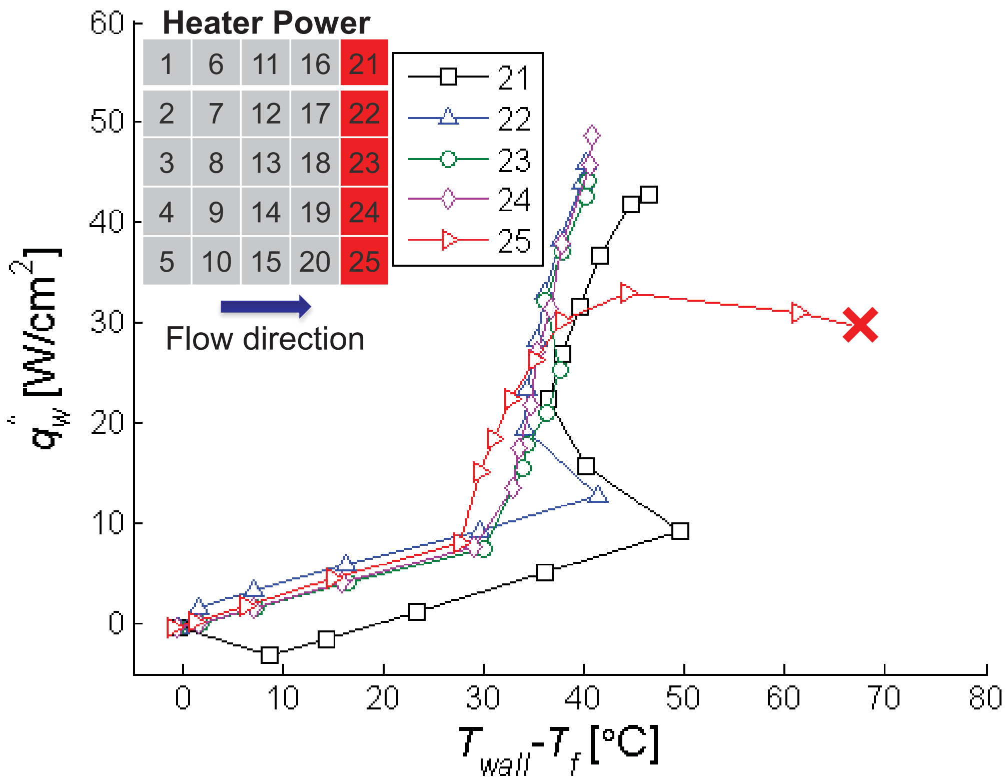


Figure F.4. Heat flux transferred to the fluid plotted against the wall excess temperature for an outlet transverse hotspot. “X” indicates the location of CHF.

Central Streamwise Hotspot

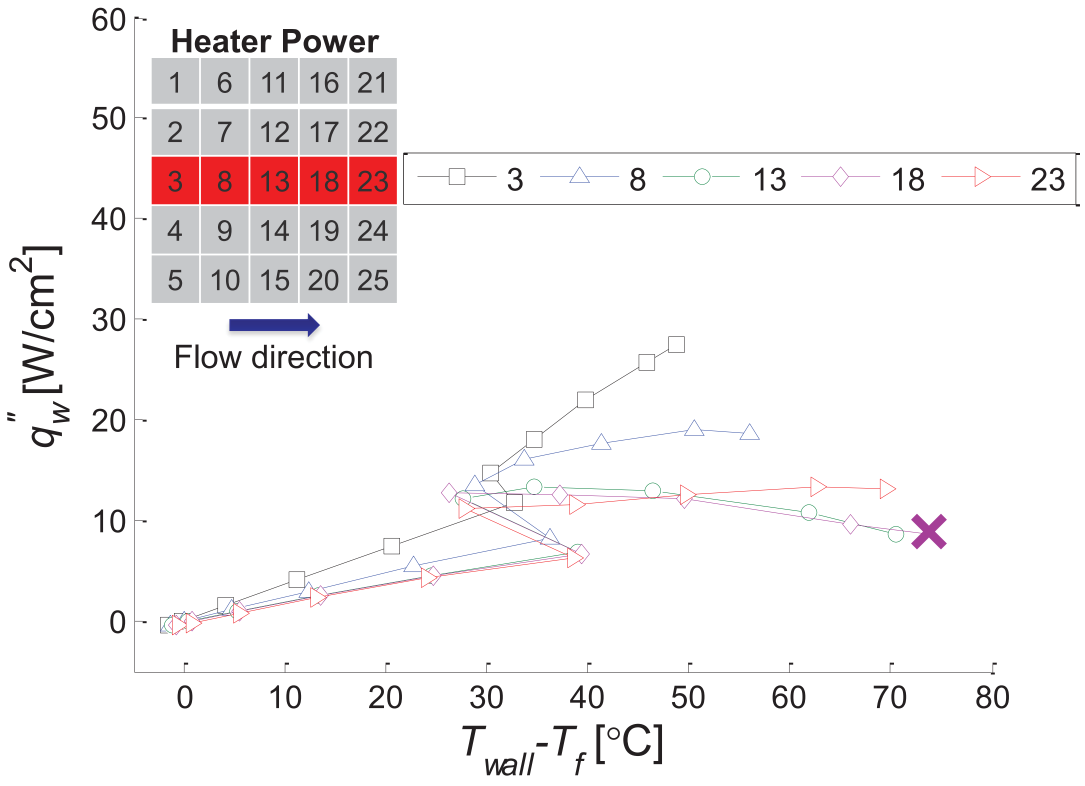


Figure F.5. Heat flux transferred to the fluid plotted against the wall excess temperature for a central streamwise hotspot. “X” indicates the location of CHF.

Dual Transverse Hotspot

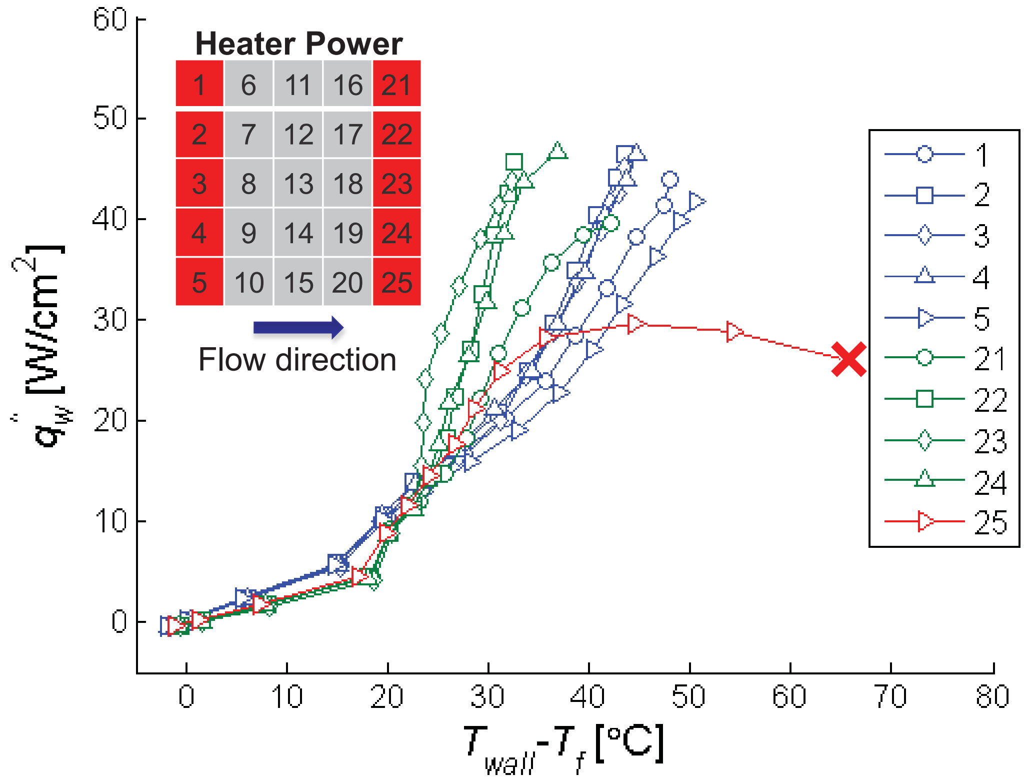


Figure F.6. Heat flux transferred to the fluid plotted against the wall excess temperature for a dual transverse hotspot. “X” indicates the location of CHF.

Appendix G MATLAB Script of the Microchannel Heat Sink Computational Model

A MATLAB script was developed to model two-phase heat transfer in a microchannel heat sink as discussed in Section 7.1. The following script takes a set of user defined inputs and calculates the local temperatures within the base of the heat sink, as well as local heat transfer coefficients and wall heat fluxes. It is split into several functions; Table G.1 displays the function number, name, description, and page number where it can be found. The author would like to thank Professor Tine Baelmans of KU Leuven for providing some of the logic for the code.

Table G.1. A list of all of the functions for the computational model.

Function Number	Function Name	Description	Page Number
31	compmodel.m	The main program	256
32	condbasemat.m	Generates coefficient matrices for conduction analysis	258
33	conduction.m	Plane-by-plane TDMA solver for conduction in the heat sink base	265
34	convection2.m	Convection analysis	268
35	discretize.m	Discretize the heat sink base	270
36	genmesh.m	Generates identification matrices for the domain	271
37	gui_input2.m	GUI to read inputs from user	272
38	gui_output.m	GUI to display the results	281
39	heattranscoeff.m	Calculates the heat transfer coefficient	284
40	inputs.m	Contains fluid properties and heating profiles	285
41	pressuredrop.m	Calculates the pressure drop	287
42	singlephasedp.m	Calculates the single-phase pressure drop	288
43	singlephaseh.m	Calculates the single-phase heat transfer coefficient	289
44	tdma.m	TDMA solver	289
45	tdmaline.m	Line-by-line TDMA solver	290
46	twophaseh.m	Calculates the two-phase heat transfer coefficient	292
47	vaporphaseh.m	Calculates the heat transfer coefficient for vapor flow	293

Function 31. compmodel.m

```
clear all
clc
[dims,matp,flow,getdata,filename]=inputs();
if getdata
    load(filename)
else
    [mesh]=discretize(dims);
    [mesh]=genmesh(dims,mesh);
    Nx=mesh.Nx_c+mesh.Nx_f+mesh.Nx_ef;
    Ny=mesh.Ny;
    Nz=mesh.Nz;
    T=ones(Nx,Ny,Nz)*flow.T_in;      %initialize T everywhere
    Tf=ones(dims.N,Ny)*flow.T_in;    %initialize Tf everywhere
    h=ones(dims.N,Ny)*5000;          %initialize h everywhere
    xe=zeros(dims.N,Ny)-1;           %initialize xe everywhere
    q_in=zeros(Nx,Ny);               %calculate input heat transfer
    q_w=zeros(dims.N,Ny);
    dx=[mesh.dx_c mesh.dx_f mesh.dx_ef];
    for i=1:Nx
        for j=1:Ny
            dx_val=dx(mesh.id.label(i));
q_in(i,j)=flow.Q_in(mod(floor(i*5/Ny),5)+1,mod(floor(j*5/Ny),5)+1)*mesh
.dy*dx_val;
            q_w(mesh.id.num(i),j)=q_w(mesh.id.num(i),j)+q_in(i,j);
        end
    end
    q_w=q_w/mesh.dy/(dims.w+2*dims.d);
end
%iterate between conduction in base and heat transfer in fins
tic;
counter=0;
err=1;
while err>0.000001
    Told=T;
    [T]=conduction(dims,matp,mesh,T,Tf,h,q_in);
    [DP,DPsp,DPtp]=pressuredrop(dims,matp,mesh,flow,xe,Tf);
    ws=zeros(dims.N,2);
    ws(:,1)=(max(q_w,[],2)-min(q_w,[],2))./mean(q_w,2);
    for i=1:dims.N
        ws(i,1)=(max(q_w(i,:))-min(q_w(i,:)))/mean(q_w(i,:));
        if isnan(ws(i,1))
            ws(i,1)=0;
        end
        x1=find(mesh.id.num==i);
        ws(i,2)=(max(max(T(x1,:,Nz)))-
min(min(T(x1,:,Nz))))/mean(mean(T(x1,:,Nz)));
    end
    [Tf,h,q_w,xe]=convection2(dims,matp,mesh,flow,T,Tf,q_w,DP,xe,ws);
    err=max(max(max(abs((Told-T)./Told))));
    counter=counter+1;
    if counter==5
        fprintf('still going...\n');
    end
    if ~mod(counter,10)
        fprintf('iteration: %i ave T: %2.3f
err: %1.6f\n',counter,mean(mean(mean(T))),err);
    end
    if counter>=1000
        fprintf('over 1000 iterations, error is %1.7f\n',err);
        err=0;
    end
    if err>15
        fprintf('diverging, %i iterations, error
is %1.7f\n',counter,err);
        err=0;
    end
end
toc;
clear Told counter i j dx dx_val x1
gui_output;
```

Function 32. condbasemat.m

```
function [a,b,c,d,e,f,g,ho]=condbasemat(mesh,dims,matp,h,q_in,Tf)
Nx=mesh.Nx_c+mesh.Nx_f+mesh.Nx_ef;
Ny=mesh.Ny;
Nz=mesh.Nz;
N=dims.N;
dc=dims.d;
id=mesh.id;
ks=matp.k_Si;
dy=mesh.dy;
dx_c=mesh.dx_c;
dx_f=mesh.dx_f;
dx_ef=mesh.dx_ef;
dx=[dx_c dx_f dx_ef];
dz=mesh.dz;
a_loss=0.003213171*25/Ny/Nx;  %constants from experiment to find q_loss
b_loss=-0.085901548*25/Ny/Nx; %q_loss=a_loss*T(Nx,Ny,1)+b_loss; %[W],
25 total
b=zeros(Nx,Ny,Nz);
c=zeros(Nx,Ny,Nz);
d=zeros(Nx,Ny,Nz);
e=zeros(Nx,Ny,Nz);
f=zeros(Nx,Ny,Nz);
g=zeros(Nx,Ny,Nz);
ho=zeros(Nx,Ny,Nz);
%interior cells
for i=2:Nx-1
    dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
    b(i,2:Ny-1,2:Nz-1)=ks*dy*dz/dx_val;
    dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
    c(i,2:Ny-1,2:Nz-1)=ks*dy*dz/dx_val;
    dx_val=dx(id.label(i));
    d(i,2:Ny-1,2:Nz-1)=ks*dx_val*dz/dy;
    e(i,2:Ny-1,2:Nz-1)=d(i,2:Ny-1,2:Nz-1);
    f(i,2:Ny-1,2:Nz-1)=ks*dx_val*dy/dz;
    g(i,2:Ny-1,2:Nz-1)=f(i,2:Ny-1,2:Nz-1);
end
a=b+c+d+e+f+g;
%top and bottom faces
for i=2:Nx-1
    dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
    b(i,2:Ny-1,1)=ks*dy*dz/dx_val;
    dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
    c(i,2:Ny-1,1)=ks*dy*dz/dx_val;
    dx_val=dx(id.label(i));
    d(i,2:Ny-1,1)=ks*dx_val*dz/dy;
    e(i,2:Ny-1,1)=d(i,2:Ny-1,1);
    f(i,2:Ny-1,1)=ks*dx_val*dy/dz;
    a(i,2:Ny-1,1)=b(i,2:Ny-1,1)+c(i,2:Ny-1,1)+2*d(i,2:Ny-1,1)+f(i,2:Ny-
1,1)+a_loss;
    ho(i,2:Ny-1,1)=q_in(i,2:Ny-1)-b_loss;
    switch id.label(i)
        case 1
            dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
            b(i,2:Ny-1,Nz)=ks*dy*dz/dx_val;
            dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
            c(i,2:Ny-1,Nz)=ks*dy*dz/dx_val;
            dx_val=dx(id.label(i));
            d(i,2:Ny-1,Nz)=ks*dx_val*dz/dy;
            e(i,2:Ny-1,Nz)=d(i,2:Ny-1,Nz);
            g(i,2:Ny-1,Nz)=ks*dx_val*dy/dz;
            Nc=id.num(i);
            a(i,2:Ny-1,Nz)=b(i,2:Ny-1,Nz)+c(i,2:Ny-1,Nz)+2*d(i,2:Ny-
1,Nz)+g(i,2:Ny-1,Nz)+h(Nc,2:Ny-1)*dx_val*dy;
            ho(i,2:Ny-1,Nz)=h(Nc,2:Ny-1)*dx_val*dy.*Tf(Nc,2:Ny-1);
        case 2
            dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
            b(i,2:Ny-1,Nz)=ks*dy*dz/dx_val;
            dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
            c(i,2:Ny-1,Nz)=ks*dy*dz/dx_val;
            dx_val=dx(id.label(i));
            d(i,2:Ny-1,Nz)=ks*dx_val*dz/dy;
            e(i,2:Ny-1,Nz)=d(i,2:Ny-1,Nz);
            g(i,2:Ny-1,Nz)=ks*dx_val*dy/dz;
            Nf=id.num(i);
            m1=sqrt(h(Nf,2:Ny-1)*2/ks/dx_val);
            m2=sqrt(h(Nf+1,2:Ny-1)*2/ks/dx_val);
            a(i,2:Ny-1,Nz)=b(i,2:Ny-1,Nz)+c(i,2:Ny-1,Nz)+2*d(i,2:Ny-
1,Nz)+g(i,2:Ny-1,Nz)+sqrt(h(Nf,2:Ny-
1)*2*dy^2*ks*dx_val).*tanh(m1*dc)/2+sqrt(h(Nf+1,2:Ny-
1)*2*dy^2*ks*dx_val).*tanh(m2*dc)/2;
            ho(i,2:Ny-1,Nz)=sqrt(h(Nf,2:Ny-
1)*2*dy^2*ks*dx_val).*tanh(m1*dc)/2.*Tf(Nf,2:Ny-1)+sqrt(h(Nf+1,2:Ny-
1)*2*dy^2*ks*dx_val).*tanh(m2*dc)/2.*Tf(Nf+1,2:Ny-1);
        case 3
            dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
            b(i,2:Ny-1,Nz)=ks*dy*dz/dx_val;
            dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
            c(i,2:Ny-1,Nz)=ks*dy*dz/dx_val;
            dx_val=dx(id.label(i));
            d(i,2:Ny-1,Nz)=ks*dx_val*dz/dy;
            e(i,2:Ny-1,Nz)=d(i,2:Ny-1,Nz);
            g(i,2:Ny-1,Nz)=ks*dx_val*dy/dz;
            Nef=id.num(i);
            m1=sqrt(h(Nef,2:Ny-1)*2/ks/dx_val/mesh.Nx_ef);
            a(i,2:Ny-1,Nz)=b(i,2:Ny-1,Nz)+c(i,2:Ny-1,Nz)+2*d(i,2:Ny-
1,Nz)+g(i,2:Ny-1,Nz)+sqrt(h(Nef,2:Ny-
1)*4*dy^2*ks*dx_val).*tanh(m1*dc)/2/mesh.Nx_ef;
            ho(i,2:Ny-1,Nz)=sqrt(h(Nef,2:Ny-
1)*4*dy^2*ks*dx_val).*tanh(m1*dc)/2/mesh.Nx_ef.*Tf(Nef,2:Ny-1);
    end
end
%left face
b(1,2:Ny-1,2:Nz-1)=ks*dy*dz/dx_ef;
d(1,2:Ny-1,2:Nz-1)=ks*dx_ef*dz/dy;
e(1,2:Ny-1,2:Nz-1)=d(1,2:Ny-1,2:Nz-1);
f(1,2:Ny-1,2:Nz-1)=ks*dx_ef*dy/dz;
g(1,2:Ny-1,2:Nz-1)=f(1,2:Ny-1,2:Nz-1);
```

```
a(1,2:Ny-1,2:Nz-1)=b(1,2:Ny-1,2:Nz-1)+2*d(1,2:Ny-1,2:Nz-1)+2*f(1,2:Ny-
1,2:Nz-1);
%right face
c(Nx,2:Ny-1,2:Nz-1)=ks*dy*dz/dx_ef;
d(Nx,2:Ny-1,2:Nz-1)=ks*dx_ef*dz/dy;
e(Nx,2:Ny-1,2:Nz-1)=d(Nx,2:Ny-1,2:Nz-1);
f(Nx,2:Ny-1,2:Nz-1)=ks*dx_ef*dy/dz;
g(Nx,2:Ny-1,2:Nz-1)=f(Nx,2:Ny-1,2:Nz-1);
a(Nx,2:Ny-1,2:Nz-1)=c(Nx,2:Ny-1,2:Nz-1)+2*d(Nx,2:Ny-1,2:Nz-
1)+2*f(Nx,2:Ny-1,2:Nz-1);
%front and back faces
for i=2:Nx-1
    dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
    b(i,1,2:Nz-1)=ks*dy*dz/dx_val;
    b(i,Ny,2:Nz-1)=ks*dy*dz/dx_val;
    dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
    c(i,1,2:Nz-1)=ks*dy*dz/dx_val;
    c(i,Ny,2:Nz-1)=ks*dy*dz/dx_val;
    dx_val=dx(id.label(i));
    d(i,1,2:Nz-1)=ks*dx_val*dz/dy;
    f(i,1,2:Nz-1)=ks*dx_val*dy/dz;
    g(i,1,2:Nz-1)=f(i,1,2:Nz-1);
    a(i,1,2:Nz-1)=b(i,1,2:Nz-1)+c(i,1,2:Nz-1)+d(i,1,2:Nz-
1)+2*f(i,1,2:Nz-1);
    e(i,Ny,2:Nz-1)=ks*dx_val*dz/dy;
    f(i,Ny,2:Nz-1)=ks*dx_val*dy/dz;
    g(i,Ny,2:Nz-1)=f(i,Ny,2:Nz-1);
    a(i,Ny,2:Nz-1)=b(i,Ny,2:Nz-1)+c(i,Ny,2:Nz-1)+e(i,Ny,2:Nz-
1)+2*f(i,Ny,2:Nz-1);
end
%front left edge
b(1,1,2:Nz-1)=ks*dy*dz/dx_ef;
d(1,1,2:Nz-1)=ks*dx_ef*dz/dy;
f(1,1,2:Nz-1)=ks*dx_ef*dy/dz;
g(1,1,2:Nz-1)=f(1,1,2:Nz-1);
a(1,1,2:Nz-1)=b(1,1,2:Nz-1)+d(1,1,2:Nz-1)+2*f(1,1,2:Nz-1);
%front right edge
c(Nx,1,2:Nz-1)=ks*dy*dz/dx_ef;
d(Nx,1,2:Nz-1)=ks*dx_ef*dz/dy;
f(Nx,1,2:Nz-1)=ks*dx_ef*dy/dz;
g(Nx,1,2:Nz-1)=f(Nx,1,2:Nz-1);
a(Nx,1,2:Nz-1)=c(Nx,1,2:Nz-1)+d(Nx,1,2:Nz-1)+2*f(Nx,1,2:Nz-1);
%back left edge
b(1,Ny,2:Nz-1)=ks*dy*dz/dx_ef;
e(1,Ny,2:Nz-1)=ks*dx_ef*dz/dy;
f(1,Ny,2:Nz-1)=ks*dx_ef*dy/dz;
g(1,Ny,2:Nz-1)=f(1,Ny,2:Nz-1);
a(1,Ny,2:Nz-1)=b(1,Ny,2:Nz-1)+e(1,Ny,2:Nz-1)+2*f(1,Ny,2:Nz-1);
%back right edge
c(Nx,Ny,2:Nz-1)=ks*dy*dz/dx_ef;
e(Nx,Ny,2:Nz-1)=ks*dx_ef*dz/dy;
f(Nx,Ny,2:Nz-1)=ks*dx_ef*dy/dz;
g(Nx,Ny,2:Nz-1)=f(Nx,Ny,2:Nz-1);
a(Nx,Ny,2:Nz-1)=c(Nx,Ny,2:Nz-1)+e(Nx,Ny,2:Nz-1)+2*f(Nx,Ny,2:Nz-1);
%bottom left edge
```

```
b(1,2:Ny-1,1)=ks*dy*dz/dx_ef;
d(1,2:Ny-1,1)=ks*dx_ef*dz/dy;
e(1,2:Ny-1,1)=d(1,2:Ny-1,1);
f(1,2:Ny-1,1)=ks*dx_ef*dy/dz;
a(1,2:Ny-1,1)=b(1,2:Ny-1,1)+2*d(1,2:Ny-1,1)+f(1,2:Ny-1,1)+a_loss;
ho(1,2:Ny-1,1)=q_in(1,2:Ny-1)-b_loss;
%bottom right edge
c(Nx,2:Ny-1,1)=ks*dy*dz/dx_ef;
d(Nx,2:Ny-1,1)=ks*dx_ef*dz/dy;
e(Nx,2:Ny-1,1)=d(Nx,2:Ny-1,1);
f(Nx,2:Ny-1,1)=ks*dx_ef*dy/dz;
a(Nx,2:Ny-1,1)=c(Nx,2:Ny-1,1)+2*d(Nx,2:Ny-1,1)+f(Nx,2:Ny-1,1)+a_loss;
ho(Nx,2:Ny-1,1)=q_in(Nx,2:Ny-1)-b_loss;
%top left edge
b(1,2:Ny-1,Nz)=ks*dy*dz/dx_ef;
d(1,2:Ny-1,Nz)=ks*dx_ef*dz/dy;
e(1,2:Ny-1,Nz)=d(1,2:Ny-1,Nz);
g(1,2:Ny-1,Nz)=ks*dx_ef*dy/dz;
m1=sqrt(h(1,2:Ny-1)*2/ks/dx_ef/mesh.Nx_ef);
a(1,2:Ny-1,Nz)=b(1,2:Ny-1,Nz)+2*d(1,2:Ny-1,Nz)+g(1,2:Ny-
1,Nz)+sqrt(h(1,2:Ny-1)*4*dy^2*ks*dx_ef).*tanh(m1*dc)/2/mesh.Nx_ef;
ho(1,2:Ny-1,Nz)=sqrt(h(1,2:Ny-
1)*4*dy^2*ks*dx_ef).*tanh(m1*dc)/2/mesh.Nx_ef.*Tf(1,2:Ny-1);
%top right edge
c(Nx,2:Ny-1,Nz)=ks*dy*dz/dx_ef;
d(Nx,2:Ny-1,Nz)=ks*dx_ef*dz/dy;
e(Nx,2:Ny-1,Nz)=d(Nx,2:Ny-1,Nz);
g(Nx,2:Ny-1,Nz)=ks*dx_ef*dy/dz;
m1=sqrt(h(N,2:Ny-1)*2/ks/dx_ef/mesh.Nx_ef);
a(Nx,2:Ny-1,Nz)=c(Nx,2:Ny-1,Nz)+2*d(Nx,2:Ny-1,Nz)+g(Nx,2:Ny-
1,Nz)+sqrt(h(N,2:Ny-1)*4*dy^2*ks*dx_ef).*tanh(m1*dc)/2/mesh.Nx_ef;
ho(Nx,2:Ny-1,Nz)=sqrt(h(N,2:Ny-
1)*4*dy^2*ks*dx_ef).*tanh(m1*dc)/2/mesh.Nx_ef.*Tf(N,2:Ny-1);
%front/back top/bottom edges
for i=2:Nx-1
    dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
    b(i,1,1)=ks*dy*dz/dx_val;
    b(i,Ny,1)=ks*dy*dz/dx_val;
    dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
    c(i,1,1)=ks*dy*dz/dx_val;
    c(i,Ny,1)=ks*dy*dz/dx_val;
    dx_val=dx(id.label(i));
    d(i,1,1)=ks*dx_val*dz/dy;
    f(i,1,1)=ks*dx_val*dy/dz;
    a(i,1,1)=b(i,1,1)+c(i,1,1)+d(i,1,1)+f(i,1,1)+a_loss;
    ho(i,1,1)=q_in(i,1)-b_loss;
    e(i,Ny,1)=ks*dx_val*dz/dy;
    f(i,Ny,1)=ks*dx_val*dy/dz;
    a(i,Ny,1)=b(i,Ny,1)+c(i,Ny,1)+e(i,Ny,1)+f(i,Ny,1)+a_loss;
    ho(i,Ny,1)=q_in(i,Ny)-b_loss;
    switch id.label(i)
        case 1
            dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
            b(i,1,Nz)=ks*dy*dz/dx_val;
            dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
            c(i,1,Nz)=ks*dy*dz/dx_val;
            dx_val=dx(id.label(i));
            d(i,1,Nz)=ks*dx_val*dz/dy;
            g(i,1,Nz)=ks*dx_val*dy/dz;
            Nc=id.num(i);
a(i,1,Nz)=b(i,1,Nz)+c(i,1,Nz)+d(i,1,Nz)+g(i,1,Nz)+h(Nc,1)*dx_val*dy;
            ho(i,1,Nz)=h(Nc,1)*dx_val*dy.*Tf(Nc,1);
            dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
            b(i,Ny,Nz)=ks*dy*dz/dx_val;
            dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
            c(i,Ny,Nz)=ks*dy*dz/dx_val;
            dx_val=dx(id.label(i));
            e(i,Ny,Nz)=ks*dx_val*dz/dy;
            g(i,Ny,Nz)=ks*dx_val*dy/dz;
            Nc=id.num(i);
a(i,Ny,Nz)=b(i,Ny,Nz)+c(i,Ny,Nz)+e(i,Ny,Nz)+g(i,Ny,Nz)+h(Nc,Ny)*dx_val*
dy;
            ho(i,Ny,Nz)=h(Nc,Ny)*dx_val*dy.*Tf(Nc,Ny);
        case 2
            dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
            b(i,1,Nz)=ks*dy*dz/dx_val;
            dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
            c(i,1,Nz)=ks*dy*dz/dx_val;
            dx_val=dx(id.label(i));
            d(i,1,Nz)=ks*dx_val*dz/dy;
            g(i,1,Nz)=ks*dx_val*dy/dz;
            Nf=id.num(i);
            m1=sqrt(h(Nf,1)*2/ks/dx_val);
            m2=sqrt(h(Nf+1,1)*2/ks/dx_val);
a(i,1,Nz)=b(i,1,Nz)+c(i,1,Nz)+d(i,1,Nz)+g(i,1,Nz)+sqrt(h(Nf,1)*2*dy^2*k
s*dx_val).*tanh(m1*dc)/2+sqrt(h(Nf+1,1)*2*dy^2*ks*dx_val).*tanh(m2*dc)/
2;
ho(i,1,Nz)=sqrt(h(Nf,1)*2*dy^2*ks*dx_val).*tanh(m1*dc)/2.*Tf(Nf,1)+sqrt
(h(Nf+1,1)*2*dy^2*ks*dx_val).*tanh(m2*dc)/2.*Tf(Nf+1,1);
            dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
            b(i,Ny,Nz)=ks*dy*dz/dx_val;
            dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
            c(i,Ny,Nz)=ks*dy*dz/dx_val;
            dx_val=dx(id.label(i));
            e(i,Ny,Nz)=ks*dx_val*dz/dy;
            g(i,Ny,Nz)=ks*dx_val*dy/dz;
            Nf=id.num(i);
            m1=sqrt(h(Nf,Ny)*2/ks/dx_val);
            m2=sqrt(h(Nf+1,Ny)*2/ks/dx_val);
a(i,Ny,Nz)=b(i,Ny,Nz)+c(i,Ny,Nz)+e(i,Ny,Nz)+g(i,Ny,Nz)+sqrt(h(Nf,Ny)*2*
dy^2*ks*dx_val).*tanh(m1*dc)/2+sqrt(h(Nf+1,Ny)*2*dy^2*ks*dx_val).*tanh(
m2*dc)/2;
ho(i,Ny,Nz)=sqrt(h(Nf,Ny)*2*dy^2*ks*dx_val).*tanh(m1*dc)/2.*Tf(Nf,Ny)+s
qrt(h(Nf+1,Ny)*2*dy^2*ks*dx_val).*tanh(m2*dc)/2.*Tf(Nf+1,Ny);
        case 3
            dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
            b(i,1,Nz)=ks*dy*dz/dx_val;
            dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
            c(i,1,Nz)=ks*dy*dz/dx_val;
            dx_val=dx(id.label(i));
            d(i,1,Nz)=ks*dx_val*dz/dy;
            g(i,1,Nz)=ks*dx_val*dy/dz;
            Nef=id.num(i);
            m1=sqrt(h(Nef,1)*2/ks/dx_val/mesh.Nx_ef);
a(i,1,Nz)=b(i,1,Nz)+c(i,1,Nz)+d(i,1,Nz)+g(i,1,Nz)+sqrt(h(Nef,1)*4*dy^2*
ks*dx_val).*tanh(m1*dc)/2/mesh.Nx_ef;
ho(i,1,Nz)=sqrt(h(Nef,1)*4*dy^2*ks*dx_val).*tanh(m1*dc)/2/mesh.Nx_ef.*T
f(Nef,1);
            dx_val=(dx(id.label(i))+dx(id.label(i+1)))/2;
            b(i,Ny,Nz)=ks*dy*dz/dx_val;
            dx_val=(dx(id.label(i))+dx(id.label(i-1)))/2;
            c(i,Ny,Nz)=ks*dy*dz/dx_val;
            dx_val=dx(id.label(i));
            e(i,Ny,Nz)=ks*dx_val*dz/dy;
            g(i,Ny,Nz)=ks*dx_val*dy/dz;
            Nef=id.num(i);
            m1=sqrt(h(Nef,Ny)*2/ks/dx_val/mesh.Nx_ef);
a(i,Ny,Nz)=b(i,Ny,Nz)+c(i,Ny,Nz)+e(i,Ny,Nz)+g(i,Ny,Nz)+sqrt(h(Nef,Ny)*4
*dy^2*ks*dx_val).*tanh(m1*dc)/2/mesh.Nx_ef;
ho(i,Ny,Nz)=sqrt(h(Nef,Ny)*4*dy^2*ks*dx_val).*tanh(m1*dc)/2/mesh.Nx_ef.
*Tf(Nef,Ny);
    end
end
%front top left corner
b(1,1,Nz)=ks*dy*dz/dx_ef;
d(1,1,Nz)=ks*dx_ef*dz/dy;
g(1,1,Nz)=ks*dx_ef*dy/dz;
m1=sqrt(h(1,1)*2/ks/dx_ef/mesh.Nx_ef);
a(1,1,Nz)=b(1,1,Nz)+d(1,1,Nz)+g(1,1,Nz)+sqrt(h(1,1)*4*dy^2*ks*dx_ef)*ta
nh(m1*dc)/2/mesh.Nx_ef;
ho(1,1,Nz)=sqrt(h(1,1)*4*dy^2*ks*dx_ef)*tanh(m1*dc)/2/mesh.Nx_ef*Tf(1,1
);
%front bottom left corner
b(1,1,1)=ks*dy*dz/dx_ef;
d(1,1,1)=ks*dx_ef*dz/dy;
f(1,1,1)=ks*dx_ef*dy/dz;
a(1,1,1)=b(1,1,1)+d(1,1,1)+f(1,1,1)+a_loss;
ho(1,1,1)=q_in(1,1)-b_loss;
%front top right corner
c(Nx,1,Nz)=ks*dy*dz/dx_ef;
d(Nx,1,Nz)=ks*dx_ef*dz/dy;
g(Nx,1,Nz)=ks*dx_ef*dy/dz;
m1=sqrt(h(N,1)*2/ks/dx_ef/mesh.Nx_ef);
a(Nx,1,Nz)=c(Nx,1,Nz)+d(Nx,1,Nz)+g(Nx,1,Nz)+sqrt(h(N,1)*4*dy^2*ks*dx_ef
)*tanh(m1*dc)/2/mesh.Nx_ef;
ho(Nx,1,Nz)=sqrt(h(N,1)*4*dy^2*ks*dx_ef)*tanh(m1*dc)/2/mesh.Nx_ef*Tf(N,
1);
%front bottom right corner
c(Nx,1,1)=ks*dy*dz/dx_ef;
d(Nx,1,1)=ks*dx_ef*dz/dy;
f(Nx,1,1)=ks*dx_ef*dy/dz;
a(Nx,1,1)=c(Nx,1,1)+d(Nx,1,1)+f(Nx,1,1)+a_loss;
ho(Nx,1,1)=q_in(Nx,1)-b_loss;
%back top left corner
b(1,Ny,Nz)=ks*dy*dz/dx_ef;
```

```
e(1,Ny,Nz)=ks*dx_ef*dz/dy;
g(1,Ny,Nz)=ks*dx_ef*dy/dz;
m1=sqrt(h(1,Ny)*2/ks/dx_ef/mesh.Nx_ef);
a(1,Ny,Nz)=b(1,Ny,Nz)+e(1,Ny,Nz)+g(1,Ny,Nz)+sqrt(h(1,Ny)*4*dy^2*ks*dx_e
f)*tanh(m1*dc)/2/mesh.Nx_ef;
ho(1,Ny,Nz)=sqrt(h(1,Ny)*4*dy^2*ks*dx_ef)*tanh(m1*dc)/2/mesh.Nx_ef*Tf(1
,Ny);
%back bottom left corner
b(1,Ny,1)=ks*dy*dz/dx_ef;
e(1,Ny,1)=ks*dx_ef*dz/dy;
f(1,Ny,1)=ks*dx_ef*dy/dz;
a(1,Ny,1)=b(1,Ny,1)+e(1,Ny,1)+f(1,Ny,1)+a_loss;
ho(1,Ny,1)=q_in(1,Ny)-b_loss;
%back top right corner
c(Nx,Ny,Nz)=ks*dy*dz/dx_ef;
e(Nx,Ny,Nz)=ks*dx_ef*dz/dy;
g(Nx,Ny,Nz)=ks*dx_ef*dy/dz;
m1=sqrt(h(N,Ny)*2/ks/dx_ef/mesh.Nx_ef);
a(Nx,Ny,Nz)=c(Nx,Ny,Nz)+e(Nx,Ny,Nz)+g(Nx,Ny,Nz)+sqrt(h(N,Ny)*4*dy^2*ks*
dx_ef)*tanh(m1*dc)/2/mesh.Nx_ef;
ho(Nx,Ny,Nz)=sqrt(h(N,Ny)*4*dy^2*ks*dx_ef)*tanh(m1*dc)/2/mesh.Nx_ef*Tf(
N,Ny);
%back bottom right corner
c(Nx,Ny,1)=ks*dy*dz/dx_ef;
e(Nx,Ny,1)=ks*dx_ef*dz/dy;
f(Nx,Ny,1)=ks*dx_ef*dy/dz;
a(Nx,Ny,1)=c(Nx,Ny,1)+e(Nx,Ny,1)+f(Nx,Ny,1)+a_loss;
ho(Nx,Ny,1)=q_in(Nx,Ny)-b_loss;
end
```

Function 33. conduction.m

```
function [T]=conduction(dims,matp,mesh,T,Tf,h,q_in)
Nx=mesh.Nx_c+mesh.Nx_f+mesh.Nx_ef;
Ny=mesh.Ny;
Nz=mesh.Nz;
[a,b,c,d,e,f,g,ho]=condbasemat(mesh,dims,matp,h,q_in,Tf);
for n=1:10
    for i=Nx:-1:1
        a1=reshape(a(i,:,:),Ny,Nz);
        b1=reshape(b(i,:,:),Ny,Nz);
        c1=reshape(c(i,:,:),Ny,Nz);
        d1=reshape(d(i,:,:),Ny,Nz);
        e1=reshape(e(i,:,:),Ny,Nz);
        f1=reshape(f(i,:,:),Ny,Nz);
        g1=reshape(g(i,:,:),Ny,Nz);
        if i==1
h1=reshape(ho(i,:,:),Ny,Nz)+b1.*reshape(T(i+1,:,1:Nz),Ny,Nz);
        elseif i==Nx
            h1=reshape(ho(i,:,:),Ny,Nz)+c1.*reshape(T(i-
1,:,1:Nz),Ny,Nz);
        else
h1=reshape(ho(i,:,:),Ny,Nz)+b1.*reshape(T(i+1,:,1:Nz),Ny,Nz)+c1.*reshap
e(T(i-1,:,1:Nz),Ny,Nz);
        end
        T1=reshape(T(i,:,1:Nz),Ny,Nz);
        T(i,:,1:Nz)=tdmaline(a1,f1,g1,d1,e1,h1,T1);
    end
    for j=Ny:-1:1
        a1=reshape(a(:,j,:),Nx,Nz);
        b1=reshape(b(:,j,:),Nx,Nz);
        c1=reshape(c(:,j,:),Nx,Nz);
        d1=reshape(d(:,j,:),Nx,Nz);
        e1=reshape(e(:,j,:),Nx,Nz);
        f1=reshape(f(:,j,:),Nx,Nz);
        g1=reshape(g(:,j,:),Nx,Nz);
        if j==1
h1=reshape(ho(:,j,:),Nx,Nz)+d1.*reshape(T(:,j+1,1:Nz),Nx,Nz);
        elseif j==Ny
            h1=reshape(ho(:,j,:),Nx,Nz)+e1.*reshape(T(:,j-
1,1:Nz),Nx,Nz);
        else
h1=reshape(ho(:,j,:),Nx,Nz)+d1.*reshape(T(:,j+1,1:Nz),Nx,Nz)+e1.*reshap
e(T(:,j-1,1:Nz),Nx,Nz);
        end
        T1=reshape(T(:,j,1:Nz),Nx,Nz);
        T(:,j,1:Nz)=tdmaline(a1,f1,g1,b1,c1,h1,T1);
    end
    for k=Nz:-1:1
        a1=reshape(a(:,:,k),Nx,Ny);
        b1=reshape(b(:,:,k),Nx,Ny);
        c1=reshape(c(:,:,k),Nx,Ny);
        d1=reshape(d(:,:,k),Nx,Ny);
        e1=reshape(e(:,:,k),Nx,Ny);
        f1=reshape(f(:,:,k),Nx,Ny);
        g1=reshape(g(:,:,k),Nx,Ny);
        if k==1
            h1=reshape(ho(:,:,k),Nx,Ny)+f1.*reshape(T(:,:,k+1),Nx,Ny);
        elseif k==Nz
            h1=reshape(ho(:,:,k),Nx,Ny)+g1.*reshape(T(:,:,k-1),Nx,Ny);
        else
h1=reshape(ho(:,:,k),Nx,Ny)+f1.*reshape(T(:,:,k+1),Nx,Ny)+g1.*reshape(T
(:,:,k-1),Nx,Ny);
        end
        T1=reshape(T(:,:,k),Nx,Ny);
        T(:,:,k)=tdmaline(a1,d1,e1,b1,c1,h1,T1);
    end
    for i=1:Nx
        a1=reshape(a(i,:,:),Ny,Nz);
        b1=reshape(b(i,:,:),Ny,Nz);
        c1=reshape(c(i,:,:),Ny,Nz);
        d1=reshape(d(i,:,:),Ny,Nz);
        e1=reshape(e(i,:,:),Ny,Nz);
        f1=reshape(f(i,:,:),Ny,Nz);
        g1=reshape(g(i,:,:),Ny,Nz);
        if i==1
h1=reshape(ho(i,:,:),Ny,Nz)+b1.*reshape(T(i+1,:,1:Nz),Ny,Nz);
        elseif i==Nx
            h1=reshape(ho(i,:,:),Ny,Nz)+c1.*reshape(T(i-
1,:,1:Nz),Ny,Nz);
        else
h1=reshape(ho(i,:,:),Ny,Nz)+b1.*reshape(T(i+1,:,1:Nz),Ny,Nz)+c1.*reshap
e(T(i-1,:,1:Nz),Ny,Nz);
        end
        T1=reshape(T(i,:,1:Nz),Ny,Nz);
        T(i,:,1:Nz)=tdmaline(a1,f1,g1,d1,e1,h1,T1);
    end
    for j=1:Ny
        a1=reshape(a(:,j,:),Nx,Nz);
        b1=reshape(b(:,j,:),Nx,Nz);
        c1=reshape(c(:,j,:),Nx,Nz);
        d1=reshape(d(:,j,:),Nx,Nz);
        e1=reshape(e(:,j,:),Nx,Nz);
        f1=reshape(f(:,j,:),Nx,Nz);
        g1=reshape(g(:,j,:),Nx,Nz);
        if j==1
h1=reshape(ho(:,j,:),Nx,Nz)+d1.*reshape(T(:,j+1,1:Nz),Nx,Nz);
        elseif j==Ny
            h1=reshape(ho(:,j,:),Nx,Nz)+e1.*reshape(T(:,j-
1,1:Nz),Nx,Nz);
        else
h1=reshape(ho(:,j,:),Nx,Nz)+d1.*reshape(T(:,j+1,1:Nz),Nx,Nz)+e1.*reshap
e(T(:,j-1,1:Nz),Nx,Nz);
        end
        T1=reshape(T(:,j,1:Nz),Nx,Nz);
        T(:,j,1:Nz)=tdmaline(a1,f1,g1,b1,c1,h1,T1);
    end
    for k=1:Nz
        a1=reshape(a(:,:,k),Nx,Ny);
        b1=reshape(b(:,:,k),Nx,Ny);
        c1=reshape(c(:,:,k),Nx,Ny);
        d1=reshape(d(:,:,k),Nx,Ny);
        e1=reshape(e(:,:,k),Nx,Ny);
        f1=reshape(f(:,:,k),Nx,Ny);
        g1=reshape(g(:,:,k),Nx,Ny);
        if k==1
            h1=reshape(ho(:,:,k),Nx,Ny)+f1.*reshape(T(:,:,k+1),Nx,Ny);
         elseif k==Nz
            h1=reshape(ho(:,:,k),Nx,Ny)+g1.*reshape(T(:,:,k-1),Nx,Ny);
        else
h1=reshape(ho(:,:,k),Nx,Ny)+f1.*reshape(T(:,:,k+1),Nx,Ny)+g1.*reshape(T
(:,:,k-1),Nx,Ny);
        end
        T1=reshape(T(:,:,k),Nx,Ny);
        T(:,:,k)=tdmaline(a1,d1,e1,b1,c1,h1,T1);
    end

end
end
```

Function 34. convection2.m

```
function
[Tf,h,q_w,xe]=convection2(dims,matp,mesh,flow,T,Tf,q_w,DP,xe,ws)
Nxef=mesh.Nx_ef;
Nxf=mesh.Nx_f/(dims.N-1);
Nx=mesh.Nx_c+mesh.Nx_f+mesh.Nx_ef;
Ny=mesh.Ny;
Nz=mesh.Nz;
N=dims.N;
dx=mesh.dx_c;
dxf=mesh.dx_f;
dxef=mesh.dx_ef;
dy=mesh.dy;
ks=matp.k_Si;
d=dims.d;
w=dims.w;
L=dims.L;
%find Tf everywhere
% fmald=ones(N,1);
fmald=ones(N,1)+.0625;
fmald(15:21)=.75;
Tsat=matp.Tsata./(matp.Tsatb-matp.Tsatc*log10(flow.P_in-DP))-273;
for i=1:N
    for j=1:Ny
Tf(i,j)=flow.T_in+sum(q_w(i,1:j))*(2*d+w)*dy/(flow.m_dot*fmald(i)*densi
ty(matp,Tf(i,j))/N/60/1e6)/specheat(matp,Tf(i,j));
        if Tf(i,j)>Tsat(i,j)
            Tf(i,j)=Tsat(i,j);
        end
    end
end
%find h everywhere
[h]=heattranscoeff(dims,matp,mesh,flow,T,Tf,Tsat,q_w,xe,fmald,ws);
%find q_w [W] everywhere
wq=zeros(N-1,Ny)+0.5;
assignin('base','wq',wq);
q_w=zeros(N,Ny);
Af=2*d*L;
Aw=N*(2*d+w)*L;
for j=1:Ny
    m1=sqrt(2*h(1,j)/ks/dxef/Nxef);
    etaf=tanh(m1*d)/m1/d;
    eta=1-N*Af/Aw*(1-etaf);
    q_w(1,j)=q_w(1,j)+eta*d*dy*h(1,j)*(mean(T(1:Nxef/2,j,Nz))-Tf(1,j));

    for i=1:(N-1)
q_w(i,j)=q_w(i,j)+h(i,j)*dx*dy*sum(T(mesh.id.num==i&mesh.id.label==1,j,
Nz)-Tf(i,j));
        m1=sqrt((h(i,j)+h(i+1,j))/ks/dxf/Nxf);
        etaf=tanh(m1*d)/m1/d;
        eta=1-N*Af/Aw*(1-etaf);
q_w(i,j)=q_w(i,j)+eta*2*d*dy*h(i,j)*wq(i,j)*(T(mesh.id.num==i&mesh.id.l
abel==2,j,Nz)-Tf(i,j));
```

```
        q_w(i+1,j)=q_w(i+1,j)+eta*2*d*dy*h(i+1,j)*(1-
wq(i,j))*(T(mesh.id.num==i&mesh.id.label==2,j,Nz)-Tf(i+1,j));
    end
q_w(N,j)=q_w(N,j)+h(N,j)*dx*dy*sum(T(mesh.id.num==N&mesh.id.label==1,j,
Nz)-Tf(N,j));
    m1=sqrt(2*h(N,j)/ks/dxef/Nxef);
    etaf=tanh(m1*d)/m1/d;
    eta=1-N*Af/Aw*(1-etaf);
    q_w(N,j)=q_w(N,j)+eta*d*dy*h(N,j)*(mean(T(Nx-Nxef/2+1:Nx,j,Nz))-
Tf(N,j));
end
%find xe everywhere
for i=1:N
    for j=1:Ny
xe(i,j)=1/matp.h_fg*(sum(q_w(i,1:j))/(flow.m_dot*fmald(i)/N*density(mat
p,Tf(i,j))/1e6/60)-specheat(matp,Tf(i,j))*(Tsat(i,j)-flow.T_in));
    end
end
%get q_w [W/m2]
q_w=q_w/dy/(w+2*d);
end
function rho=density(matp,Tf)
rho=matp.rho_fa+matp.rho_fb*Tf;
end
function cp=specheat(matp,Tf)
cp=matp.c_pfa+matp.c_pfb*Tf;
end
```

Function 35. discretize.m

```
function [mesh]=discretize(dims)
n1=6;
mesh.Nz=round(n1*(dims.t-dims.d)/dims.t);
mesh.dz=(dims.t-dims.d)/mesh.Nz;
Nx=round(dims.W/dims.t*0.95*n1);
mesh.Nx_c=round(dims.N*dims.w/dims.W*Nx);
mesh.Nx_f=round((dims.N-1)*dims.w_f/dims.W*Nx);
mesh.Nx_ef=Nx-mesh.Nx_c-mesh.Nx_f;
while mod(mesh.Nx_c+mesh.Nx_f+mesh.Nx_ef,5)
    mesh.Nx_ef=mesh.Nx_ef+1;
    while mod(mesh.Nx_c,dims.N)
        mesh.Nx_c=mesh.Nx_c+1;
    end
    while mod(mesh.Nx_f,dims.N-1)
        mesh.Nx_f=mesh.Nx_f+1;
    end
    while mod(mesh.Nx_ef,2)
        mesh.Nx_ef=mesh.Nx_ef+1;
    end
end
mesh.dx_c=dims.N*dims.w/mesh.Nx_c;
mesh.dx_f=(dims.W-dims.N*dims.w-2*dims.w_ef)/(mesh.Nx_f);
mesh.dx_ef=2*dims.w_ef/mesh.Nx_ef;
mesh.Ny=mesh.Nx_c+mesh.Nx_f+mesh.Nx_ef;
mesh.dy=dims.L/mesh.Ny;
end
```

Function 36. genmesh.m

```
function [mesh]=genmesh(dims,mesh)
Nx_f=mesh.Nx_f;
Nx_ef=mesh.Nx_ef;
Nx_c=mesh.Nx_c;
Nx=Nx_f+Nx_ef+Nx_c;
N=dims.N;
id.num=zeros(Nx,1,'uint8');
id.num(1:Nx_ef/2)=1;
id.num(Nx-Nx_ef/2:end)=N;
id.label=zeros(Nx,1,'uint8');
id.label(1:Nx_ef/2)=3;
id.label(Nx-Nx_ef/2:end)=3;
for i=1:dims.N
    x1=Nx_ef/2+1+(i-1)*(Nx_c/N+Nx_f/(N-1));
    x2=Nx_ef/2+i*Nx_c/N+(i-1)*Nx_f/(N-1);
    id.num(x1:x2)=i;
    id.label(x1:x2)=1;
end
for i=1:dims.N-1
    x1=Nx_ef/2+1+i*Nx_c/N+(i-1)*Nx_f/(N-1);
    x2=Nx_ef/2+i*(Nx_c/N+Nx_f/(N-1));
    id.num(x1:x2)=i;
    id.label(x1:x2)=2;
end
mesh.id=id;
end
```

Function 37. gui_input2.m

```
function [out]=gui_input2
f=figure('visible','off','position',[360,400,450,385],'name','Input
GUI');
%initialize variables in case user doesn't select them
out.dims.L=0.01267;
out.dims.W=0.012668;
out.dims.t=0.000648;
out.dims.w=0.000239114;
out.dims.w_f=0.0001108;
out.dims.d=0.00037144;
out.dims.N=35;
out.flow.P_in=101000;
out.flow.T_in=90.5;
out.flow.m_dot=104;
out.power=33;
out.Phi=1;
out.heatsink='Silicon';
out.fluid='FC-77';
out.heating_case='Case 1a';
out.getdata=0;
out.filename=' ';
out.heaters=zeros(5,5);
ah1=axes('parent',f,'units','pixels','position',[10 200 150 150]);
imshow('microchannel1.tif')
ah2=axes('parent',f,'units','pixels','position',[240 295 150 50]);
imshow('microchannel3.png')
Ltag=uicontrol(f,'style','text','string','L
[mm]','units','pixels','position',[60 365 50 15]);
Lh=uicontrol(f,'style','edit','string','12.67','units','pixels','positi
on',[60 345 50 20],'callback',@Lh_callback);
Wtag=uicontrol(f,'style','text','string','W
[mm]','units','pixels','position',[160 280 50 15]);
Wh=uicontrol(f,'style','edit','string','12.668','units','pixels','posit
ion',[160 260 50 20],'callback',@Wh_callback);
ttag=uicontrol(f,'style','text','string','t
[um]','units','pixels','position',[195 325 50 15]);
th=uicontrol(f,'style','edit','string','648','units','pixels','position
',[195 305 50 20],'callback',@th_callback);
wtag=uicontrol(f,'style','text','string','w
[um]','units','pixels','position',[240 365 50 15]);
wh=uicontrol(f,'style','edit','string','239.114','units','pixels','posi
tion',[240 345 50 20],'callback',@wh_callback);
wftag=uicontrol(f,'style','text','string','wf
[um]','units','pixels','position',[290 365 50 15]);
wfh=uicontrol(f,'style','edit','string','110.8','units','pixels','posit
ion',[290 345 50 20],'callback',@wfh_callback);
dtag=uicontrol(f,'style','text','string','d
[um]','units','pixels','position',[385 330 50 15]);
dh=uicontrol(f,'style','edit','string','371.44','units','pixels','posit
ion',[385 310 50 20],'callback',@dh_callback);
Ntag=uicontrol(f,'style','text','string','# of
channels','units','pixels','position',[250 262 100 15]);
```

```
Nh=uicontrol(f,'style','edit','string','35','units','pixels','position'
,[350 260 50 20],'callback',@Nh_callback);
Ptag=uicontrol(f,'style','text','string','P in
[kPa]','units','pixels','position',[250 242 100 15]);
Ph=uicontrol(f,'style','edit','string','101','units','pixels','position
',[350 240 50 20],'callback',@Ph_callback);
Ttag=uicontrol(f,'style','text','string','T in
[C]','units','pixels','position',[250 222 100 15]);
Th=uicontrol(f,'style','edit','string','90.5','units','pixels','positio
n',[350 220 50 20],'callback',@Th_callback);
Vtag=uicontrol(f,'style','text','string','V in
[mL/min]','units','pixels','position',[250 202 100 15]);
Vh=uicontrol(f,'style','edit','string','104','units','pixels','position
',[350 200 50 20],'callback',@Vh_callback);
hspmh=uicontrol(f,'style','popupmenu','string',{'Silicon','Copper'},'va
lue',1,'position',[20 180 60 20],'callback',@hspmh_callback);
fpmh=uicontrol(f,'style','popupmenu','string',{'FC-77','HFE
7100'},'value',1,'position',[80 180 60 20],'callback',@fpmh_callback);
heattag=uicontrol(f,'style','text','string','Heating
Diagram','units','pixels','position',[20 140 120 15]);
ah3=axes('parent',f,'units','pixels','position',[20 20 120 115]);
imshow('heaters.png');
h01=uicontrol(f,'style','checkbox','string','','value',0,'position',[35
110 15 15],'callback',@h01_callback);
h02=uicontrol(f,'style','checkbox','string','','value',0,'position',[35
90 15 15],'callback',@h02_callback);
h03=uicontrol(f,'style','checkbox','string','','value',0,'position',[35
70 15 15],'callback',@h03_callback);
h04=uicontrol(f,'style','checkbox','string','','value',0,'position',[35
50 15 15],'callback',@h04_callback);
h05=uicontrol(f,'style','checkbox','string','','value',0,'position',[35
30 15 15],'callback',@h05_callback);
h06=uicontrol(f,'style','checkbox','string','','value',0,'position',[55
110 15 15],'callback',@h06_callback);
h07=uicontrol(f,'style','checkbox','string','','value',0,'position',[55
90 15 15],'callback',@h07_callback);
h08=uicontrol(f,'style','checkbox','string','','value',0,'position',[55
70 15 15],'callback',@h08_callback);
h09=uicontrol(f,'style','checkbox','string','','value',0,'position',[55
50 15 15],'callback',@h09_callback);
h10=uicontrol(f,'style','checkbox','string','','value',0,'position',[55
30 15 15],'callback',@h10_callback);
h11=uicontrol(f,'style','checkbox','string','','value',1,'position',[75
110 15 15],'callback',@h11_callback);
h12=uicontrol(f,'style','checkbox','string','','value',1,'position',[75
90 15 15],'callback',@h12_callback);
h13=uicontrol(f,'style','checkbox','string','','value',1,'position',[75
70 15 15],'callback',@h13_callback);
h14=uicontrol(f,'style','checkbox','string','','value',1,'position',[75
50 15 15],'callback',@h14_callback);
h15=uicontrol(f,'style','checkbox','string','','value',1,'position',[75
30 15 15],'callback',@h15_callback);
h16=uicontrol(f,'style','checkbox','string','','value',0,'position',[95
110 15 15],'callback',@h16_callback);
```

```
h17=uicontrol(f,'style','checkbox','string','','value',0,'position',[95
90 15 15],'callback',@h17_callback);
h18=uicontrol(f,'style','checkbox','string','','value',0,'position',[95
70 15 15],'callback',@h18_callback);
h19=uicontrol(f,'style','checkbox','string','','value',0,'position',[95
50 15 15],'callback',@h19_callback);
h20=uicontrol(f,'style','checkbox','string','','value',0,'position',[95
30 15 15],'callback',@h20_callback);
h21=uicontrol(f,'style','checkbox','string','','value',0,'position',[11
5 110 15 15],'callback',@h21_callback);
h22=uicontrol(f,'style','checkbox','string','','value',0,'position',[11
5 90 15 15],'callback',@h22_callback);
h23=uicontrol(f,'style','checkbox','string','','value',0,'position',[11
5 70 15 15],'callback',@h23_callback);
h24=uicontrol(f,'style','checkbox','string','','value',0,'position',[11
5 50 15 15],'callback',@h24_callback);
h25=uicontrol(f,'style','checkbox','string','','value',0,'position',[11
5 30 15 15],'callback',@h25_callback);
heatpmh=uicontrol(f,'style','popupmenu','string',{'Case 1a','Case
1b','Case 1c','Case 1d','Case 2a','Case
2b','custom'},'value',1,'position',[250 140 120
20],'callback',@heatpmh_callback);
powtag=uicontrol(f,'style','text','string','Q in
[W]','units','pixels','position',[250 122 60 15]);
powh=uicontrol(f,'style','edit','string','33','units','pixels','positio
n',[310 120 60 20],'callback',@powh_callback);
phitag=uicontrol(f,'style','text','string','Phi','units','pixels','posi
tion',[250 102 60 15]);
phih=uicontrol(f,'style','edit','string','1','units','pixels','position
',[310 100 60 20],'callback',@phih_callback);
datah=uicontrol(f,'style','checkbox','string','Use saved
data?','value',0,'position',[250 75 150 15],'callback',@datah_callback);
datatx=uicontrol(f,'style','edit','string','results.mat','horizontalali
gnment','left','position',[250 55 150 20],'callback',@datatx_callback);
goh=uicontrol(f,'style','pushbutton','string','GO!','position',[285 5
60 40],'callback',@goh_callback);
myhandles=guihandles(f);
set(f,'visible','on');
uiwait(f);
    function Lh_callback(hObject,~,~)
        out.dims.L=str2double(get(hObject,'string'))/1000;
        guidata(hObject,myhandles);
    end
    function Wh_callback(hObject,~,~)
        out.dims.W=str2double(get(hObject,'string'))/1000;
        guidata(hObject,myhandles);
    end
    function th_callback(hObject,~,~)
        out.dims.t=str2double(get(hObject,'string'))/1e6;
        guidata(hObject,myhandles);
    end
    function wh_callback(hObject,~,~)
        out.dims.w=str2double(get(hObject,'string'))/1e6;
        guidata(hObject,myhandles);
    end
```

```
function wfh_callback(hObject,~,~)
    out.dims.w_f=str2double(get(hObject,'string'))/1e6;
    guidata(hObject,myhandles);
end
function dh_callback(hObject,~,~)
    out.dims.d=str2double(get(hObject,'string'))/1e6;
    guidata(hObject,myhandles);
end
function Nh_callback(hObject,~,~)
    out.dims.N=str2double(get(hObject,'string'));
    guidata(hObject,myhandles);
end
function Ph_callback(hObject,~,~)
    out.flow.P_in=str2double(get(hObject,'string'))*1000;
    guidata(hObject,myhandles);
end
function Th_callback(hObject,~,~)
    out.flow.T_in=str2double(get(hObject,'string'));
    guidata(hObject,myhandles);
end
function Vh_callback(hObject,~,~)
    out.flow.m_dot=str2double(get(hObject,'string'));
    guidata(hObject,myhandles);
end
function powh_callback(hObject,~,~)
    out.power=str2double(get(hObject,'string'));
    guidata(hObject,myhandles);
end
function phih_callback(hObject,~,~)
    out.Phi=str2double(get(hObject,'string'));
    guidata(hObject,myhandles);
end
function hspmh_callback(hObject,~,~)
    val=get(hObject,'value');
    strlist=get(hObject,'string');
    out.heatsink=strlist{val};
    guidata(hObject,myhandles);
end
function fpmh_callback(hObject,~,~)
    val=get(hObject,'value');
    strlist=get(hObject,'string');
    out.fluid=strlist{val};
    guidata(hObject,myhandles);
end
function heatpmh_callback(hObject,~,~)
    val=get(hObject,'value');
    strlist=get(hObject,'string');
    out.heating_case=strlist{val};
    guidata(hObject,myhandles);
    reseth();
    switch strlist{val}
        case 'Case 1a'
            set(h11,'value',1);
            set(h12,'value',1);
            set(h13,'value',1);
```

```
                set(h14,'value',1);
                set(h15,'value',1);
            case 'Case 1b'
                set(h03,'value',1);
                set(h08,'value',1);
                set(h13,'value',1);
                set(h18,'value',1);
                set(h23,'value',1);
            case 'Case 1c'
                set(h01,'value',1);
                set(h02,'value',1);
                set(h03,'value',1);
                set(h04,'value',1);
                set(h05,'value',1);
            case 'Case 1d'
                set(h01,'value',1);
                set(h02,'value',1);
                set(h03,'value',1);
                set(h04,'value',1);
                set(h05,'value',1);
                set(h21,'value',1);
                set(h22,'value',1);
                set(h23,'value',1);
                set(h24,'value',1);
                set(h25,'value',1);
            case 'Case 2a'
                set(h11,'value',1);
                set(h12,'value',1);
                set(h13,'value',1);
                set(h14,'value',1);
                set(h15,'value',1);
            case 'Case 2b'
                set(h03,'value',1);
                set(h08,'value',1);
                set(h13,'value',1);
                set(h18,'value',1);
                set(h23,'value',1);
            otherwise
        end
    end
    function datah_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.getdata=1;
        else
            out.getdata=0;
        end
    end
    function datatx_callback(hObject,~,~)
        out.filename=get(hObject,'string');
    end
    function h01_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(1,1)=1;
        else
            out.heaters(1,1)=0;
        end
    end
    function h02_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(2,1)=1;
        else
            out.heaters(2,1)=0;
        end
    end
    function h03_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(3,1)=1;
        else
            out.heaters(3,1)=0;
        end
    end
    function h04_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(4,1)=1;
        else
            out.heaters(4,1)=0;
        end
    end
    function h05_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(5,1)=1;
        else
            out.heaters(5,1)=0;
        end
    end
    function h06_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(1,2)=1;
        else
            out.heaters(1,2)=0;
        end
    end
    function h07_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(2,2)=1;
        else
            out.heaters(2,2)=0;
        end
    end
    function h08_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(3,2)=1;
        else
            out.heaters(3,2)=0;
        end
    end
    function h09_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(4,2)=1;
        else
```

```
            out.heaters(4,2)=0;
        end
    end
    function h10_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(5,2)=1;
        else
            out.heaters(5,2)=0;
        end
    end
    function h11_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(1,3)=1;
        else
            out.heaters(1,3)=0;
        end
    end
    function h12_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(2,3)=1;
        else
            out.heaters(2,3)=0;
        end
    end
    function h13_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(3,3)=1;
        else
            out.heaters(3,3)=0;
        end
    end
    function h14_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(4,3)=1;
        else
            out.heaters(4,3)=0;
        end
    end
    function h15_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(5,3)=1;
        else
            out.heaters(5,3)=0;
        end
    end
    function h16_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(1,4)=1;
        else
            out.heaters(1,4)=0;
        end
    end
    function h17_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(2,4)=1;
        else
            out.heaters(2,4)=0;
        end
    end
    function h18_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(3,4)=1;
        else
            out.heaters(3,4)=0;
        end
    end
    function h19_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(4,4)=1;
        else
            out.heaters(4,4)=0;
        end
    end
    function h20_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(5,4)=1;
        else
            out.heaters(5,4)=0;
        end
    end
    function h21_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(1,5)=1;
        else
            out.heaters(1,5)=0;
        end
    end
    function h22_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(2,5)=1;
        else
            out.heaters(2,5)=0;
        end
    end
    function h23_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(3,5)=1;
        else
            out.heaters(3,5)=0;
        end
    end
    function h24_callback(hObject,~,~)
        if get(hObject,'value')==1
            out.heaters(4,5)=1;
        else
            out.heaters(4,5)=0;
        end
    end
    function h25_callback(hObject,~,~)
        if get(hObject,'value')==1
```

```
            out.heaters(5,5)=1;
        else
            out.heaters(5,5)=0;
        end
    end
    function reseth()
        set(h01,'value',0);
        set(h02,'value',0);
        set(h03,'value',0);
        set(h04,'value',0);
        set(h05,'value',0);
        set(h06,'value',0);
        set(h07,'value',0);
        set(h08,'value',0);
        set(h09,'value',0);
        set(h10,'value',0);
        set(h11,'value',0);
        set(h12,'value',0);
        set(h13,'value',0);
        set(h14,'value',0);
        set(h15,'value',0);
        set(h16,'value',0);
        set(h17,'value',0);
        set(h18,'value',0);
        set(h19,'value',0);
        set(h20,'value',0);
        set(h21,'value',0);
        set(h22,'value',0);
        set(h23,'value',0);
        set(h24,'value',0);
        set(h25,'value',0);
    end
    function goh_callback(~,~,~)
        uiresume(f);
        display Calculating
        close(f)
    end
end
```

Function 38. gui_output.m

```
function gui_output
f=figure('visible','off','position',[360,300,600,500],'name','Output
GUI');
filename=' ';
flow=evalin('base','flow');
heating_case=flow.heating_case;
quantity='T [C]';
position='bottom';
T=evalin('base','T');
Tf=evalin('base','Tf');
h=evalin('base','h');
q_w=evalin('base','q_w');
xe=evalin('base','xe');
DP=evalin('base','DP');
DPsp=evalin('base','DPsp');
DPtp=evalin('base','DPtp');
mesh=evalin('base','mesh');
id=mesh.id;
ah=axes('parent',f,'units','pixels','position',[100 150 370 320]);
quanttag=uicontrol(f,'style','text','string','Quantity:','position',[50
0 450 75 15]);
quanth=uicontrol(f,'style','popupmenu','string',{'T [C]','h
[W/m2K]','q_wall [W/cm2]','x_e [-]','DP
[Pa]'},'value',1,'position',[500 430 75
20],'callback',@quanth_callback);
postag=uicontrol(f,'style','text','string','Location:','position',[500
410 75 15]);
posh=uicontrol(f,'style','popupmenu','string',{'bottom','fluid','fins',
'x section fn','x section ch'},'value',1,'position',[500 390 75
20],'callback',@posh_callback);
plotb=uicontrol(f,'style','pushbutton','string','Plot','position',[510
340 55 30],'callback',@plotb_callback);
savetag=uicontrol(f,'style','text','string','Save results
to:','horizontalalignment','left','units','pixels','position',[20 40
100 15]);
saveh=uicontrol(f,'style','edit','string','results.mat','horizontalalig
nment','left','units','pixels','position',[20 20 300
20],'callback',@saveh_callback);
saveb=uicontrol(f,'style','pushbutton','string','Save','position',[320
20 50 20],'callback',@saveb_callback);
set(f,'toolbar','figure','visible','on');
    function quanth_callback(hObject,~,~)
        val=get(hObject,'value');
        strlist=get(hObject,'string');
        quantity=strlist{val};
    end
    function posh_callback(hObject,~,~)
        val=get(hObject,'value');
        strlist=get(hObject,'string');
        position=strlist{val};
    end
    function plotb_callback(~,~,~)
        switch quantity
```

```
            case 'T [C]'
                switch position
                    case 'bottom'
                        s=size(T);
                        x=0:1/(s(1)-1):1;
                        surf(x,x,T(:,:,1),'edgecolor','none')
                        xlabel('Normalized Flow Length')
                        ylabel('Normalized Width')
                        zlabel('T_b [C]')
                        title(heating_case)
                    case 'fluid'
                        s=size(Tf);
                        x=0:1/(s(2)-1):1;
                        y=0:1/(s(1)-1):1;
                        surf(x,y,Tf,'edgecolor','none')
                        xlabel('Normalized Flow Length')
                        ylabel('Normalized Width')
                        zlabel('T_f [C]')
                        title(heating_case)
                    case 'fins'
                        s=size(T);
                        x=0:1/(s(1)-1):1;
                        surf(x,x,T(:,:,mesh.Nz),'edgecolor','none')
                        xlabel('Normalized Flow Length')
                        ylabel('Normalized Width')
                        zlabel('T_{fin} [C]')
                        title(heating_case)
                    case 'x section fn'
                        [row,~]=find(id.label==2);
                        row2=unique(row);
                        s=size(T);
                        Tfin=T(row2,floor(s(2)/2),:);
                        s=size(Tfin);
                        x=0:1/(s(1)-1):1;
                        z=0:1/(s(3)-1):1;
surf(z,x,reshape(Tfin,s(1),s(3)),'edgecolor','none')
                        xlabel('Normalized Height')
                        ylabel('Normalized Width')
                        zlabel('T [C]')
                        title(heating_case)
                    case 'x section ch'
                        [row,~]=find(id.label==1);
                        row2=unique(row);
                        s=size(T);
                        Tch=T(row2,floor(s(2)/2),:);
                        s=size(Tch);
                        x=0:1/(s(1)-1):1;
                        z=0:1/(s(3)-1):1;
surf(z,x,reshape(Tch,s(1),s(3)),'edgecolor','none')
                        xlabel('Normalized Height')
                        ylabel('Normalized Width')
                        zlabel('T [C]')
                        title(heating_case)
                end
            case 'h [W/m2K]'
```

```
                s=size(h);
                x=1:1:s(1);
                y=0:1/(s(2)-1):1;
                surf(y,x,h,'edgecolor','none')
                xlabel('Normalized Flow Length')
                ylabel('Channel')
                zlabel('h [W/m^2K]')
                title(heating_case)
            case 'q_wall [W/cm2]'
                s=size(q_w);
                x=1:1:s(1);
                y=0:1/(s(2)-1):1;
                surf(y,x,q_w/10000,'edgecolor','none')
                xlabel('Normalized Flow Length')
                ylabel('Channel')
                zlabel('q_w [W/cm^2]')
                title(heating_case)
            case 'x_e [-]'
                s=size(xe);
                x=1:1:s(1);
                y=0:1/(s(2)-1):1;
                surf(y,x,xe,'edgecolor','none')
                xlabel('Normalized Flow Length')
                ylabel('Channel')
                zlabel('x_e [-]')
                title(heating_case)
            case 'DP [Pa]'
                s=size(DP);
                x=1:1:s(1);
                y=0:1/(s(2)-1):1;
                surf(y,x,DP,'edgecolor','none')
                xlabel('Normalized Flow Length')
                ylabel('Channel')
                zlabel('DP [Pa]')
                title(heating_case)
        end
    end
    function saveh_callback(hObject,~,~)
        filename=get(hObject,'string');
    end
    function saveb_callback(~,~,~)
        evalin('base',['save(''', filename ''')']);
    end
end
```

Function 39. heattranscoeff.m

```
function
[h]=heattranscoeff(dims,matp,mesh,flow,T,Tf,Tsat,q_w,xe,fmald,ws)
N=dims.N;
Ny=mesh.Ny;
Nz=mesh.Nz;
dy=mesh.dy;
xonb=zeros(N,1)+Ny+1;
xsat=xonb;
for i=1:N
    for j=Ny:-1:1
        if xe(i,j)>=0
            xsat(i)=j;
        elseif (sqrt(mean(T(mesh.id.num==i,j,Nz)))-
sqrt(Tsat(i,j)))>=sqrt(2*matp.sigma*q_w(i,j)/densityg(matp,Tf(i,j))/mat
p.h_fg/thermcond(matp,Tf(i,j)))
            xonb(i)=j;
        end
    end
end
h=zeros(N,Ny);
for i=1:N
    for j=1:Ny
        Ltf=dy*j;
        if xe(i,j)>0 && xe(i,j)<1
h(i,j)=twophaseh(dims,matp,flow,Tf(i,j),q_w(i,j),xe(i,j),fmald(i),ws,Lt
f);
        elseif xe(i,j)<=0
            if j<xonb(i)
h(i,j)=singlephaseh(dims,matp,flow,Tf(i,j),fmald(i),ws(i,:),Ltf);
            else
h1=singlephaseh(dims,matp,flow,Tf(i,j),fmald(i),ws(i,:),Ltf);
h2=twophaseh(dims,matp,flow,Tf(i,j),q_w(i,j),xe(i,j),fmald(i),ws,Ltf);
                xstar=(j-xonb(i))/(xsat(i)-xonb(i));
                h(i,j)=(1-xstar)*h1+xstar*h2;
            end
        else
h(i,j)=vaporphaseh(dims,matp,flow,Tf(i,j),fmald(i),ws(i,:),Ltf);
        end
    end
end
assignin('base','xonb',xonb);
assignin('base','xsat',xsat);
end
function rho=densityg(matp,Tf)
rho=matp.rho_ga*Tf^3+matp.rho_gb*Tf^2+matp.rho_gc*Tf+matp.rho_gd;
end
function k=thermcond(matp,Tf)
k=matp.kfa-matp.kfb*Tf;
end
```

Function 40. inputs.m

```
function [dims,matp,flow,getdata,filename]=inputs()
[out]=gui_input2;
dims=out.dims;
flow=out.flow;
flow.heating_case=out.heating_case;
power=out.power;
Phi=out.Phi;
flow.Phi=Phi;
getdata=out.getdata;
filename=out.filename;
heaters=out.heaters;
if getdata
    matp=0;
    return
end
dims.w_ef=(dims.W-dims.N*dims.w-(dims.N-1)*dims.w_f)/2;
switch out.heatsink
    case 'Silicon'
        matp.k_Si=140;
    case 'Copper'
        matp.k_Si=388;
end
switch out.fluid
    case 'FC-77'
        matp.h_fg=89000;
        matp.c_pfa=1014; %cpf=1014+1.554*T[C]
        matp.c_pfb=1.554;
        matp.c_pga=0.0019; %cpg=0.0019*T[K]+0.3031
        matp.c_pgb=0.822085;
        matp.sigma=0.013; %at STP, sigma=0.0062 used by Tannaz
        matp.rho_fa=1838; %rhof=1838-2.45*T[C]
        matp.rho_fb=-2.45;
        matp.rho_ga=2.9064e-5; %15.84 used by Tannaz, rhog=2.9064e-
5*T[C]^3-2.9053e-3*T[C]^2+0.17218*T[C]-2.6758
        matp.rho_gb=-2.9053e-3;
        matp.rho_gc=0.17218;
        matp.rho_gd=-2.6758;
        matp.mu_f=0.0004655; %0.00052 used by Tannaz, muf=0.0013 at STP
        %at 95C nuf=0.29 cSt, rhof=1605.25 kg/m3 => muf=0.0004655
        matp.mu_g=0.00002; %mug[cP]=2.953e-9*T[C]^3-1.045e-
6*T[C]^2+1.528e-4*T[C]+3.76e-3
        matp.kfa=0.065; %kf=a-bT, kf=0.065-0.00008*T[C]
        matp.kfb=0.00008;
        matp.k_ga=0.0000569; %kg=(0.0569*T[C]+5.4799)/1000
        matp.k_gb=0.0054799;
        matp.p_c=1.58e6;
        matp.M=416;
        matp.Tsat=97;
        matp.Tsata=1928;
        matp.Tsatb=10.216;
        matp.Tsatc=1;
    case 'HFE 7100'
        matp.h_fg=112000;
        matp.c_pf=1233; %at 50 C, 1183 at 25 C
        matp.c_pg=898.682;
        matp.sigma=.0136;
        matp.rho_f=1429; %at 55 C, 1510 at 25 C
        matp.rho_g=9.87;
        matp.mu_f=0.00039; %at 55 C, 0.00058 at 25 C
        matp.mu_g=0.00001113;
        matp.kfa=0.073714; %kf=a-bT
        matp.kfb=0.00019548;
        matp.k_g=0.01586;
        matp.p_c=2.23e6;
        matp.M=250;
        matp.Tsat=61;
        matp.Tsata=3641.9;
        matp.Tsatb=22.415;
        matp.Tsatc=1/log10(e);
end
matp.e_rb=0.82;
matp.e_rs=0.1;
Q_in=zeros(5,5);
switch out.heating_case
    case 'Case 1a'
        Q_in(:,3)=power/5/dims.L/dims.W*25;
    case 'Case 1b'
        Q_in(3,:)=power/5/dims.L/dims.W*25;
    case 'Case 1c'
        Q_in(:,1)=power/5/dims.L/dims.W*25;
    case 'Case 1d'
        Q_in(:,1)=power/10/dims.L/dims.W*25;
        Q_in(:,5)=power/10/dims.L/dims.W*25;
    case 'Case 2a'
        Q_in(:,:)=power*(1-Phi)*4/100/dims.L/dims.W*25;
        Q_in(:,3)=power*(1/5+4/5*Phi)/5/dims.L/dims.W*25;
    case 'Case 2b'
        Q_in(:,:)=power*(1-Phi)*4/100/dims.L/dims.W*25;
        Q_in(3,:)=power*(1/5+4/5*Phi)/5/dims.L/dims.W*25;
    case 'custom'
        Nh=sum(sum(heaters));
        Q_in(heaters==0)=power*(1-Phi)/dims.L/dims.W;
        Q_in(heaters==1)=power*(Phi*(25-Nh)+Nh)/dims.L/dims.W/Nh;
end
flow.Q_in=Q_in;
end
```

Function 41. pressuredrop.m

```
function [DP,DPsp,DPtp]=pressuredrop(dims,matp,mesh,flow,xe,Tf)
N=dims.N;
Ny=mesh.Ny;
dy=mesh.dy;
DP=zeros(N,Ny);
DPsp=DP;
DPtp=DP;
for i=1:N
    for j=1:Ny
        if xe(i,j)>0 && xe(i,j)<1
            %two phase
            Ltf=j*dy;
            mutp=xe(i,j)*matp.mu_g+(1-xe(i,j))*matp.mu_f;
            rhotp=1/((1-
xe(i,j))/density(matp,Tf(i,j))+xe(i,j)/densityg(matp,Tf(i,j)));
            DPtp=singlephasedp(dims,mutp,rhotp,mesh,flow,Ltf);
        else
            %single phase
            Ltf=j*dy;
            mu=matp.mu_f;
            rho=density(matp,Tf(i,j));
            DPsp(i,j)=singlephasedp(dims,mu,rho,mesh,flow,Ltf);
        end
    end
end
DP=DPsp+DPtp;
end
function rho=density(matp,Tf)
rho=matp.rho_fa+matp.rho_fb*Tf;
end
function rhog=densityg(matp,Tf)
rhog=matp.rho_ga*Tf^3+matp.rho_gb*Tf^2+matp.rho_gc*Tf+matp.rho_gd;
end
```

Function 42. singlephasedp.m

```
function DPsp=singlephasedp(dims,mu,rho,mesh,flow,Ltf)
Dh=4*dims.d*dims.w/(2*dims.d+dims.w);
AR=min(dims.d/dims.w,dims.w/dims.d);    %always greater than 1
sqA=sqrt((AR+1)^2*Dh^2/(4*AR));
G=flow.m_dot*rho/dims.N/60/1e6/dims.w/dims.d;
Resq=abs(G*sqA/mu);
Re=abs(G*Dh/mu);
if Re<=2300
    zcross=Ltf/sqA/Resq;
    fre=((3.44/sqrt(zcross))^2+(12/(sqrt(AR)*(1+AR)*(1-
192*AR/pi^5*tanh(pi/(2*AR)))))^2)^(1/2);
    f=4*fre/Resq;
    ftot=1/2*f*mesh.dy/Dh/rho;
elseif Re>4000
    f=((0.79*log(Re)-1.64)^(-2))*(1+4/5*(Dh/Ltf)^(3/4));
    ftot=1/2*f*mesh.dy/Dh/rho;
else
    zcp=Ltf/(sqA*2300/2*(AR+1)/sqrt(AR));
    frep=((3.44/sqrt(zcp))^2+(12/(sqrt(AR)*(1+AR)*(1-
192*AR/pi^5*tanh(pi/(2*AR)))))^2)^(1/2);
    frp=4*frep/(2300/2*(AR+1)/sqrt(AR));
    Afac=Ltf/(sqA/2*(AR+1)/sqrt(AR));
    Cfac=(12/(sqrt(AR)*(1+AR)*(1-192*AR/pi^5*tanh(pi/(2*AR)))))^2;
    Efac=8/((AR+1)/sqrt(AR));
    dfr=-
Efac*(2*Afac*Cfac+3.44^2*2300)/(2*Afac*2300^2*sqrt(3.44^2*2300/Afac+Cfa
c));
    Ret=4000+1i*1e-20;
    ff=((0.79*log(Ret)-1.64)^(-2))*(1+4/5*(Dh/Ltf)^(3/4));
    ft=real(ff);
    dft=imag(ff)/1e-20;
    fft=spline([2300,4000],[dfr,frp,ft,dft]);
    f=ppval(fft,Re);
    ftot=1/2*f/rho*mesh.dy/Dh;
end
DPsp=ftot*abs(G)*G;
end
```

Function 43. singlephaseh.m

```
function [h]=singlephaseh(dims,matp,flow,Tf,fmald,~,Ltf)
Dh=4*dims.d*dims.w/(2*dims.d+dims.w);
AR=max(dims.d/dims.w,dims.w/dims.d);
G=flow.m_dot*fmald*density(matp,Tf)/dims.N/60/1e6/dims.w/dims.d;
ReD=abs(G*Dh/matp.mu_f);
Pr=matp.mu_f*specheat(matp,Tf)/thermcond(matp,Tf);
zstar=Ltf/ReD/Dh/Pr;
RHS=-1.275e-6*AR^6+4.709e-5*AR^5-6.902e-4*AR^4+5.014e-3*AR^3-
0.01769*AR^2+0.01845*AR+0.05691;
if zstar<RHS
    h=(1.766*(ReD*Pr*Dh/Ltf)^0.378*AR^0.1224)*thermcond(matp,Tf)/Dh;
else
    h=thermcond(matp,Tf)/Dh*8.235*(1-2.0421/AR+3.0853/AR^2-
2.4765/AR^3+1.0578/AR^4-0.1861/AR^5);
end
end
function rho=density(matp,Tf)
rho=matp.rho_fa+matp.rho_fb*Tf;
end
function cp=specheat(matp,Tf)
cp=matp.c_pfa+matp.c_pfb*Tf;
end
function k=thermcond(matp,Tf)
k=matp.kfa-matp.kfb*Tf;
end
```

Function 44. tdma.m

```
function [phi] = tdma(a,b,c,d)
N=length(a);    %number of grid points
P=zeros(1,N);
Q=zeros(1,N);
P(1)=b(1)/a(1);
Q(1)=d(1)/a(1);
for i=2:N
    P(i)=b(i)/(a(i)-c(i)*P(i-1));
    Q(i)=(d(i)+c(i)*Q(i-1))/(a(i)-c(i)*P(i-1));
end
phi(N)=Q(N);
for i=N-1:-1:1
    phi(i)=P(i)*phi(i+1)+Q(i);
end
end
```

Function 45. tdmaline.m

```
function [phi] = tdmaline(a,b,c,d,e,f,g)
N=size(a);        %number of grid points
                  %N(1) is number of rows
                  %N(2) is number of columns
    for i=1:N(1)          %rows first
        a1=a(i,:);
        b1=b(i,:);
        c1=c(i,:);
        if i==1
            d1=f(i,:)+d(i,:).*g(i+1,:);
        elseif i==N(1)
            d1=f(i,:)+e(i,:).*g(i-1,:);
        else
            d1=f(i,:)+d(i,:).*g(i+1,:)+e(i,:).*g(i-1,:);
        end
        g(i,:)=tdma(a1,b1,c1,d1);    %pass to tdma solver
    end
    for j=1:N(2)          %columns second
        a1=a(:,j);
        b1=d(:,j);
        c1=e(:,j);
        if j==1
            d1=f(:,j)+b(:,j).*g(:,j+1);
        elseif j==N(2)
            d1=f(:,j)+c(:,j).*g(:,j-1);
        else
            d1=f(:,j)+b(:,j).*g(:,j+1)+c(:,j).*g(:,j-1);
        end
        g(:,j)=tdma(a1,b1,c1,d1);    %pass to tdma solver
    end
    for i=N(1):-1:1       %backward rows third
        a1=a(i,:);
        b1=b(i,:);
        c1=c(i,:);
        if i==1
            d1=f(i,:)+d(i,:).*g(i+1,:);
        elseif i==N(1)
            d1=f(i,:)+e(i,:).*g(i-1,:);
        else
            d1=f(i,:)+d(i,:).*g(i+1,:)+e(i,:).*g(i-1,:);
        end
        g(i,:)=tdma(a1,b1,c1,d1);    %pass to tdma solver
    end
    for j=N(2):-1:1       %backward columns fourth
        a1=a(:,j);
        b1=d(:,j);
        c1=e(:,j);
        if j==1
            d1=f(:,j)+b(:,j).*g(:,j+1);
        elseif j==N(2)
            d1=f(:,j)+c(:,j).*g(:,j-1);
        else
            d1=f(:,j)+b(:,j).*g(:,j+1)+c(:,j).*g(:,j-1);
        end
        g(:,j)=tdma(a1,b1,c1,d1);    %pass to tdma solver
    end
phi=g;
end
```

Function 46. twophaseh.m

```
function [h]=twophaseh(dims,matp,flow,Tf,q_w,xe,fmald,~,Ltf)
Dh=4*dims.d*dims.w/(2*dims.d+dims.w);
G=flow.m_dot*fmald*density(matp,Tf)/dims.N/60/1e6/dims.w/dims.d;
ReD=abs(G*Dh/matp.mu_f);
Pr=matp.mu_f*specheat(matp,Tf)/thermcond(matp,Tf);
hcl=(3.66+(0.0668*Dh/Ltf*ReD*Pr)/(1+0.04*(Dh/Ltf*ReD*Pr)^(2/3)))*thermc
ond(matp,Tf)/Dh;
Gv=flow.m_dot*fmald*densityg(matp,Tf)/dims.N/60/1e6/dims.w/dims.d;
ReDv=abs(Gv*Dh/matp.mu_g);
Prv=matp.mu_g*specheatg(matp,Tf)/thermcondg(matp,Tf);
hcv=(3.66+(0.0668*Dh/Ltf*ReDv*Prv)/(1+0.04*(Dh/Ltf*ReDv*Prv)^(2/3)))*th
ermcondg(matp,Tf)/Dh;
Co=(matp.sigma/(9.81*(density(matp,Tf)-densityg(matp,Tf))*Dh^2))^(1/2);
Pred=flow.P_in/matp.p_c;
hnb=55*Pred^(0.12-0.2*log(matp.e_rb)/log(10))*(-log(Pred)/log(10))^(-
0.55)*matp.M^(-0.5)*abs(q_w)^0.67;
hconv=hcl*(1-xe)+hcv*xe;
h=hnb*(1-xe)+hconv*(1+80*(xe^2-xe^6)*exp(-0.6*Co));
end
function rho=density(matp,Tf)
rho=matp.rho_fa+matp.rho_fb*Tf;
end
function cp=specheat(matp,Tf)
cp=matp.c_pfa+matp.c_pfb*Tf;
end
function k=thermcond(matp,Tf)
k=matp.kfa-matp.kfb*Tf;
end
function rhog=densityg(matp,Tf)
rhog=matp.rho_ga*Tf^3+matp.rho_gb*Tf^2+matp.rho_gc*Tf+matp.rho_gd;
end
function cp=specheatg(matp,Tf)
cp=matp.c_pga*Tf+matp.c_pgb;
end
function k=thermcondg(matp,Tf)
k=matp.k_ga*Tf+matp.k_gb;
end
```

Function 47. vaporphaseh.m

```
function [h]=vaporphaseh(dims,matp,flow,Tf,fmald,~,Ltf)
Dh=4*dims.d*dims.w/(2*dims.d+dims.w);
AR=max(dims.d/dims.w,dims.w/dims.d);
G=flow.m_dot*fmald*densityg(matp,Tf)/dims.N/60/1e6/dims.w/dims.d;
ReD=abs(G*Dh/matp.mu_g);
Pr=matp.mu_g*specheatg(matp,Tf)/thermcondg(matp,Tf);
zstar=Ltf/ReD/Dh/Pr;
RHS=-1.275e-6*AR^6+4.709e-5*AR^5-6.902e-4*AR^4+5.014e-3*AR^3-
0.01769*AR^2+0.01845*AR+0.05691;
if zstar<RHS
    h=(1.766*(ReD*Pr*Dh/Ltf)^0.378*AR^0.1224)*thermcondg(matp,Tf)/Dh;
else
    h=thermcondg(matp,Tf)/Dh*8.235*(1-2.0421/AR+3.0853/AR^2-
2.4765/AR^3+1.0578/AR^4-0.1861/AR^5);
end
end
function rho=densityg(matp,Tf)
rho=matp.rho_ga*Tf^3+matp.rho_gb*Tf^2+matp.rho_gc*Tf+matp.rho_gd;
end
function cp=specheatg(matp,Tf)
cp=matp.c_pga*Tf+matp.c_pgb;
end
function k=thermcondg(matp,Tf)
k=matp.k_ga*Tf+matp.k_gb;
end
```

Appendix H List of Experimental Facility Equipment

This section contains a list of equipment used in the experimental facilities described in this thesis. Table H.1 contains the equipment used in the impedance-based void fraction sensor facility. Table H.2 contains the equipment used in the non-uniform heating facility.

Table H.1. Equipment used in the impedance-based void fraction sensor facility.

Part Name	Vendor / Manufacturer	Part Number	Description
Microchannel test section	ME Research Machine Shop	Custom	Crosswise electrode configuration
Microchannel test section	Thermophysical Properties Research Laboratory, Inc.	Custom	Streamwise electrode configuration
Deionized Water	Birck Nanotechnology Center	-	Liquid
Morpholine	Sigma-Aldrich	CAS 1132-61-2	Chemical added to liquid
Ammonium hydroxide	Mallinckrodt Chemicals	CAS 1336-21-6	Chemical added to liquid
Water pump	Micropump	415A	500-6000 rpm
Micro-turbine flow meter	McMillan	Flo-106	10-100 mL/min
Micro-turbine flow meter	McMillan	Flo-106	20-200 mL/min
Air cylinder	Purdue University Stores	-	Gas
Mass flow sensor	Omega	FMA6704	0-100 mL/min
Mass flow sensor	Omega	FMA6705	0-200 mL/min
Mass flow sensor	Omega	FMA6706	0-500 mL/min
Pressure transmitter	WIKA	8642885	0-10 psig
Thermocouple	Omega	T type, sheathed	Temperature measurement
Impedance-based void fraction sensor	ME Electronics Shop	Custom	Electrical impedance measurement

Table H.1. Continued.

Part Name	Vendor / Manufacturer	Part Number	Description
High-speed camera	Photron	Fastcam Ultima APX	1024 × 1024 pixel CMOS sensor, 2000 fps at full resolution
Lens	Keyence	VH-Z50L	50-500 X
Light source	Henke-Sass Wolf		Light source
Light source	Cole Parmer	41720	Light source
Light source	Luminar Ace	LA-150UE	Light source
Camera mount	Velmex, Inc.	A6009C-S6-TL-BK	Translation stage
Camera mount	Velmex, Inc.	A6012C-S6-TL-BK	Translation stage
Camera mount	Velmex, Inc.	B6012C-S6-TL-BK	Translation stage
Data acquisition system	National Instruments	USB-6225	Acquisition system

Table H.2. Equipment used in the non-uniform heating facility.

Part Name	Vendor / Manufacturer	Part Number	Description
Microchannel heat sink	Delphi Electronics and Safety	Custom	Heat sinks
Cover plate	Thermophysical Properties Research Laboratory, Inc.	Custom	Microchannel cover plate
Pyrex sheet		-	Microchannel cover plate
FC-77	3M	-	Liquid
Liquid pump	Micropump	415A	500-6000 rpm
Preheater	Omegalux	AHPF	Inline heater
Heat exchanger	Comair Rotron	PT2B3	Liquid to air heat exchanger
Expandable reservoir	Made in house	Custom	Fluid reservoir
Vacuum pump	Thomas	2688VE44 B	Vacuum pump
Micro-turbine flow meter	McMillan	Flo-114	20-200 mL/min
Thermocouple	Omega	T type, sheathed	Temperature measurement
Pressure transducer	Gems Sensors	2200 series	0-30 psia
Differential pressure transducer	Omega	PX2300 series	+/- 10 psid

Table H.2. Continued.

Part Name	Vendor / Manufacturer	Part Number	Description
High-speed camera	Photron	Fastcam PCI 1024	1024 × 1024 pixel CMOS sensor, 1000 fps at full resolution
Lens	Keyence	VH-Z50L	50-500 X
Lens	Nikon	AF Micro-Nikkor	200 mm IF ED lens
Light source	Cole Parmer	41720	Light source
Camera mount	Velmex, Inc.	A6009C-S6-TL-BK	Translation stage
Camera mount	Velmex, Inc.	A6012C-S6-TL-BK	Translation stage
Camera mount	Velmex, Inc.	B6012C-S6-TL-BK	Translation stage
Power supply	Sorensen	DCS33-33E	0-33 V, 0-33 A
Power supply	Sorensen	DCS20-50E	0-20 V, 0-50 A
Power supply	Sorensen	DCS40-25E	0-40 V, 0-25 A
Data acquisition system	Agilent	34970A	Acquisition system

VITA

VITA

Susan Nicole Ritchey received her Bachelor of Science in Nuclear Engineering and Bachelor of Science in Radiological Health Engineering from Texas A&M University in 2007. She received her Master of Science in Nuclear Engineering from Texas A&M University in 2009 under the advisement of Prof. Karen Vierow. She is currently pursuing her PhD in Mechanical Engineering at Purdue University under the advisement of Prof. Suresh V. Garimella. She was the recipient of the Ingersoll-Rand Fellowship for graduate studies in Mechanical Engineering at Purdue University and the Ward A. Lambert Graduate Teaching Fellowship. She also received the Purdue University Graduate School's Excellence in Teaching Award.

PUBLICATIONS

PUBLICATIONS

S.N. Williams and D.E. Mueller, 2006, Survey of operating parameters for use in burnup credit calculations, Transactions of the American Nuclear Society 95, Albuquerque, NM, November 14.

K. Vierow, I. Choutapalli, K. Hogan, Y. Liao, M. Solmos, S.N. Williams, 2008, Countercurrent flow limitation experiments and modeling for improved reactor safety, NEER Technical Report DE-FG07-05ID14696, US DOE.

S.N. Williams, 2009, Flooding experiments with steam and water in a large diameter vertical tube, M.S. thesis, Texas A&M University.

S.N. Williams, M. Solom, O. Draznin, I. Choutapalli, K. Vierow, 2009, Flooding experiments with steam and water in a large diameter vertical tube, 13th International Topical meeting on Nuclear Reactor Thermal Hydraulics (NURETH-13), Kanazawa, Japan, September-October.

O. Draznin, S.N. Ritchey, K. Vierow, 2010, Experimental study of water subcooling effect on steam-water flooding in a large diameter vertical tube, International Congress on Advances in Nuclear Power Plants (ICAPP), San Diego, CA, June 13-17.

S. Paranjape, S.N. Ritchey, S.V. Garimella, 2011, Impedance-based void fraction measurement and flow regime identification in microchannel flows, Pacific Rim Technical Conference & Exposition on Packaging and Integration of Electronic and Photonic Systems (InterPACK), Portland, OR, July 6-8.

S.N. Ritchey, M. Solom, O. Draznin, I. Choutapalli, K. Vierow, 2011, Flooding experiments with steam and water in a large diameter vertical tube, Journal of Nuclear Technology 175, pp. 529-537.

S. Paranjape, S.N. Ritchey, S.V. Garimella, 2012, Electrical impedance-based void fraction measurement and flow regime identification in microchannel flows under adiabatic conditions, International Journal of Multiphase Flow 42, pp. 175-183.

S.N. Ritchey, J.A. Weibel, S.V. Garimella, 2013, Effects of Non-Uniform Heating on Two-Phase Flow through Microchannels, Pacific Rim Technical Conference & Exposition on Packaging and Integration of Electronic and Photonic Systems (InterPACK), Burlingame, CA, July 16-18, Paper No. InterPACK2013-73058.

S.N. Ritchey, J.A. Weibel, S.V. Garimella, 2014, Local Measurement of Flow Boiling Heat Transfer in an Array of Non-Uniformly Heated Microchannels, International Journal of Heat and Mass Transfer 71, pp 206-216.

P. Valiorgue, S.N. Ritchey, J.A. Weibel, S.V. Garimella, 2014, Design of a Non-Intrusive Electrical Impedance-Based Void Fraction Sensor for Microchannel Two-Phase Flows, Measurement Science and Technology 25, 095301.

S.N. Ritchey, J.A. Weibel, S.V. Garimella, Effects of Non-Uniform Heating on the Location and Magnitude of Critical Heat Flux in a Microchannel Heat Sink, International Journal of Micro-Nano Scale Transport, *in preparation*.