PERFORMANCE CHARACTERIZATION OF A HELIUM PHP VIA FLUENT

By

Chen Xu

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN - MADISON

2022

Date of final oral examination: 04/27/2022

The dissertation is approved by the following members of the Final Oral Committee:

John M. Pfotenhauer, Professor, Mechanical Engineering

Franklin K. Miller, Professor, Mechanical Engineering

Frank Pfefferkorn, Professor, Mechanical Engineering

Jennifer Franck, Assistant Professor, Engineering Physics

Mark Anderson, Assistant Professor, Mechanical Engineering

Abstract

This thesis explores the development of a 3D model using ANSYS FLUENT 17.2 to characterize the two-phase (gas/liquid) flow of helium in a cryogenic pulsating heat pipe. The model includes condensation and evaporation phenomenon and uses the VOF model to capture aspects of the two-phase flow. Temperature dependent properties were applied for helium in the range between 2.2 K and 20 K. Bubble formation and movement are observed as well as flow direction at various times in each of the individual pipes. Grid independence tests are performed by plotting the average temperature of different sections. The influence of the number of turns on the performance of a PHP is also investigated.

Various challenges were encountered and solved during the process of modeling this complex two-phase problem. Firstly, an unbalanced mass problem was solved by using a real gas model for gaseous helium. Secondly, there is no data documenting the evaporation rate of helium. As a result, the evaporation rate was fitted as a function of heat flux based on the experimental data. Thirdly, the available experimental data utilize a large geometry which results in a huge amount of mesh nodes. Using a different mesh density for the radial and axial direction of the geometry and using a virtual wall for modeling the copper blocks in the evaporator and condenser regions significantly reduced the number of mesh nodes. Fourthly, the computational time had to be reduced to make the model more efficient. Using a guessed initialization velocity, temperature and pressure proved to help the PHP model reach steady state faster. Fifthly, the model showed a temperature oscillation which was not present in the experimental data. The problem was solved by adding the copper block that matches the copper block configuration in experiments and this feature stabilized the temperature eliminating the temperature oscillations.

Parametric investigations were conducted to investigate the effects of geometry and operational conditions on the performance of the PHP. The results show that the flow velocity increases with heat flux, providing a possible explanation for the increase of effective thermal conductivity with heat flux observed in experiments. The fill ratio is the primary factor determining whether the unidirectional flow is steady or pulsating. A higher fill ratio leads to a higher possibility of steady circulatory flow. Characteristic frequencies for the pulsating flow vary between 0.2 to 0.9 Hz. Finally, a higher heat flux leads to a higher mass transfer rate, but a lower percentage of heat transfer via latent heat.

Acknowledgement

I would like to thank my academic advisor, Professor John for accepting me as his PhD student and for trusting me with this important project. Your positivity, encouragement and understanding were crucial for the success of both my schoolwork and research work. I would also like to thank my co-advisor, Franklin Miller; your insights were also so important to me. This project will not be successful without your input and guidance. Your knowledge in numerical modeling and PHPs helped me tremendously. Meeting with Professor John and Professor Miller has always been my favorite part of the week. That is because every time after a meeting, I feel refreshed, filled with new ideals and guidance, and the positive energy to make progress again.

I would like to thank the department of Mechanical Engineering for supporting me with a TA to support my degree financially. Also, thanks for maintaining such a high-quality graduate program. I would like to thank Sumitomo Cryogenics of American for the funding and the engineer team there for giving me feedback about my research project.

My gratitude goes to my family, who have always been supportive of whatever I do and was particular in support of me pursuing a higher education in engineering. I feel more connected to my father, who is also a successful engineer after this PhD project. And this endeavor made me more appreciative of how much work he puts into his career. I would like to thank you for both your financial support and emotional support. My mother, who always provided me with unconditional love and gave me freedom to explore the world. I dedicate this thesis to you.

To my best friend, Xinyi Liu. Your patience and kindness made me a better person. You mellowed my temperament and made me realize that kindness always wins. One of the luckiest days of my life was the day I met you on the airplane when I came to study in the US. And we have been best friends since that day. I am glad we are the kind of friends that never had any fight or disagreement. And always support and love each other. The same gratitude goes to my other friends: Dian Yao, Dan Wu, Jinli Shen.

My thanks go to Courtney Leeds, who showered me with friendliness and kindness. Your good working ethic as an engineer is inspiring. I enjoy those time talking about Ansys Fluent with you. Uzoma Mmeje, who is always there whenever I need help. You opened my eyes to how helpful a person can be. You helped me tremendously with my qualifying exam and so many other things. You are one of the smartest people and best engineers I ever met. Conversing with you and hanging out with you has always been fun and interesting. Many thanks also go to Jennifer Detlor, who invited me to PhD writing camp with her. Drafting this thesis with you at Memorial Union is certainly one of my favorite memories in Madison. You filled my PhD life with smiles and laughter. I would also like to thank other engineers I met during my PhD program: Ahmad Khayyat, Chaitanya Kavuri, Zhimin Guo, and Anna Dreyson.

Table of Content

List of Figures

List of Tables

1. Introduction

Pulsating heat pipes are a heat transfer device that has received much attention for the past three decades since their invention, primarily because their large value of effective thermal conductivity, or equivalently, a small temperature difference between the heat source and heat sink. Pulsating heat pipes are tubes that are filled partially with a working fluid and have a small enough diameter so that the surface temperature forces exceed gravitational forces. This condition results in a random distribution of liquid and vapor regions within the tubes. When heat is applied to the heated section, liquid evaporates and then condenses in the condenser section where the heat is removed.

Although being invented in 1991, many of the details regarding the PHP operation are not fully understood by researchers. Numerous experimental data with both room temperature and cryogenic temperature have been gathered by researchers all over the world. However, there is no existing model to effectively predict the effective thermal conductivities given certain fluids, geometry and heat load. Furthermore, the effect of different sections lengths, number of turns, and heat load on the effective thermal conductivities is still not predictable.

The primary goal of this research is to get a better understanding of the working principle of PHPs and how different parameters affect its ability to transfer heat. An intermediate goal is to be able to predict the effective thermal conductivity based on a given geometry and heat load. The ultimate goal of the model is to provide a reliable description of the various types of two-

phase flow that develop within the PHP, the conditions that cause transitions between the flow types, and their influence on the overall thermal performance of the PHP.

Although numerical studies with room-temperature PHPs have been performed, no published efforts except a spring-mass-damper model by Fonseca [1], its subsequent improvement by Mueller [2] , and a study carried by Tang Kai [3] have been reported to simulate a pulsating heat pipe with helium as the working fluid. This work is dedicated to the 3D modelling of helium-based pulsating heat pipes using ANSYS Fluent. The numerical models developed utilize the HPC (high performance computer center) at the University of Wisconsin – Madison in batch mode. Experimental data from similar operating conditions are used to compare with the simulation results.

1 Literature Review

1.1 Cryogenic PHPs

In the cryogenic temperature range, the PHP is a heat transfer device that has thermal conductivities several orders of magnitude above solids. As a result, it is expected to be a good cooling option for superconducting magnets [4]. Cryogenic PHPs using hydrogen, neon and nitrogen as a working fluid have been developed for this purpose as discussed by Mito and Bonnet [5, 6]. PHPs have important applications for NASA as well. First, PHPs can be used for cooling cryogen storage tanks such as a hydrogen tank. Second, PHPs are good solutions for transferring heat over long distances because of their tubing structure.

In addition, nitrogen PHPs have also been developed for an ultra-fast cooling solution for cell cryopreservation which is several orders of magnitude faster than the traditional method [7].

The traditional cooling method for cell cryopreservation requires permeating cryo-protective agents (CPA) to protect the cells. However, CPA can cause damage to cells and can be hard to remove from cells. The new cooling method using PHPs is proposed to achieve much faster cell cryopreservation with lower concentration of CPA.

Bonnet [6] built a 5 turn – 177 mm helium PHP which achieved a maximum power of 145 mW with a cold source temperature of 4.2 K. Xu [8] built a 4 turn – 200 mm PHP and concluded that the PHP has an 'optimum' fill ratio that produces a maximum thermal conductivity. Fonseca [4] built a helium-based PHP with 330 mm total length and 33 turns and successfully measured the heat transfer properties of the device. Additionally, Fonseca also [1] built a helium-based PHP with both 300 mm and 1000 mm adiabatic lengths and 21 turns with heat loads from 0.025 W up to 0.8W. And Last but not least, Li, Li and Xu [9-10] conducted experiments with both a 4 turn- and 24 turn- helium PHP with an overall length of 200mm and a heat load from 0.03 W up to 1.1W. The data from these experimental studies form a valuable resource for comparison with the present modeling study [9]. Their study also shows that with a fill ratio of 48.8%, large temperature fluctuations occur when the heat load is medium, but these disappear when the heat load is high [10]. Pfotenhauer et al built a 4 turn - 1222mm PHP to test the effects of aspect ratio between the vertical section and horizontal section on the performance of PHP as illustrated in Figure 1-1 [11]. It is found that the performance of the PHP decreases as the geometry transitions from vertical to horizontal.

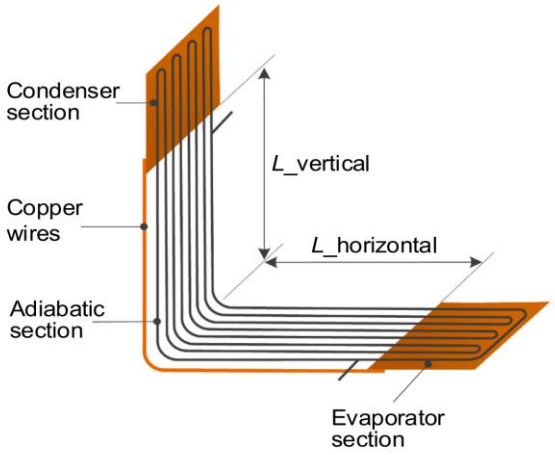

Figure 1-1: Schematic plot of PHP with aspect ratio $\propto = L_{verticle}/L_{horizontal}$

A theoretical study carried out by D. Xu et al [12] showed that for a helium PHP, only 1%-20% of heat transfer is by latent heat. Furthermore, the proportion decreases with increasing temperature and fill ratio. Monan Li et al [13] conducted a theoretical study and concluded that the start-up power of a PHP increases with increasing fill ratio. Xiao Sun [14] conducted a theoretical study about fill ratio and concluded that the effective thermal conductivity increases first and then decreases with fill ratio.

1.2 Numerical studies of PHPs

In the numerical study conducted by Raffles [15] built using FORTRAN, a PHP with and without bubble generation were compared. It was concluded that with bubble generation, both latent heat and sensible heat were increased. The latent heat was increased due to the increasing mass transfer while the sensible heat was increased by the increased amplitude of motion of the plus and slugs.

A 2D numerical study was conducted by Fadhl [16] for a straight tube thermosiphon with the VOF method and the Lee model within Fluent. A thermosiphon has a number of similarities with pulsating heat pipes, for example, in the way that they both have gravity, surface tension, evaporation and condensation influencing their behavior. Thermosiphons do not satisfy the critical diameter criteria that makes surface tension the dominant force in a PHP. However, this model showed good agreement with experimental data displaying a relative error for the evaporator, adiabatic, and condenser average temperatures of 7.9%, 9.9%, and 1.9%, respectively. The model over-predicted the thermal resistance. Nevertheless, the study demonstrates that Fluent can be a valuable tool for modeling thermally driven two-phase flows.

A single loop pulsating heat pipe was simulated by Jiangshan Wang [17] to investigate the effects of adding a corrugated configuration as demonstrated in Figure 1-2 .

Figure 1-2: Corrugated PHP

Simulations depicted a $150mm$ long water PHP with input power ranging from $5W$ to $40W$ and the fill ratio ranging from 0.3 to 0.6. Three corrugated configurations were investigated with a corrugated evaporator, adiabatic and condenser, respectively. This study proved the geometry sensitivities of modeling PHPs, and as a result, the dimensionality of our numerical model should be carefully selected.

Wang chose Fluent with a VOF model and Lee model to build the numerical model. The k-epsilon viscous model and a standard wall function were adopted. Constant heat flux, zero heat flux, and convective boundary conditions were used for the evaporator, adiabatic, and condenser boundary conditions, respectively. The investigation showed that the numerical method predicted the thermal resistance of the experimental data very well with a maximum 20% deviation of thermal resistance which was reasoned as due to variations in initial conditions and the heat loss of the experimental apparatus. The Lee model was chosen to simulate mass transfer and Lee model frequencies were set as 0.1 for both the evaporator and the condenser. The results also showed a 29% decrease in start-up time and a 38% improvement in thermal resistance with the corrugated configuration. Both corrugated evaporator and condenser sections showed improved heat transfer performance.

A CFD model of a single loop PHP was built by Li et al [18] to investigate the effects of the adiabatic section parameters on the performance of the PHP. The group concluded that by increasing the adiabatic length, the start-up time of the PHP becomes shorter. A methanol-based PHP [19] was also successfully modeled using Ansys CFX. The study concluded that the optimum fill ratio for a methanol-based is 60% and the thermal resistance of the PHP decreases with increasing heat input.

A 4-turn 870mm-long water PHP was modeled by Pouryoussefi [20] using Fluent with the VOF method. However, the mass transfer mechanism was not mentioned in the paper. Constant heat flux and constant temperature boundary conditions were applied for heating and cooling sections respectively. The simulation successfully produced vapor plugs and liquid slugs and their movement. The results show that the PHP is a chaotic system due to the absence of dominating peaks in the power spectrum density. The results also show that the optimal filling ratio of the PHP is 60%. However, there is no experimental comparison for this study. Duy-Tan et al [21] developed a 3D numerical model for an R123-based PHP using Fluent and concluded that the use of density as both a function of temperature and pressure is crucial for the success of the modeling. The circulating behavior that was observed in the model matches with the circulating behavior that was observed in the visualization experiments done by the same group.

All the simulations mentioned above were performed with room-temperature PHP fluids. Kalpak [22] simulated a nitrogen-based PHP with 3 turns and 20 mm total length using Fluent. It was concluded that thermal oscillations are present with a dominant frequency in the range of 0.25-1.85 Hz. Helium-based and nitrogen-based PHPs with 3 turns and 330 mm length were simulated with Fluent, utilizing a VOF model and Lee model by Tang Kai [3]. The effects of diameter, fill ratio, heat flux, and the number of turns were investigated. However, this model was verified by comparing the simulation data using water as the circulating fluid against experimental data from a water-based PHP.

Helium PHPs have different fluid properties and temperature range than room-temperature PHPs. As a result, even though the model for a helium PHP uses the same VOF method and mass transfer mechanism as a room temperature PHP, the meshing and numerical

methods for discretization are set differently. In addition, the room temperature PHP simulation used Lee model frequencies of 0.1 for both the evaporation and condensation processes, while the Lee model frequencies in this work for the helium PHP simulation are shown to depend on the evaporator heat flux.

Finally, the only previous helium PHP model was limited to a 2D model [3]. This work is dedicated to the 3D modelling of helium-based pulsating heat pipes using ANSYS Fluent. Numerical models were calculated using the HPC (high performance computing) cluster in the CHTC (center for high throughput computing) in the University of Wisconsin – Madison. Experimental data of similar operating conditions were used to compare with and calibrate the simulation results.

1.3 PHP System Flow Visualization

Tong [23] conducted a visualization study with a 7 turn PHP that had an overall length of 160 mm and used methanol as the working fluid. The methanol is observed to oscillate during start-up as shown in Figure 1-3 and circulates in a steady operating state as shown in Figure 1-4. Furthermore, according to their observation, the direction of circulation is consistent once the circulation is obtained but the circulation direction varies with different conditions.

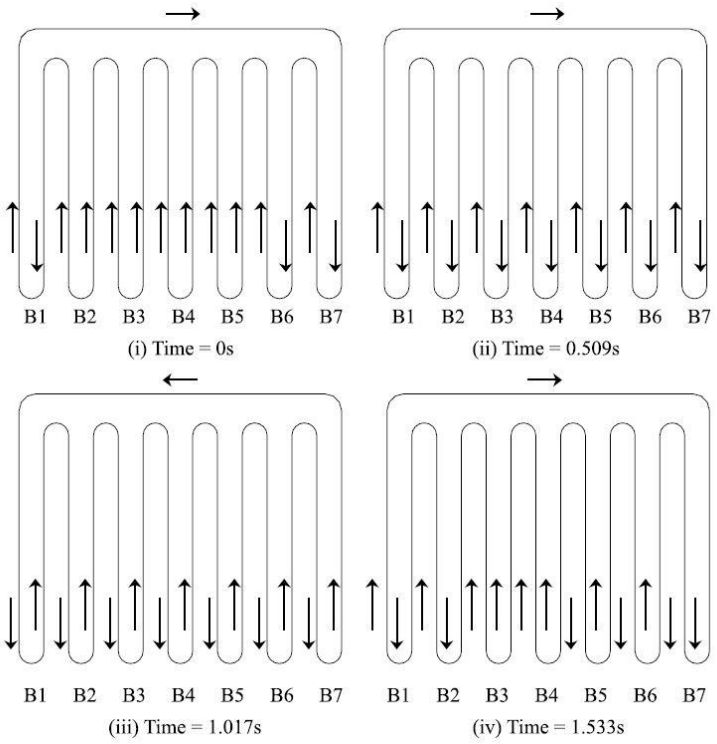

Figure 1-3: Flow direction of PHP during start-up

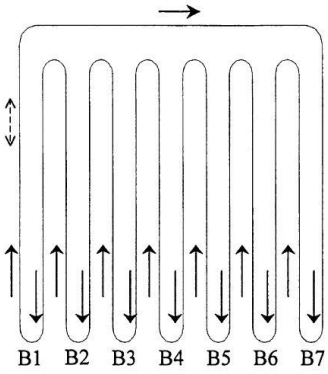

Figure 1-4: Flow direction of PHP at steady state

Sameer [24] also conducted a visualization study and found that in the adiabatic section semi-annular flow exists in alternating tubes as shown in Figure 1-5. He also mentioned that the alternate tubes are alternatively hot and cold.

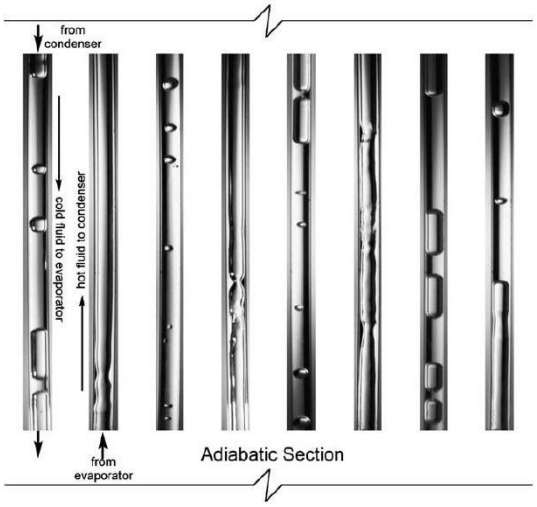

Figure 1-5: Flow pattern of alternative tubes in PHP

In a visualization study by Xu [25], bulk circulation flow as well as local switching flow were both observed. Furthermore, bubble displacements and velocities both display sinusoidal oscillations as shown in Figure 1-6. With 2 mm as the diameter of the PHPs, they also observe both short ($1mm$) and long (100 mm) vapor plugs in a methanol PHP while only short (1 mm) vapor plugs in a water PHP.

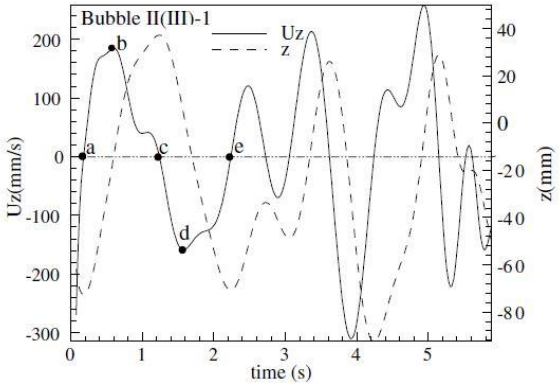

Figure 1-6: Velocity and bubble displacement in PHP

2 Fluent set up

This chapter includes all the setup steps for building a numerical model for PHPs in Fluent. In general, the necessary steps are as follows:

1. The choice of dimensionality is set based on the direction of surface tension and gravity forces in the PHPs.
2. Geometry and mesh are established utilizing mesh independence studies.
3. The properties of both gaseous and liquid helium are determined and integrated into the numerical model.
4. The model options of Fluent (detailed below) are set to capture the multiple physics phenomenon behind a working PHP and the numerical method is chosen to ensure the convergence of all the numerical equations.
5. Lee model frequencies are set as a variable parameter.
6. The time step and iteration are chosen to obtain a converged solution for a transient numerical model.
7. Key parameters are identified, and transient solutions are saved to analyze the results.

2.1 Dimensionality consideration

There are temperature variations in both the axial and radial direction of the tubes. As a result, at least 2D models should be used for PHP modeling. However, for a 2D Fluent model, the z-direction is considered to extend 1m into the paper. Moreover, even though the PHP is

axisymmetric along a meandering line, the axisymmetric option in Fluent is restricted to a straight axisymmetric line. Lastly, orienting a PHP horizontally will result in the direction of gravity being perpendicular to the direction of surface tension, and 3D modeling would be necessary for such a case. For these reasons, 3d modeling was selected to model the PHPs.

2.2 Geometry

2.2.1 Fluid Geometry

The geometry is modeled in the Fluent 'Design Modeler' including values for tube diameter, number of turns, tube length, bending radius. All these parameters can be easily changed by design point. Figure 2-1 is a schematic representation of the PHP geometry. In ANSYS Design Modeler, the PHP is first cut into two parts for the purpose of consistent sweeping in the subsequent meshing process. Secondly, Shared topology in Design Modeler is turned on so that ANSYS meshing can process the PHP as a connected geometry.

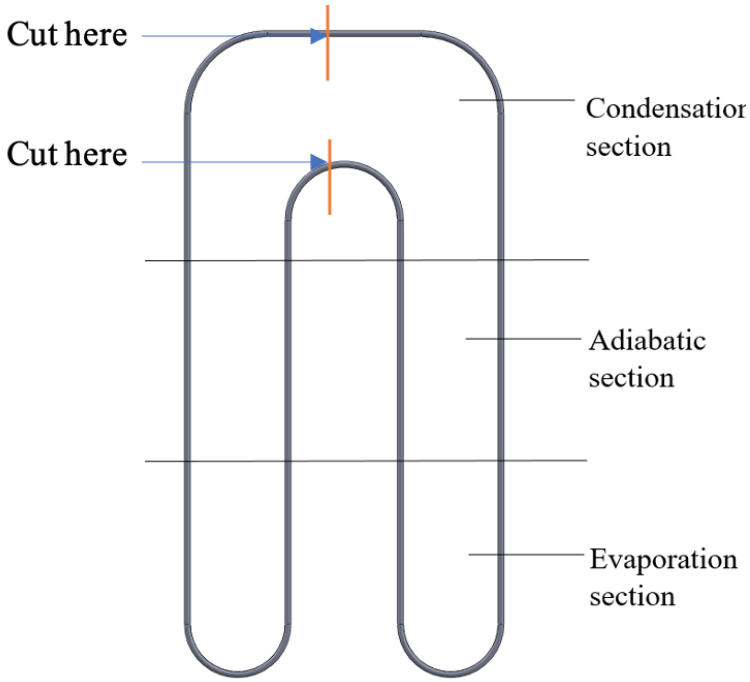

Figure 2-1: Schematic PHP geometry

2.2.2 Solid geometry

Tubing walls around the fluid were not modeled in Fluent for the following reasons. At a typical operating temperature, the thermal conductivity of helium is 0.0085 $w/(m \cdot k)$ while the thermal conductivity of stainless steel and copper are 0.5 $w/(m \cdot k)$ and 642 $w/(m \cdot k)$, respectively. Due to this large difference, it is safe to assume there are variations of the fluid temperature even while the temperature of the wall remains constant. In addition, after conducting a comparison study of modeling the PHP with and without solid walls, it is evident that the temperature profile is very similar for both cases.

On the other hand, the copper blocks that connect to the evaporator and condenser sections are modelled after a comparison study described in detail in chapter 9.4.

2.3 Mesh

2.3.1 Inflation layers

Inflation layers were added around the walls for the following reason. Without inflation layers and at low heat input, the model demonstrates that the heat coming into the copper block matches the heat input into the evaporator outer wall. However, without inflation layers and at high heat input the two heat flow values do not match. When inflation layers are added, the model is able to predict all levels of evaporator heat. Furthermore, without inflation layers, the model fails to generate bulk circulation at high heat input. In comparison, using the inflation layer, the velocity was able to maintain bulk circulation. One possible reason for the difference is that the inflation layer helped to predict the boundary layer more accurately and hence increase the sensible heat and help circulation. The heat flux plot with and without inflation layers is shown in Figure 2-2. Here the term ‘block-back’ refers to the heat applied to the back of the copper block. On the left plot, the evaporator heat is lower than the heat that is applied on the block back. In contrast, on the right plot, the evaporator heat is nearly the same as the heat that is applied on the block back. As a result, the right plot has a better heat balance as a system.

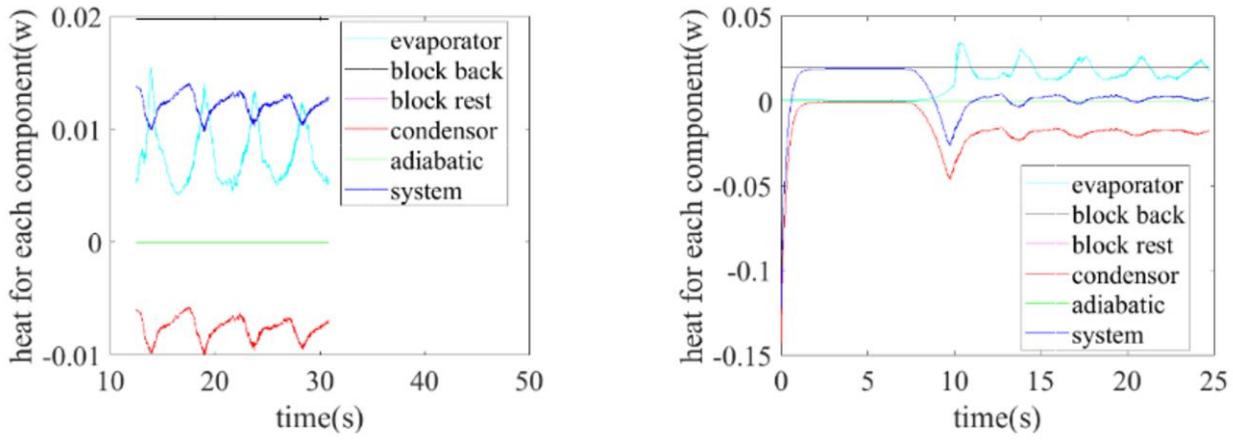

Figure 2-2: Heat transfer plots without inflation layer and with inflation layer

2.3.2 Fluid meshing parameters choice

There are a couple of meshing parameters that must be fixed to properly mesh a PHP. Firstly, the radial mesh size for the cross-section area. Second, the axial mesh size for the tube direction. Thirdly, two more meshing parameters need to be fixed because the model considers inflation layers: the number of inflation layers and the total thickness of the inflation layers. Note that the growth rate of the inflation layer is set to 1.2 by default.

Mesh studies were conducted to decide the meshing parameters. A 2-turn- 50 mm PHP with a 70% fill ratio and 10 mm evaporator length, 30 mm adiabatic length, and 10 mm condenser length was chosen for the study. A 0.001W heat load was applied to the back of the copper block of the PHP. Ideally, the heat input on the back of the copper block should match the heat coming into the evaporator.

Different meshes with different combinations of radial and axial mesh sizes along with inflation layers were tested for the sensitivity study. The radial mesh sizes included 0.2mm, 0.15mm 0.1mm, 0.05mm. The axial mesh sizes included 2mm, 1.5mm, 1mm, 0.5mm. The

number of inflation layers included 5, 10, 15, 20, and the inflation layer total thickness included values of 0.1mm, 0.05mm, and 0.01mm. Heat coming into the evaporator was plotted as a function of mesh size for each set of meshing parameters as shown in Figure 2-3, Figure 2-4, Figure 2-5, and Figure 2-6. As shown in the figures, when the mesh size is sufficiently small, the heat coming into the evaporator eventually matches the heat put into the copper block and remains at the same value of applied heat even though the mesh sizes become even smaller.

Consequently, the coarse mesh with a mesh size of 1 mm in the axial direction, 0.1 mm in the radial direction and with 5 inflation layers and a total thickness of 0.05 mm for the inflation layers was chosen for the PHP simulation as show in Figure 2-7 and Figure 2-8.

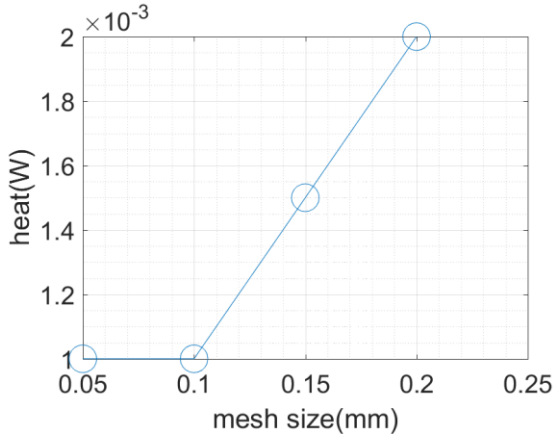

Figure 2-3: Radial mesh sizes sensitivity study

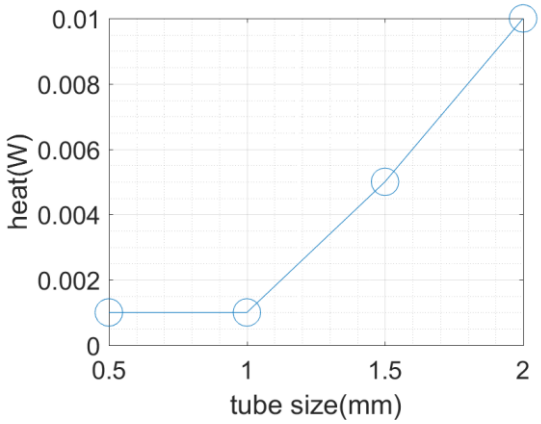

Figure 2-4: Axial mesh sizes sensitivity study

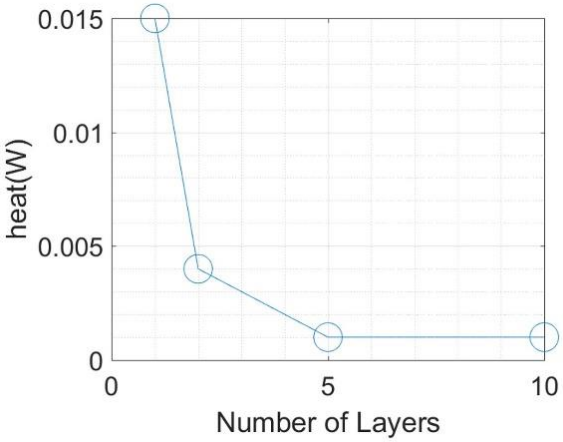

Figure 2-5: Number of inflation layers sensitivity study

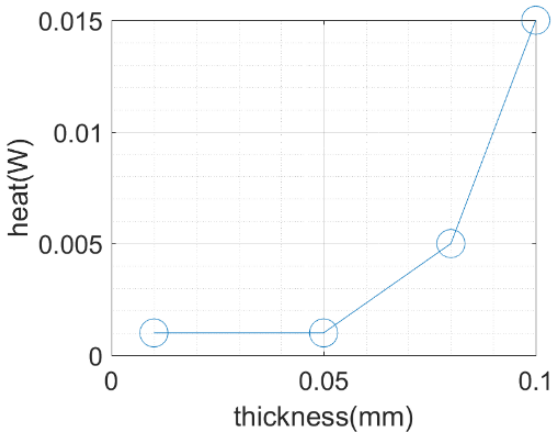

Figure 2-6: Total thickness for inflation layers sensitivity study

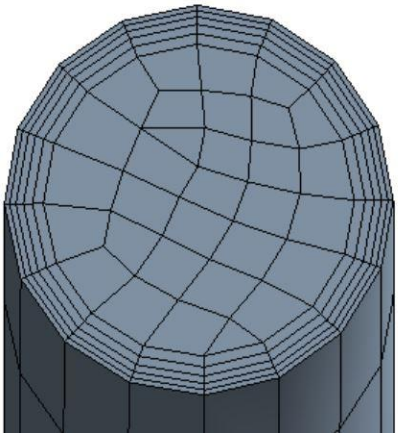

Figure 2-7: Mesh-radial view

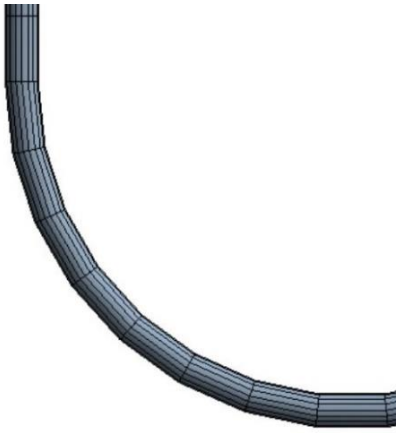

Figure 2-8: Mesh-axial view

In ANSYS meshing, face size represents the size of surface meshing while size represents the size of volume meshing. For meshing PHPs, the Max face size and the Min size are both set up as 0.1mm to determine the radial-direction mesh size. The length of the sweeping method is set to 1mm to determine the axial-direction mesh size. Next, the meshing process was recorded to make sure the mesh faces match around all the interfaces. In addition, all mesh values were

parameterized in ANSYS workbench so that meshes with different setups can be easily obtained. Lastly, in order for the ANSYS workbench to save the mesh parameters, a FLUENT module should be connected to the mesh module in the ANSYS workbench.

2.3.3 Fine mesh vs coarse mesh

With a fine mesh pair of 0.1 mm for the mesh size in the axial direction and 0.05 mm in the radial direction with 5 inflation layers and 0.05 mm total thickness for inflation layers, thin films can be clearly observed. In addition, clearer vapor plugs, liquid slugs and bubble formations can be seen as shown in Figure 2-9. Nevertheless, a fine mesh is computationally intense for a large geometry.

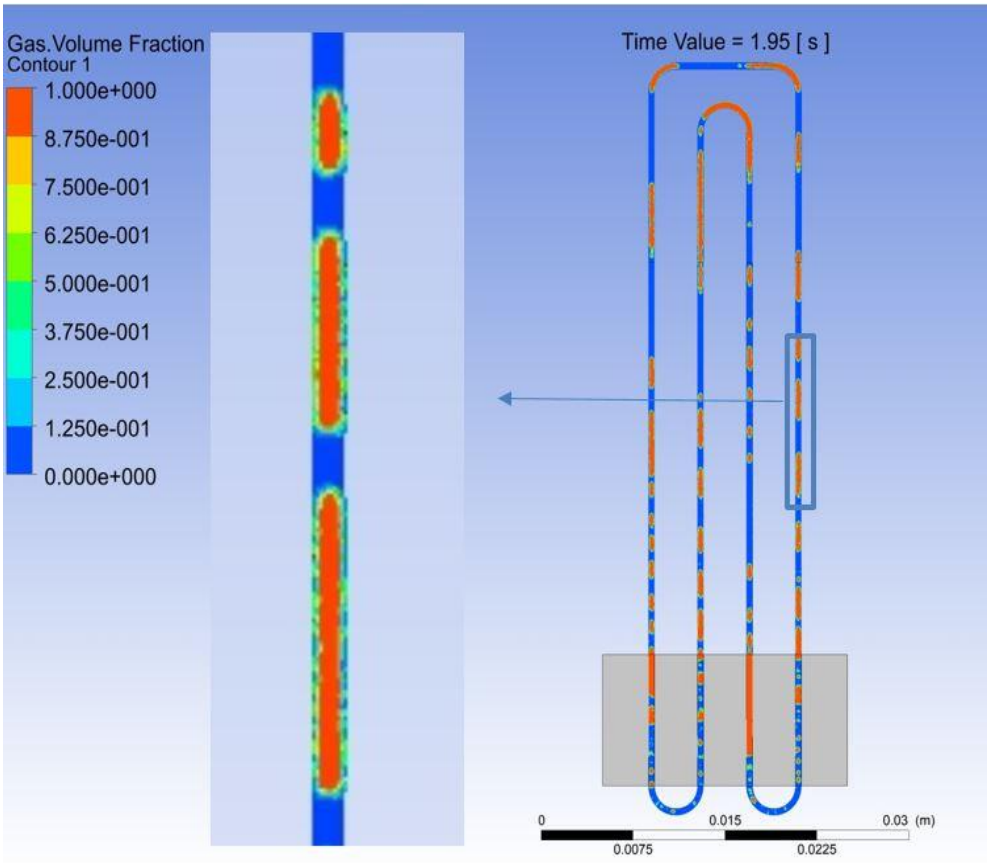

Figure 2-9: VOF plot with fine mesh

Lee model frequencies were adjusted in order to minimize the number of mesh elements. With the same Lee model frequency, a smaller mesh size produces a larger mass transfer rate. By appropriately adjusting the Lee model frequencies, the model with a coarse mesh shows the same results as that with a fine mesh. Much more will be discussed regarding the Lee model below. At this point it is just worth mentioning the importance of adjusting the Lee model frequencies when changing the mesh size until the two cases give the same mass transfer rate in the evaporator and condenser sections. Because our goal is first to validate the model against

experimental data that was generated in a rather large geometry, the mesh size needs to be relatively coarse to limit the number of mesh elements.

2.3.4 Solid meshing

The copper block around the evaporator was initially meshed and modeled due to its importance for thermally connecting neighboring tubing. However, the meshes of the block added a significant number of nodes to the existing mesh. Instead, virtual walls with shell conduction were added to provide the same effect as a copper block.

In Fluent, enabling virtual walls adds a thermal resistance outside of the wall. The effects of the thickness of the virtual wall in the evaporator region on temperature profile were investigated. Testing the influence of the thickness of the virtual wall with values of 0.3, 0.7 and 1.2 times the distance between tubes revealed that the thickness of the virtual wall has no effect on the temperature profile. The results suggest that the virtual wall does not really extend in the radial direction and that the thermal connection of the tubing wall in the axial direction is enough to smooth out the evaporator temperature. Although in the experimental hardware, neighboring tubes in the evaporator and condenser sections are thermally connected by a copper block, simulations show that there is actually no temperature difference between neighboring tubes. As a result, the thickness of the virtual wall was defined as 0.7 times the distance between tubes, but it can be defined to any reasonable value.

2.4 Material properties

Several properties of the fluid are of interest for a PHP model: density, specific heat, thermal conductivity, viscosity, contact angle, and latent heat.

For PHP modeling, the properties of liquid helium are set as a function of only temperature. In contrast, the properties of gaseous helium are set as a function of both temperature and pressure. The temperature ranges for properties of both gaseous helium and liquid helium are between 2 K and 20 K. The temperature range is somewhat larger than the corresponding experimental data but allows the model to calculate the properties via its iteration process.

Lastly, the properties of the two-phase mixture of helium is set as a function of both the properties of liquid helium and the properties of gaseous helium based on the VOF model.

2.4.1 Liquid helium properties

The properties for liquid helium are defined only as a function of temperature and the liquid is assumed to be incompressible. Moreover, when defining the properties of liquid helium in Fluent, they can only be defined in the mathematical form of piecewise-polynomial functions. Furthermore, usage of polynomial equations is limited in Fluent: only three ranges are allowed, and the maximum number of coefficients is 8. Properties of liquid helium are difficult to fit for the entire operating range. Thus, the temperature was divided into three ranges for all properties.

Additionally, property fits tend to become more inaccurate around the critical point, but the usual operating conditions of a helium PHP will not exceed the critical point. As a result, the overall error of the simulation is small. Furthermore, the specific heat of the liquid helium has a huge spike around the critical temperature and is difficult to fit around this area. A smoothing technique was performed around this area by averaging the peak to the neighboring area.

The fitting equations for liquid helium density are as follows:

$If\ 0K < T < 5.19K$

$$\rho = -2819.01 + 5415.12T - 4057.26T^2 + 1598.36T^3 - 349.525T^4 + 40.2272T^5 - 1.90564T^6 \quad (2\text{-}1)$$

$If\ 5.19K < T < 5.195K$

$$\rho = 41795.57 - 8036.69T \quad (2\text{-}2)$$

$If\ 5.195K < T < 20K$

$$\rho = 1105.21 - 548.42T + 112.74T^2 - 12.0871T^3 + 0.710936T^4 - 0.0217583T^5 + 0.000271052T^6 \quad (2\text{-}3)$$

All the other property fitting equations for liquid helium are attached in Appendix 14.1.

2.4.2 Gaseous helium properties

Some of the properties of gaseous helium are considered both a function of temperature and pressure while other properties are considered only a function of temperature in the model. Specifically, both density and enthalpy of helium gas are considered a function of both temperature and pressure. Programming the density of gaseous helium as both a function of temperature and pressure is crucial for the system mass to balance. On the other hand, viscosity, specific heat and thermal conductivity are considered as a function of only temperature.

Fluent uses a UDF (user defined function) to set the properties of helium gas because the UDF allows the properties to be set as both a function of temperature and pressure. Additionally,

Fluent only allows one phase to be defined using a UDF. Hence, the vapor state is chosen to be the fluid defined both by temperature and pressure.

A UDF is a function that is written by users to enhance the numerical algorithm of Fluent. A UDF uses the C++ language coupled with special functions to define cell information. Using a UDF, properties can be defined both as a function of temperature and pressure.

The UDFs for the gas properties such as density and enthalpy were obtained by fitting each property as both a function of temperature and pressure when the temperature and pressure pairs are in the superheated vapor region. Properties collapse into just a function of temperature when the temperature and pressure pairs are on the saturation line. Furthermore, when the temperature of the fluid is increased higher than the critical temperature, the pressure of fluid is set to remain at the critical pressure. As a result, properties were determined along the critical pressure line when the temperature of the fluid is above the critical temperature.

The fitting method for the helium properties is as follows. First, data points of temperature and pressure pairs are generated in MATLAB as shown in Figure 2-10. Secondly, the data pairs are imported into EES to calculate the properties. Thirdly, the data points of temperature and pressure as well as the corresponding properties were imported back into MATLAB to generate the fit equations as shown in Figure 2-11.

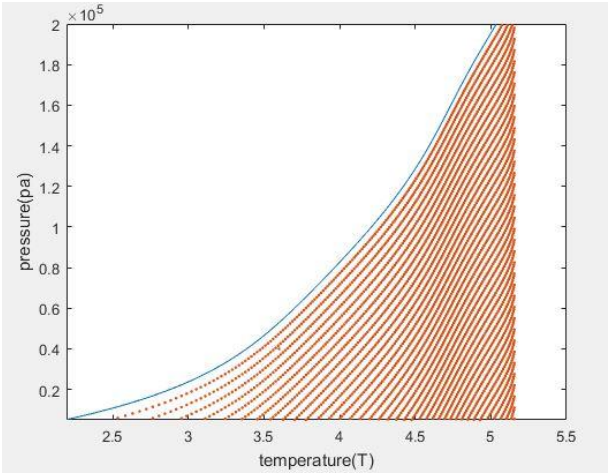

Figure 2-10: Temperature and pressure pairs under saturation line

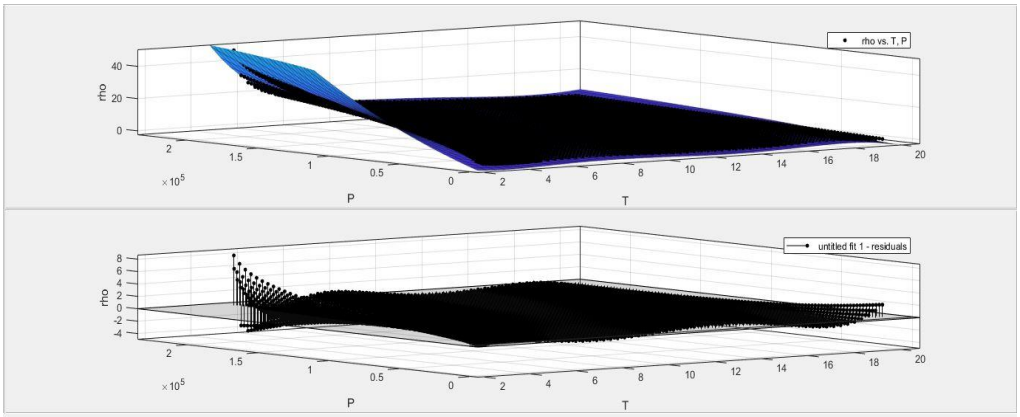

Figure 2-11: Polynomial fit and residual of fitting in MATLAB for density of helium vapor

The density fitting equations for helium gas as both a function of temperature and pressure are as follows:

$$If\ T > 1.84826 + 0.000073856P - 1.41996\mathrm{e}^{-9}P^2 + 1.74229\mathrm{e}^{-14}P^3 - 1.16493\mathrm{e}^{-19}P^4 + 3.92289\mathrm{e}^{-25}P^5 - 5.20498\mathrm{e}^{-31}P^6$$

$$\rho = -1.702 + 1.072T + 0.0004251P - 0.212\,T^2 - 0.0001344TP + 1.669\mathrm{e}^{-9}P^2 + 0.0158T^3 + 1.843\mathrm{e}^{-5}T^2P + -4.429\mathrm{e}^{-10}TP^2 + 9.402\mathrm{e}^{-16}P^3 + -0.0003884T^4 + -1.115\mathrm{e}^{-6}T^3P + 3.985\mathrm{e}^{-11}T^2P^2 + -4.921\mathrm{e}^{-16}TP^3 + 1.168\mathrm{e}^{-20}P^3 + 2.412\mathrm{e}^{-8}T^4P + -1.097\mathrm{e}^{-12}T^3P^2 + 1.807\mathrm{e}^{-17}T^2P^3 - 7.272\mathrm{e}^{-23}TP^4 + -1.86\mathrm{e}^{-26}P^5) \quad (2\text{-}4)$$

$$If\ T < 1.84826 + 0.000073856\mathrm{P} - 1.41996\mathrm{e}^{-9}P^2 + 1.74229\mathrm{e}^{-14}P^3 - 1.16493\mathrm{e}^{-19}P^4 + 3.92289\mathrm{e}^{-25}P^5 - 5.20498\mathrm{e}^{-31}P^6$$

$$\rho = 0.498078 + 0.000164067T - 7.54221\mathrm{e}^{-11}T^2 - 1.45519\mathrm{e}^{-16}T^3 + 6.63976\mathrm{e}^{-21}T^4 \quad (2\text{-}5)$$

Here ρ represents density, T represents temperature, and P represents pressure.

All the fitting equations for other properties of helium gas are attached in Appendix 14.2 , and the code for UDF that writes properties of helium gas is attached in Appendix 14.4.

2.4.3 Mixture helium properties, contact angle and specific latent heat

The actual properties of two-phase flow used in the VOF model are a combination of liquid helium and gaseous helium. For density, it is calculated by incorporating the volume fraction of each cell as follows:

$$\rho = \sum \alpha_q \rho_q \qquad 2\text{-}6$$

where α represents the volume fraction, ρ represents density, and q represents phase. All other material properties for two-phase helium are calculated in a similar manner.

The contact angle determines how good the surface of the tubing is wetted. It is found that the maximum heat load will be limited if the wetting ability of the fluid is insufficient because the vapor does not transfer heat as efficiently as liquid. Fortunately, helium has very good wetting ability. Although the exact value of the contact angle is uncertain and dynamic, a contact angle of 174 degrees for liquid Helium is an appropriate value.

The enthalpy of liquid helium is defined by a temperature integral of the heat capacity, with respect to a reference temperature. The enthalpy of gaseous helium is defined by both the temperature and pressure. As a result, to obtain the correct specific latent heat, the enthalpy of the vapor must use the same reference point as the liquid.

2.4.4 Copper block properties

The copper block is the only solid material that is modelled in the current model. Material properties such as density, specific heat, and thermal conductivity need to be set up in Fluent and all the material properties are set up as constant values because the result of the PHP model is not

sensitive to the material properties of the copper. Consequently, density, specific heat, and thermal conductivity are fixed at their values at $4K$.

2.5 General Setting

2.5.1 Physical settings

The implicit body force is turned on and the gravity value in the y direction is set as -9.8 m/s^2 to simulate gravity. The viscous model is chosen as laminar, and viscous heating is selected to model heat produced by shear forces. Sharp, as opposed to dispersed, is used for interface modelling due to the presence of a distinct interface between the liquid and vapor regions inside the pulsating heat pipes.

2.5.2 Boundary conditions

A no-slip boundary condition is imposed on the wall. In order to apply heat and take away heat, a constant heat flux is set on the evaporator and a constant temperature is set on the condenser area. A zero-heat flux boundary condition is specified along the walls of the adiabatic section to simulate an insulation condition.

2.5.3 Operating conditions

Operating conditions are set up in Fluent to help improve the simulation efficiency. Fluent recommends setting up the operating temperature and operating densities to simulate flow with buoyancy. Hence, the operating temperature is set as 4 K and the operating density is set as 1.225kg/m^3. Moreover, the algorithm of the Lee model uses gauge pressure to calculate the saturation temperature. As a result, it is important to set the operating pressure as zero to ensure

that Fluent uses the absolute pressure in order to calculate the saturation temperature and properties.

2.6 Mathematical model

2.6.1 Main governing equations

For a numerical model with thermal consideration, there are three sets of equations that must be included: conservation of mass, conservation of momentum, and conservation of energy.

In addition, because the flow is a two-phase flow, all three governing equations need to take that into consideration. A two-phase model must be used to distinguish liquid and vapor phase. The VOF model is chosen as the two-phase flow model because the VOF model is most suitable for modelling slug/plug flow.

For a two phase VOF model the mass conservation equation for the secondary phase q is as follows:

$$\frac{1}{\rho_q}\left[\frac{\partial}{\partial t}\left(\alpha_q \rho_q\right) + \nabla \cdot \left(\alpha_q \rho_q \overrightarrow{v_q}\right) = S_{\alpha_q} + \sum_{p=1}^{n}\left(\dot{m}_{pq} - \dot{m}_{qp}\right)\right] \qquad 2\text{-}7$$

Where ρ, α, v, t represent the density, volume fraction, velocity, and time, respectively. The mass source term S_{α_q} is zero for the current model since PHPs are closed systems. The term $\dot{m}_{pq}$ represents the mass transfer rate from phase p to phase q.

After the mass conservation equation for the secondary phase was calculated, the same equation for the primary phase p is solved by the following constraint:

$$\sum_{a=1}^{n} \alpha_q = 1 \qquad 2\text{-}8$$

Liquid helium is chosen to be the primary phase while gaseous helium is chosen to be the secondary phase. A convergence issue arises if the primary phase and secondary phase are chosen the other way around. The volume fraction of 0 or 1 for the primary phase indicates saturated vapor or liquid respectively while the volume fraction between 0 and 1 of the primary phase indicates an interface inside of the cells.

Only one momentum equation needs to be solved for the VOF model, and the velocity is shared among the phases:

$$\frac{\partial}{\partial t}(\rho\vec{v}) + \nabla \cdot (\rho\vec{v}\vec{v}) = -\nabla p + \nabla \cdot [\mu(\nabla\vec{v} + \nabla\vec{v}^T)] + \rho\vec{g} + \vec{F} \qquad 2\text{-}9$$

Here p represents pressure, μ represents dynamic viscosity, and g represents gravity. The force F represents the extra force exert on the cell and surface tension is such an extra force F in PHP modelling. Due to the small size of the capillary tubes, the surface tension effect is comparable to the gravity effect.

Finally, the energy equation that is shared among phases is given by:

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\vec{v}(\rho E + p)) = \nabla \cdot \left[k_{eff}\nabla T\right] + S_h \qquad 2\text{-}10$$

Here E represents energy, and T represents temperature. In addition, k_{eff} represents the thermal conductivity and ρ represents density, both are shared by the phases. The source term S_h contains any volumetric heat sources, which is zero for current model because a PHP is a closed system.

It is worth noting that in the energy equation 2-10, E and T are mass-averaged as in the following equation:

$$E = \frac{\sum_{q=1}^{n} \alpha_q \rho_q E_q}{\sum_{q=1}^{n} \alpha_q \rho_q} \quad 2\text{-}11$$

Where E_q is based on the specific heat of that phase and the shared temperature.

2.6.2 Additional equations

As mentioned above, a surface tension force term must be added to the momentum equation. The continuum surface force (CSF) model was used to simulate the surface tension model. Wall adhesion is modelled for adhesive forces between the wall and fluid.

The term n represents the surface normal, and α_q is the volume fraction of the q phase.

$$n = \nabla \alpha_q \quad 2\text{-}12$$

A unit normal is defined as:

$$\hat{n} = \frac{n}{|n|} \quad 2\text{-}13$$

The curvature k, is defined in terms of the divergence of the unit normal, $\hat{n}$:

$$k = \nabla \cdot \hat{n} \quad 2\text{-}14$$

The wall adhesion force with the contact angle θ_w is defined using the surface normal:

$$\hat{n} = \hat{n}_w cos\theta_w + \hat{t}_w sin\theta_w \quad 2\text{-}15$$

Here $\hat{n}_w$ and $\hat{t}_w$ are the unit vectors that are normal and tangential to the wall.

The surface tension force F_{vol} is added to the momentum equation to model the surface tension effect:

$$F_{vol} = \sigma_{ij} \frac{\rho k_i \nabla \alpha_i}{\frac{1}{2}(\rho_i + \rho_j)} \quad \text{2-16}$$

Here ρ is the volume-averaged density and σ_{ij} is the surface tension coefficient.

2.7 Lee model

2.7.1 Definition

Mass transfer between the vapor and liquid phases is crucial for the PHP simulation. Fluent has a built-in evaporation-condensation model called the Lee model. The Lee model calculates the evaporation/condensation rate based on the cell temperature, pressure, liquid or vapour density (ρ), and liquid or vapour volume fraction (α). The algorithm of Lee model is defined by the following equations:

If $T > T_{sat}$

$$m_{lv} = coeff_{eva} * \alpha_l \rho_l \frac{(T - T_{sat})}{T_{sat}} \quad \text{2-17}$$

If $T < T_{sat}$

$$m_{lv} = coeff_{cond} * \alpha_v \rho_v \frac{(T - T_{sat})}{T_{sat}} \quad \text{2-18}$$

The coefficients in equations 2-17 and 2-18 characterize the rate at which the mass changes phase and are subject to considerable adjustment in order for the model to satisfy the mass and energy balance. The saturation temperature of a cell is determined by the saturation pressure of the cell by the following equation:

$$T_{sat} = 1.84826 + 0.000073856P_{sat} - 1.41996\mathrm{e}^{-9}{P_{sat}}^{2} + 1.74229\mathrm{e}^{-14}{P_{sat}}^{3} - 1.16493\mathrm{e}^{-19}{P_{sat}}^{4} + 3.92289\mathrm{e}^{-25}{P_{sat}}^{5} - 5.20498\mathrm{e}^{-31}{P_{sat}}^{6} \quad \text{2-19}$$

The sign convention in equations 2-17 and 2-18 is such that mass transfer is positive when liquid becomes vapor. At locations where the liquid reaches its boiling temperature, the liquid begins to evaporate. At places where the liquid boils, vapour slugs form. Liquid condenses in the condenser region where the local temperature falls below the saturation temperature associated with the local pressure.

2.7.2 Lee model frequency for evaporation

Lee model frequencies dictate how fast evaporation and condensation happen. It is found in the simulation indeed that the mass transfer rate increases with increasing Lee model frequency. Energy transfer results from the mass transfer and is determined by the multiplication of the mass transfer rate and the latent heat. Because the evaporation and condensation rate change with geometry and thermal conditions, there are many factors that influence the Lee model frequencies: the area of condensation section, the area of evaporation section, the condensation boundary temperature, and the applied heat on the evaporator.

As a result, an important part of PHP simulation is to figure out the correct Lee model frequencies to apply for different geometry and thermal conditions. In order to do so, the influence of the geometry and thermal conditions on the Lee model frequency are investigated. Meanwhile because evaporation frequencies and condensation frequencies are inherently linked, only evaporator frequencies are analyzed closely as the representation for both.

For the geometry influence, it is found that evaporator frequencies are related to the heat flux applied to the evaporator instead of heat applied to the evaporator. In other words, with the same heat applied to the evaporator, evaporator frequencies are inversely linked to the surface area of evaporator.

For the thermal influences, it has been found that evaporator frequencies influence both the adiabatic temperature and the amount of heat transferred by the evaporator. Firstly, the adiabatic temperature decreases with the increasing evaporator frequency. Secondly, the amount of heat transferred by the evaporator increases with increasing evaporator frequency. However, there is a plateau after which the heat transferred by the evaporator will stay the same even with a higher evaporator frequency.

Combining both the geometry and thermal influences, one can draw the conclusion that the evaporator frequency needs to be expressed as a function of heat flux in order to adapt to different geometry and thermal conditions. The detailed fitting process is described in Chapter 4.1.

When the evaporator frequency is higher, the evaporation rate in the evaporator is higher. However, there is a limited amount of liquid in the evaporator that is available to be evaporated. Consequently, the evaporator frequency is limited.

2.7.3 Lee model frequency for condensation

The ratio of the evaporation frequency to the condensation frequency must be adjusted to ensure a mass balance for different boundary conditions. A UDF has been written to calculate the condenser frequency based on the evaporator frequency and system condition. The UDF can adjust the condenser frequency at the end of every time step based on the system conditions. To provide an overall mass balance, it is assumed by the UDF that the evaporation rate at the evaporator is equal to the condensation rate in the condenser. The UDF performs successfully with a 50 mm total length and 2 turn geometry. However, with the large geometry, the system has a divergence problem. The UDF is attached in Appendix 14.5.

Instead, with the large geometry, the strategy for determining the condensation frequency is to calculate it based on the same principle of mass balance but only using the initial condition. Instead of changing it for every time step, the condensation frequency is only calculated at the beginning of the run.

2.8 Numerical solver

The type of the solver decides in which order the governing equations are solved. There are two types of solvers: pressure-based and density based. The pressure-based solver is usually used for incompressible low-speed flow while the density-based solver is typically used for compressible high-speed flow. With the density-base solver, pressure is obtained from the

equation of state while density is obtained from the continuity equation. On the other hand, with the pressure-based solver, pressure is obtained by solving a pressure equation which is obtained from manipulating the continuity and momentum equations.

The VOF model is only compatible with the pressure-based solver. As a result, the pressure-based solver was chosen. The steps for each iteration of the pressure-based solver are as follows:

1. Update the flow properties.

2. Solve the momentum equations.

3. Solve the pressure correction equation.

4. Correct the mass flux, pressure, and velocity fields on each face of the unit cell using the pressure correction equation.

5. Solve the energy equation.

6. Check for convergence of the equations

There are five methods to manipulate the continuity and momentum equations to obtain the pressure field. Such methods are called pressure-velocity coupling method. There are: SIMPLE, SIMPLEC, PISO, Fractional Step, and Coupled.

After trial and error, SIMPLEC is chosen as the Pressure-velocity coupling method. Even though Pressure-Implicit with Splitting of Operators (PISO) is recommended for VOF models and works for small geometries, SIMPLEC is selected in order to obtain convergence for the large geometries. Here ‘small’ geometries include the following combinations of the overall

length and number of turns: 50 mm 2 turns, 200 mm 2 turns, 500 mm 2 turns, 200 mm 5 turns. In contrast, the ‘large’ geometry case is defined by 500 mm 21 turns.

2.9 Solution method

In the numerical study, discretization methods need to be selected for the governing equations, and for variables such as density and pressure. The URFs for the discretization method were selected to determine how fast variables updated with each iteration. A gradient scheme must be selected to calculate gradients. Because the PHP model is a transient study, a discretization method is also needed for time. Lastly, for two-phase flow, an interpolation scheme also must be selected to visualize the two-phase flow.

Both the discretization methods and interpolation scheme are selected carefully after trial and error and are justified and demonstrated in this chapter.

2.9.1 Discretization methods

The mass conservation equation is chosen to be solved explicitly using the following scheme because an explicit scheme is more stable for two-phase flow with surface tension involved:

$$\frac{{\alpha_q}^{n+1}{\rho_q}^{n+1}}{\Delta t}V + \sum_f \left(\rho_q {U_f}^n {\alpha_{q,f}}^n\right) = [\sum_{p=1}^{n}\left(\dot{m}_{pq} - \dot{m}_{qp}\right) + S_{\alpha_q}]V \qquad 2\text{-}20$$

Here n+1, and n represent the current and previous time step, respectively, f represents the face value of a cell, U represents volume flux, and V represents cell volume.

The momentum equation is solved using the QUICK scheme. The QUICK scheme can be used only for quadrilateral and hexahedron faces. Also it is more accurate for the structured mesh than the unstructured mesh. The simulation of the PHP uses a hexahedron structured mesh, consequently QUICK is well suited for the model.

The QUICK scheme is explained by equation 2-21. Also, the face values of e, p, and w are illustrated in Figure 2-12 as connecting control volumes in a computational domain.

$$\emptyset_e = \theta\left[\frac{S_d}{S_c + S_d}\emptyset_p + \frac{S_c}{S_c + S_d}\emptyset_E\right] + (1-\theta)\left[\frac{S_u + 2S_c}{S_u + S_c}\emptyset_p - \frac{S_c}{S_u + S_c}\emptyset_w\right] \qquad 2\text{-}21$$

Where Fluent uses a solution-based value of θ.

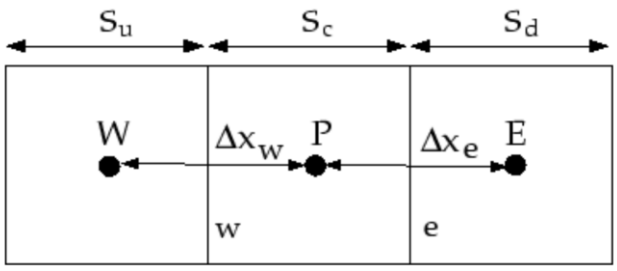

Figure 2-12: One-Dimensional Control Volume

The energy equation is solved using a first order upwind scheme. First order upwind simply means that the face value is set equal to the cell-center value of the upstream cell.

The pressure interpolation scheme is set to body-force-weighted. There are two reasons for that. Firstly, the body-force weighted scheme works well when the body force is known a-priori in the calculation. In the PHP model, the only body force is gravity, and the value of gravity is known. Secondly, the body-force-weighted scheme also takes into account the

discontinuity of the pressure gradients for flow with rapidly changing densities, which is suitable for two-phase flow.

The density interpolation scheme is set to second order upwind. The face value of $\emptyset_{f,sou}$ is calculated using the following expression:

$$\emptyset_{f,sou} = \emptyset + \nabla\emptyset \cdot \vec{r} \qquad 2\text{-}22$$

Here $\emptyset$ represents the upstream cell centre value, $\nabla\emptyset$ represents the gradient from the upstream cell, and $\vec{r}$ represents the displacement vector from the upstream cell centroid to the face centroid.

2.9.2 URF

Under-relaxation factors (URF) are used to control the speed for updating the computed variables at each iteration. Large URFs cause solution divergence while small URFs increase the computational time. Moreover, the sensitivities of the residual to different URFs are different case by case. As a result, the sensitivity of the residual to different URFs was investigated for the PHP simulation. The results show that convergence is sensitive to the URF for pressure, density and vaporization mass. However, it is not sensitive to the URF of body force, momentum and energy. Usually, the URF of the density is set at a value of 0.1 and the URF of the vaporization mass is set smaller than 0.3. Furthermore, reducing the URF of the pressure is an effective method for achieving convergence for the larger geometries. Lastly, the URFs for the body forces, momentum and energy are set to 1, 0.7, and 0.3, respectively.

2.9.3 Gradient scheme

Gradients are useful both in constructing a value at a cell face and for computing derivatives. There are three options in Fluent for computing gradients: the Gauss Cell-Based method, Green-Gauss Node-Based, and Least Square Cell-based. The Least Square Cell-based method is less expensive to compute compared to the Green-Gauss Node-Based method, and more accurate than the Green-Gauss Cell-Based method. Thus, the Least Square Cell-based method is chosen for the current model.

2.9.4 The discretization of the time

The time discretization approach chosen for the PHP model is first order implicit. For the implicit time scheme, $\emptyset$ is calculated using the following expression:

$$\frac{\emptyset^{n+1} - \emptyset^{n}}{\Delta t} = F(\emptyset^{n+1}) \qquad 2\text{-}23$$

Where the function F incorporates any spatial discretization.

2.9.5 Interpretation scheme

The interpretation scheme near the surface between the liquid and vapor for the VOF model must be carefully chosen to represent the actual interface during flow visualization. A geometric reconstruction scheme was chosen because it is highly similar to an actual interface. Figure 2-13 shows an actual interface surface while Figure 2-14 shows how the interface is represented with the geometric reconstruction scheme.

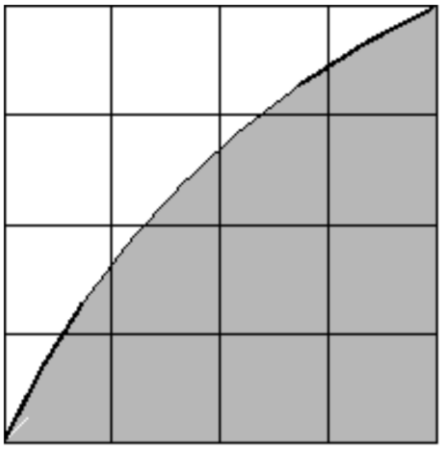

Figure 2-13: Actual interface surface

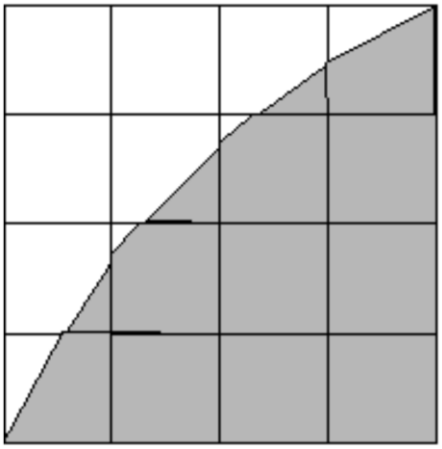

Figure 2-14: Interface represented by geometric reconstruction

2.10 Initialization

Initial guess values of the temperature, pressure, volume fraction and velocity need to be carefully selected for solution convergence.

First, the initial temperature guesses are different for different sections of the PHP. Cell temperatures on the evaporator wall are initialized as the guessed evaporator wall temperature, cell temperatures in the adiabatic region are initialized as the guessed adiabatic temperature, and cell temperatures on the condenser wall are initialized as the condenser wall temperature. The guessed adiabatic cell temperature is the average of the evaporator wall temperature and condensation wall temperature.

Second, cell temperatures of the evaporator and condenser on the wall are set differently from cell temperatures of the evaporator and condenser fluid inside of the tubes. Specifically, the cell temperatures of the evaporator/condenser fluid inside of the tubes are set as the average temperature of the adiabatic temperature and the evaporator/condenser wall temperature. The

reason for this setting is that there is an expected temperature difference between wall temperatures and fluid temperatures due to the wall heat flux.

Third, the cell pressure values are initialized as the saturation pressure of the guessed adiabatic temperature. The volume fraction of the cells is set according to the fill ratio of the PHP. After some time, slugs and plugs are formed which means most cells have a volume fraction of 0 or 1.

Fourth, the initial guess values of the velocities in each cell are initialized as 1 m/s and the directions at the different tubing areas are initialized in a way that it forms a clockwise circulation for large PHP geometry. The reason behind this is to facilitate the system reaching steady state faster. On top of that, in experiments from both Fonseca [1] and Li, Li, & Xu [9], heat was added gradually and subsequently circulation was already formed before a new heat load was added. However, when the PHP geometry is equal to or smaller than the 10 turn – 200 mm geometry, such a push slows down the simulation. As a result, when the PHP geometry is small, the initial guess values of velocities are set to zero.

2.11 Time step and iteration

Unsteady time conditions are used to model the transient problem. The simulation time is usually set to 30 seconds which is long enough for pulsating heat pipes to reach stable operation.

Time steps are usually set to 0.0025-0.005 seconds and the solution is iterated between 70-400 times in each time step to ensure convergence. When the time steps become too small, divergence occurs. Time steps are also limited by the Courant number. The Courant number defines how many fluids cells flow through one mesh cell during one time step. As a default,

Fluent sets the maximum limit for the courant number to 250. Ideally, the Courant number should be about 1-2. However, to ensure convergence, the Courant number was limited to about 20 in the radial direction while remaining 1-2 in the axial direction.

2.12 Convergence

Convergence is judged by monitoring a few key variables such as mass, temperature, applied heat. The total mass in the domain was monitored to judge the mass convergence and the changes were controlled to within 5 percent during 30 second simulation time. The temperature vs iteration plots were also plotted to judge convergence. Plateaus were observed as a sign of convergence as shown in Figure 2-15 for every time step.

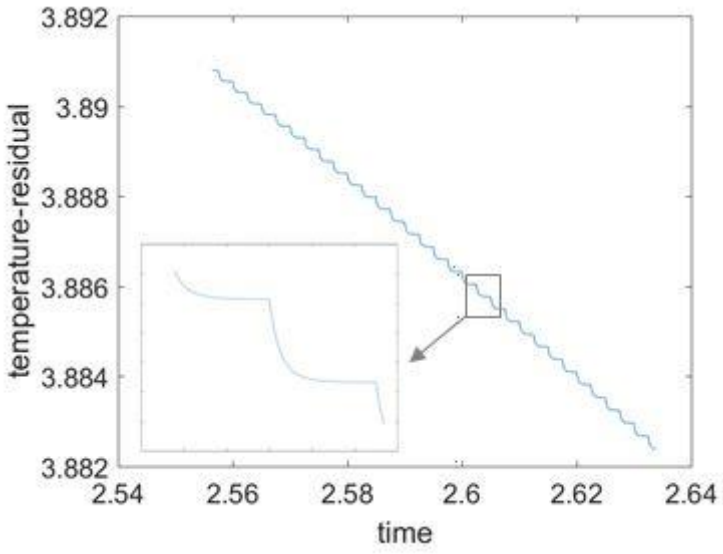

Figure 2-15: Temperature plotted vs time

2.13 Mass balance

Ideally, the total system mass should be constant throughout the run. Realistically, the total system mass will have small changes during the simulation time due to the numerical method. The mass will change drastically if the density is not both a function of temperature and pressure. Additionally, an unmatched evaporation and condensation rate in the system will cause an unbalanced mass for a system that reaches steady state. Thus, the evaporation and condensation frequencies are adjusted case by case to ensure a balanced system. Finally, it is easier to achieve good mass balance with a small heat load and small Lee model frequencies.

2.14 Data saving

Typically, data files retain the information of volume fraction, temperature, pressure and velocity for each cell value and some face values over the whole computational domain.

The data file can be huge for the large geometry. For example, for the 5 turn 500 mm case, every data file is 0.1G in size. Consequently, data files were usually saved every 100 iterations.

Fluent saves 120 data files when simulating 30 seconds of PHP operation. As a result, a 2.4 G file size was generated in such case. The HPC has a 100G data size limitation per user, which also limits how many runs can be simulated at the same time in a day.

In addition, for this project, the mass transfer rate and Reynolds number are interesting parameters to monitor as well. A C-data file is another kind of data file that is used to store cell information. A C-data file can choose which quantities are to be saved while a data file cannot. Thus, the C-data file was saved which adds the information of mass transfer rate and Reynold

number when necessary. Lastly, the format of the C-data file was chosen with a CFD-post compatible format because most post-processing was performed with CFD-post software.

2.15 Monitors for measuring quantities

Data files save all the cell information for the whole domain and are not saved every time step. In contrast, some interesting singular cell information or some information related to a mathematical combination of a cluster of cells can be saved for every time step or even every iteration for the purpose of validating or evaluating a numerical PHP. Such information is stored in monitor files.

There are several quantities that are of interest to save in monitor files: mass, volume fraction, temperature, heat flux/heat, velocity, pressure, and mass transfer rate/total mass transfer. There are also five types of quantity monitors in Fluent to organize numerical cell information in different ways: wall-averaged monitor, wall-integral monitor, volume-averaged monitor, volume-integral monitor and point monitor. As a result, different types of monitors that measure different quantities can be created. The rest of this section discusses the types of monitors that were chosen for each quantity of interest and what purpose they serve.

For mass monitors, the volume-integral mass for each section and for the whole PHP system are monitored to ensure mass balance for the system.

Volume fraction monitors are used to save the volume-averaged volume fraction for each section and the whole PHP system to obtain the time dependent sectional volume fraction in the PHP.

For temperature monitors, the area-averaged wall temperature as well as the volume-averaged temperature of the fluid in the evaporator, condenser, and adiabatic sections were tracked to monitor the thermal performance of the PHP.

For heat flux/heat monitors, the area-integral wall heat for each section and the whole PHP system is monitored to check the energy balance for the system. Also, the area-averaged wall heat flux is monitored to obtain the time dependent sectional heat flux.

For velocity monitors, the point velocity monitor is set up in the middle of different sections for each tube to track velocity changes in the PHPs. It is also worth mentioning that all of the velocity point monitors are written in such a way that they will adjust automatically with different length and turn number by using the scheme language in Fluent.

For pressure monitors, the volume-averaged pressure at different sections is monitored to obtain the time dependence of each sectional pressure.

For mass transfer rate monitors, the volume-integral total mass transfer for each section and the whole PHP system are monitored to assist the balance between evaporation and condensation. Also, the volume-averaged mass transfer rate for each section is monitored to obtain the time dependence of the sectional mass transfer rate.

Lastly, a custom function can be written for a combination of quantities for the purpose of monitoring. The latent heat is of interest for PHP modeling and thus is written as the following equation:

$$h_{fg} = m_{lv}(h_g - h_f) \qquad 2\text{-}24$$

where m_{lv} is the mass transfer rate, h_g is the enthalpy of gaseous helium and h_f is the enthalpy of liquid helium.

Volume-Integral monitors for the value of h_{fg} were created to monitor the total latent heat for each section of the PHP.

2.16 Journal file for settings and data saving

The various cases explored in Fluent are generated on a desktop computer, by selecting the necessary settings via a graphic interface. However, each of the Fluent cases for the PHP model are run by an HPC cluster because its complexity calls for higher computing power. Fluent cases run by the UW-Madison HPC cluster established all the necessary settings by writing codes in a setup text file called a Journal file. All the setups were written in the form of a journal file also for convenient changes. A typical journal file for PHP modeling is attached in Appendix 14.3.

'Scheme' is a language used in Fluent for defining variables, defining the relationship between different variables and for exchanging data between the journal code and the UDF code. In order to easily set up Fluent for various other geometries, the scheme language is also used in a journal file. The journal file with scheme language can automatically update wall thickness, heat flux and other related parameters according to the input geometry and thermal operating information.

3 High performance computing and simulation time

A typical desktop PC has 4 CPUs while the HTC System at UW-Madison has 8 CPUs and an HPC system usually uses 80 CPUs. Furthermore, the HPC system can instantly write files during a simulation while the HTC system can only show files after the simulation is finished. After trying both options, the HPC system was found to suit this project better than the HTC system.

3.1 Simulation capacity for the HPC system

In the current setup with the HPC system at the UW-Madison CHPC Center, 10 simulations can be run simultaneously. For the small geometries such as 50 mm 2 turns, 200 mm 2 turns, 500 mm 2 turns, and 200 mm 5 turns, a 12 second simulation time can be achieved in a day with the HPC system. For the geometry using 500 mm and 21 turns, a few days or even a week is required.

3.2 Submitting a file for the HPC system

To use the HPC system, a submit file needs to be written to specify the name of the partition, simulation time, number of nodes, number of CPU per node, RAM per CPU. A typical submit file is attached in Appendix 14.6.

There are three types of partitions in the HPC system within the UW-Madison system. Int partition allows each simulation job to run for up to 30 minutes. These are usually used for testing simulation jobs instead of running full simulations. Univ2 partition allows each simulation job to run up to 7 days, but the waiting time for this partition is long. Pre partition allows each simulation job to run up to 24 hours and there is almost no waiting time for this partition. Consequently, pre partition is chosen for the current study for running full simulations.

The computational power is calculated using the following equation:

$$P = n_{node} \cdot n_{CPU} \cdot RAM \quad \text{3-1}$$

where n_{node} represents the number of computational nodes per CPU, n_{CPU} represents the number of CPUs, and RAM represents the random-access memory for each CPU.

The top limit of n_{node} in pre partition is 16. However, setting n_{node} as 16 limits the number of jobs that can be simulated at the same time. In addition, for a smaller geometry such as the 5-turn 100 mm, setting a value of n_{node} as 16 can even make the simulation run slower. As a result, in order to run 10 jobs at the same time and not compromise simulation speed, n_{node} is usually set as 4, On the other hand, when simulating a geometry as large as 21 turns and 500 mm, n_{node} is set as 16.

The top limits for n_{CPU} and RAM are 20 and 4 GB/cpu and are the options chosen for the current simulation.

3.3 Simulation procedure for the HPC system

A case file, a journal file, a UDF file, and a submit file were placed in the same folder to create a case file for each simulation. The residual information that is usually printed on the console using a PC is written as an output file in the HPC/HTC systems instead. The output file, monitor files, and data files are saved and updated simultaneously during the simulation.

4 Experimental data verification

4.1 Lee model frequency for evaporation fitting equation

One of the key parameters that was used to confirm the model's accuracy was the correct prediction of the evaporator temperature when given the applied heat and condenser temperature with a certain PHP geometry.

One of the difficulties of this method is finding the correct Lee model frequencies for different values of the heat load. The process of verifying the numerical PHP model included calibrating its adjustable parameters, specifically the Lee model frequency in the evaporator as a function of the heat flux applied to the evaporator, using Fonseca's helium data, and subsequently confirming the model's accuracy on a second set of data for a helium PHP with a different geometric configuration.

Because two sets of data are available for this process, the model uses the data from Fonseca [26] to make a fit of evaporator frequency vs. heat flux and subsequently confirms whether the fit is effective for predicting the data experimentally generated by Li, Li and Xu [9]. Li, Li & Xu successfully ran a helium-based PHP with 24 turns and an adiabatic length of 200 mm.

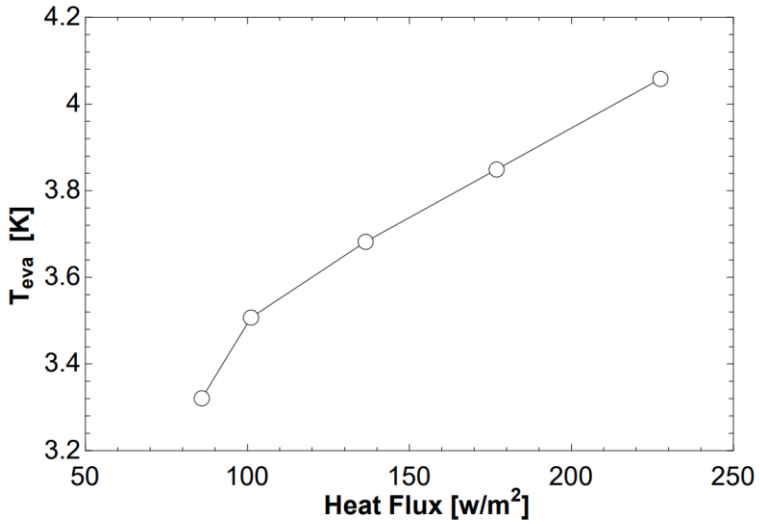

Figure 4-1: Experimental data from Fonseca [26]

Five of the data points from Fonseca [26] were chosen to obtain the evaporation frequency for the fit as shown in Figure 4-1. The corresponding evaporation frequency is plotted in Figure 4-2. The empirical fit developed for the evaporator frequency as a function of heat flux is as follows:

$$Eva_{freq} = -1.36137 + 0.224081 * heatflux - 0.00202189 * heatflux^2 + 0.00000715905 \quad 4\text{-}1$$

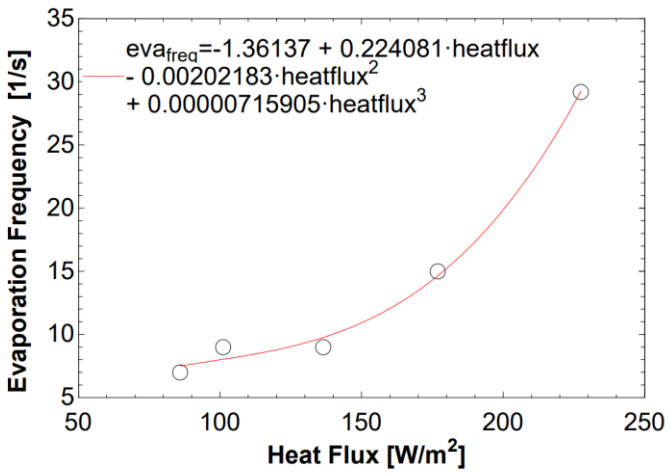

Figure 4-2: Evaporation Frequency vs Heat Flux

4.2 Comparison between experimental data with model results

Experimental data reveal that steady state conditions following a step increase in heat load require about 200 seconds. To simulate the same time in the numerical model would take many weeks. To avoid the relatively long computational time needed to model the entire time for the evaporator temperature to come to a steady-state value after changing the heat load, a guess value is used for the steady state evaporator temperature. The guess value is increased until a heat balance is observed. The evaporator temperatures for each heat flux were obtained using this method. Also, the temperature difference between the evaporator and condenser obtained from the model was compared with the experimental data. It can be observed from Figure 4-3 that the model predicts the experimental data well. The further comparison between predicted effective thermal conductivity and effective thermal conductivity that came from experimental data are in good agreement as well as shown in Figure 4-4.

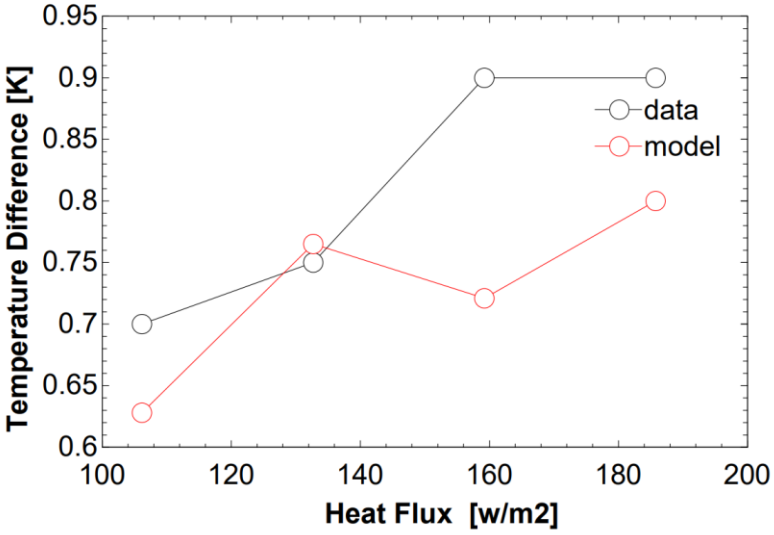

Figure 4-3: Temperature comparison of model predictions with data from Li, Li, Xu [9]

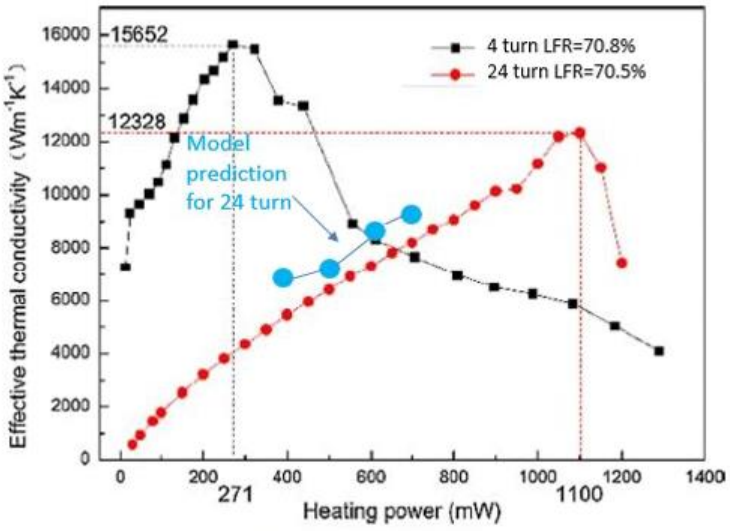

Figure 4-4: Effective thermal conductivity comparison of model predictions with data from Li, Li, Xu [5]

5 The choice of investigated plan

5.1 Using the model

Typically, in a helium PHP experiment [26] both temperature and pressure are measured. Two temperature sensors are put on the surface of the evaporator copper block and the condenser copper block to measure the average temperatures of the evaporator and condenser sections. Several temperature sensors are fixed at several points on the stainless-steel tubing of the adiabatic section to measure the local adiabatic temperature. One pressure sensor is located on the gas line that is connected to the adiabatic section of the PHP to measure system pressure of the PHP. The amount of temperature and pressure information coming from the experiment is thus limited and undetailed. However, with numerical modeling, detailed information of temperature and pressure distributions can be obtained. Moreover, experiments cannot measure any flow quantities that are inside of the PHP. Numerical modeling can describe flow quantities that are happening inside of the PHP such as fluid temperature, flow velocity, flow pattern, circulatory pattern, and pulsating frequencies. In addition, the numerical model can also determine the amount of evaporation and condensation that occurs inside of the evaporator/condenser. Furthermore, the start-up process can only be visualized in the numerical model. Finally, because of the difficulty of conducting helium PHP experiments, it is more convenient to use a working PHP numerical model to predict the thermal conductivities for PHPs of different geometries and fill ratios and PHPs working under different thermal conditions.

The geometry and thermal conditions of a helium PHP have unclear effects on all the quantities listed above. Nevertheless, understanding those effects is beneficial for understanding the working principle and the design of a helium PHP. As a result, a parametric study using the numerical model was constructed to understand those effects.

The detailed geometry and thermal conditions that were varied are the number of turns, total length, evaporator/condenser length, fill ratio, condenser temperature, and heat flux applied to the PHP. Geometric parameters such as turn number and total length were limited by the computational cost. The choices of fill ratio, condenser temperature, and heat flux were made by observing the experimental PHP results from Fonseca [26] and the experimental study by the Beijing group [9]. The length of the evaporator and condenser were chosen based on initial tests with the model.

Additionally, because the trends of each parameter's influence on the performance of the PHPs are of interest, three values of each parameter are used to explore the space. However, adjustments of the original plan were made along the way while testing the feasibility of the investigated plan.

5.2 The choice of total length and turn number

The total size of the PHP system being modeled is limited by the number of mesh elements and the computational time it takes to generate about 30 seconds of real time behavior. The number of mesh elements of a PHP is determined by both the total length and turn number. The simulation time increases almost linearly with the increasing size of the PHPs. As a result, in order to complete the simulation part of the study within a reasonable amount of time (~ 1 year), the optimum running time for each case should be limited to within a day. From previous simulations, it is found that a PHP simulation with 24 turns and an overall length of 200 mm takes about two days to complete. As a result, the upper limit of the total length for the investigated plan is 200 mm while the turn number is limited to 10.

5.3 The choice of evaporator/condenser length

For consistency purposes, the evaporator length is held constant as the total length varies. Because the performance of the PHP is influenced by the total applied heat, a change of evaporator length would change the total heat coming into the evaporator section. As a result, changing the evaporator length in this study would complicate the problem. Thus, although the total length of the investigated PHPs is fixed at 50 mm, 100 mm, or 200 mm, one might envision a constant percentage of the length being comprised of the evaporator and condenser sections. However, here the evaporator and condenser lengths are each held constant at 10 mm.

In the various explored models, the adiabatic length is always larger than zero and the evaporator length could have been chosen as 5 mm, 10 mm, or 20 mm. However, after conducting some initial testing, it was found that a 5 mm evaporator length generates heat that is too small to drive flow in the PHP while a 20 mm evaporation length generates so much heat that the system was overwhelmed. In conclusion, 10 mm was chosen as the evaporation length for the investigated plan.

5.4 The choice of heat flux/condenser temperature

Once the heat flux and condenser temperature is fixed, the evaporator temperature can be obtained through the same guessing method as described in Chapter 4.2.

Three levels of heat flux: 85.94, 136.5, and 227.5 w/m^2 were selected. Our evaporator frequency/heat flux correlation was derived from five heat flux levels: 85.94, 101.1, 136.5, 176.9, and 227.5w/m^2 from Fonseca [26]'s data, and to save simulation time, three values were picked from these five heat flux values. To stay within the valid zone where the correlation

works while saving simulation time, three levels of heat flux were selected within the range of 85.94 to 227.5w/m^2.

The condenser temperature is chosen as the same reported in Fonseca's experiment [26] corresponding to the load map of the cryocooler in the experiment. Except for the highest heat flux, the condenser temperature was lowered to keep the highest evaporation temperature under the critical temperature.

5.5 The choice of fill ratio

Although the range of fill ratios reported in Fonseca's experiment [1] extends from 22.45% to 90.9%, it was observed in the simulation that both the 20% and 50% fill ratio were insufficient to drive the PHP into operating mode. Figure 5-1 shows the model results using a 5 turn – 100 mm PHP with a 50% fill ratio and 0.02 W of heat that is applied to the back of the copper block. Figure 5-1 shows the evolution of evaporator heat when the evaporator "guess" temperature is set to 5.1K. According to the solution method, the appropriate 'guessed' evaporator temperature will be such that the resulting steady state heat flow will match the applied heat load. The steady state heat flow in Figure 5-1 will only reach 0.007 W. Thus, an evaporator temperature larger than 5.1 K will be necessary for the heat flow to match the applied heat load of 0.02 W. However, since the critical temperature of helium, 5.19 K represents an upper limit for the model functionality, it is not possible to use the 50% fill ratio setting. For the same reason, the model will not work for any fill ratio less than 50%. Consequently, the set of fill ratio cases were chosen as 70%, 80% and 90%.

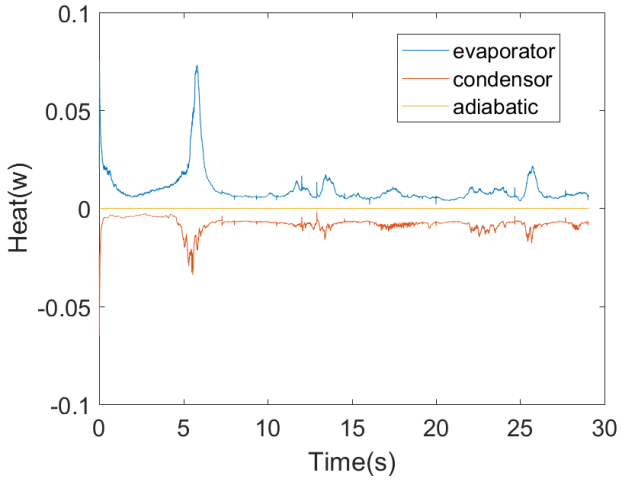

Figure 5-1: Heat versus time with evaporator temperature 5.1K

5.6 Final plan

The final plan for the parametric study is detailed in Table 5-1 and Table 5-2.

Table 5-1: Heat flux/condenser temperature plan with 70%, 80%, 90% fill ratios

Heat flux w/m^2	**Condenser Temperature(K)**
85.94	3.32
136.5	3.68
227.5	3.68

Table 5-2: Turn number/total length plan with evaporator/condenser length of 10mm

	2turn	5turn	10turn
50mm			
100mm			
200mm			

6 Flow pattern

6.1 Visualization method

For each of the conditions defined in Table 5-1and Table 5-2, Fluent was used to generate data in the form of velocity, temperature, pressure, and VOF as a function of time with a time step set between 0.015-0.005 seconds. The data files were saved every 100-time steps and were post-processed in CFD-post. As a result, for each of the conditions defined by Table 5-1 and Table 5-2, 120 data files are saved providing 30 seconds of real-time data.

The modeled PHP is oriented in Ansys Fluent so that the axial direction of the straight tubes line up with the y direction and the direction that the turns evolve lines up with the x direction. The z=0 position is located at the center of the tube. After creating an XY plane and sliding it through the middle of the PHP, all the cell information in the axial direction can be visualized. The VOF, temperature and pressure contour plots, and the velocity vector plot are usually plotted to analyze the flow behavior.

Animations are also generated to visualize the PHP's evolution with time. The animation of the VOF is of particular interest since it displays the movement of vapor plugs and liquid slugs inside of the PHP. Data are saved and plotted at 4 frames per second. However, CFD post processing can only generate animations with options of 24/30/50/60 frame rate per second which results in a short and condensed video time. Instead, the pictures and animations are saved first in CFD-post and then the frame rate of the animation was adjusted using MATLAB which can be played at a real-time rate of 4 frames per second. Because all the post-processing is the same for different runs, CFD-post codes called session files that record the post-processing steps were written to automate this process.

6.2 The initialization of the PHP system

The numerical start-up of the 2 turn – 50 mm PHP can be clearly visualized using the VOF animations. The start-up process can be broken into two parts. During the first part, in the alternative tube, one single big vapor plug rises or falls until the flow starts to circulate in one direction. All PHPs go through this part.

During the second part, slugs and plugs in the PHP stop until more vapor plugs grow in the evaporator section that makes the slugs and plugs circulate again. PHPs might or might not go through the second part and they might go through it multiple times.

The first part can be broken into the following few steps illustrated with a 2 turn – 50 mm PHP with a 70% fill ratio and an applied heat flux of $85 w/m^2$. First, the evaporator section is quickly filled with one vapor plug through fast evaporation with stagnant liquid for each tube as shown in Figure 6-1. Secondly, vapor plugs start to rise or fall in alternative tubes as illustrated in Figure 6-2. Lastly, the fluid starts to circulate in one direction as shown in Figure 6-3.

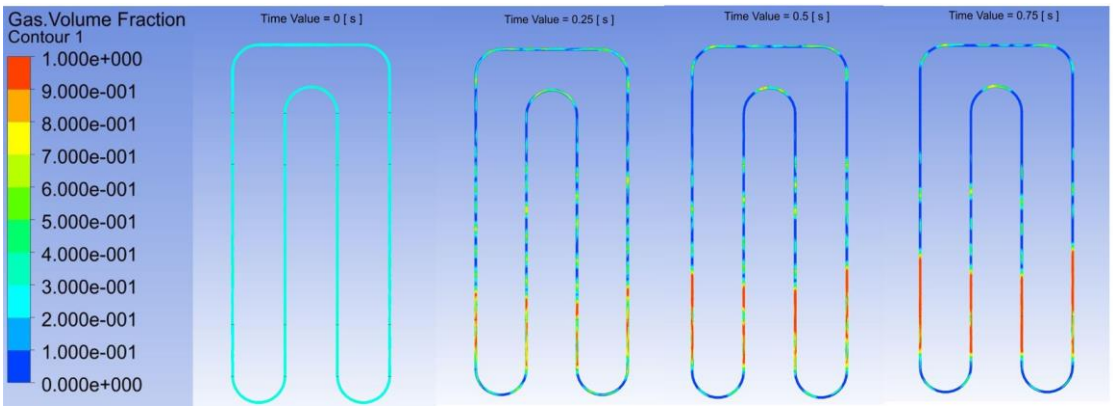

Figure 6-1: Time evolution of VOF plots for 2turn - 50mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 0-0.75 second

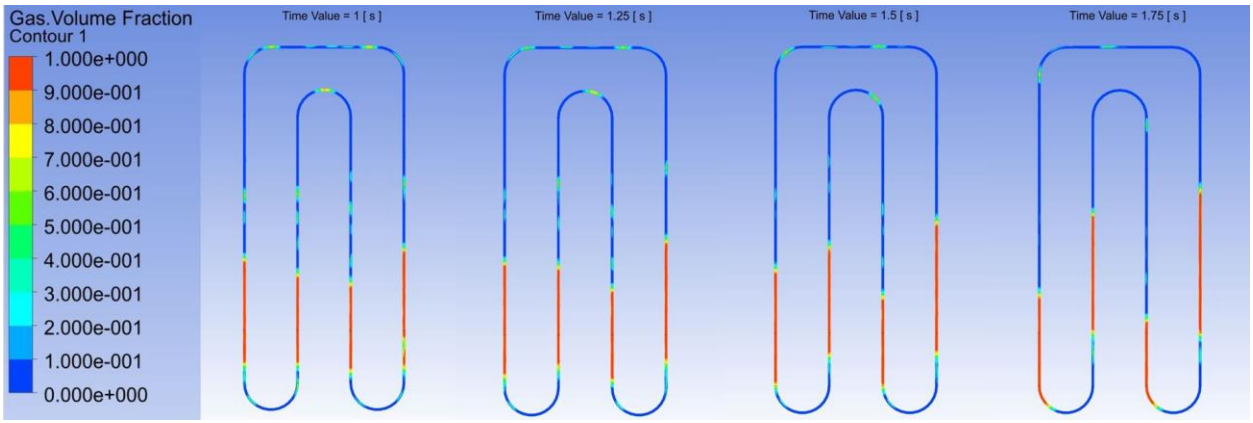

Figure 6-2 Time evolution of VOF plots for 2turn - 50mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 1-1.75 second

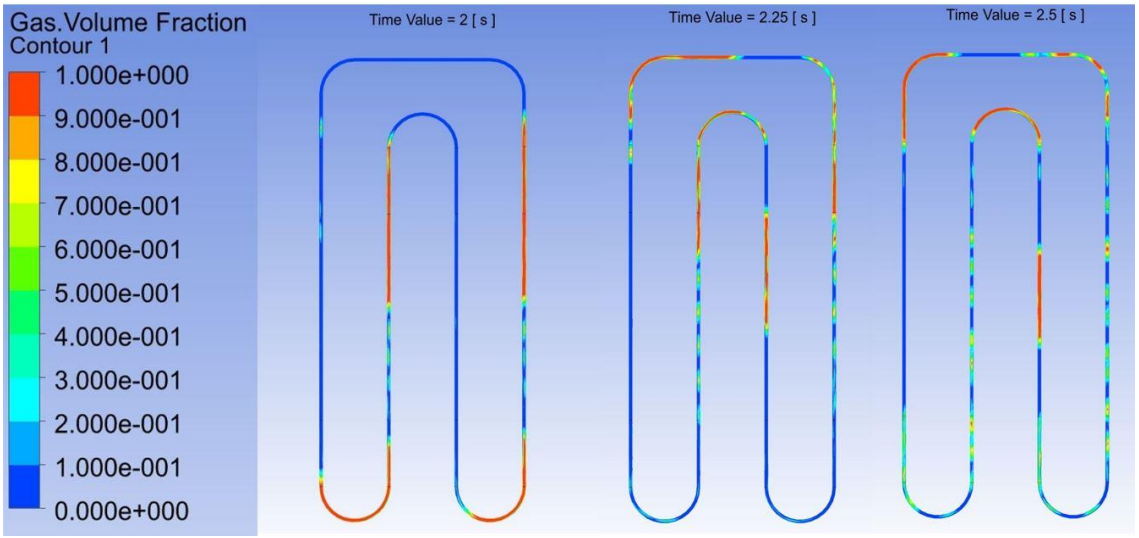

Figure 6-3: Time evolution of VOF plots for 2 turn - 50mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 2-2.5 second

The second part starts after the initial circulation brings vapor plugs into the condenser region. The flow through the PHP starts to slow down after the circulation is initiated. From 2.75 to 9 seconds, vapor plugs in the condenser region condense while liquid in the evaporator region evaporates as shown in Figure 6-4. However, now there is more than one vapor plug per tube.

Vapor plugs in the evaporator start to rise and fall again to induce circulation. This time, the circulation is stable with many vapor plugs inside of each tube as illustrated in Figure 6-5.

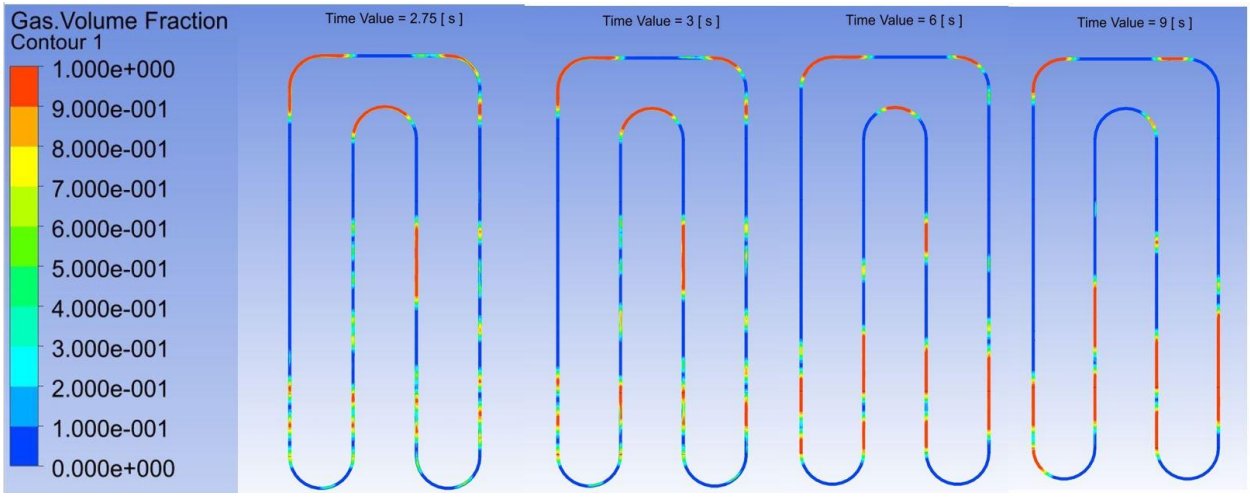

Figure 6-4: Time evolution of VOF plots for 2turn - 50mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 2.75-9 second

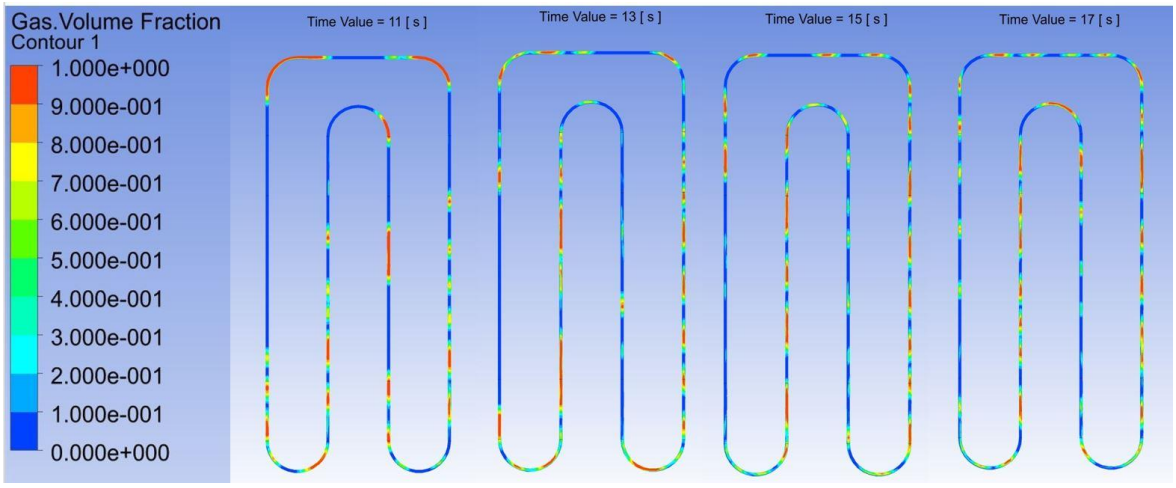

Figure 6-5: Time evolution of VOF plots for 2turn - 50mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 11-17 second

6.3 The shrinkage, breakdown, growth and merger of vapor plugs

When fluid motion in a PHP starts, there are mostly large vapor plugs inside. However, with time, there are mostly small vapor plugs circulating.

It happens through two kinds of mechanisms: bubble shrinkage and bubble breakdown.

As the vapor plugs go through the condenser, large vapor plugs shrink into small vapor plugs. One example of this process is shown in Figure 6-6.

As the vapor plugs go through the condenser, large vapor plugs also break into multiple smaller vapor plugs. One example of this behavior is shown in Figure 6-7.

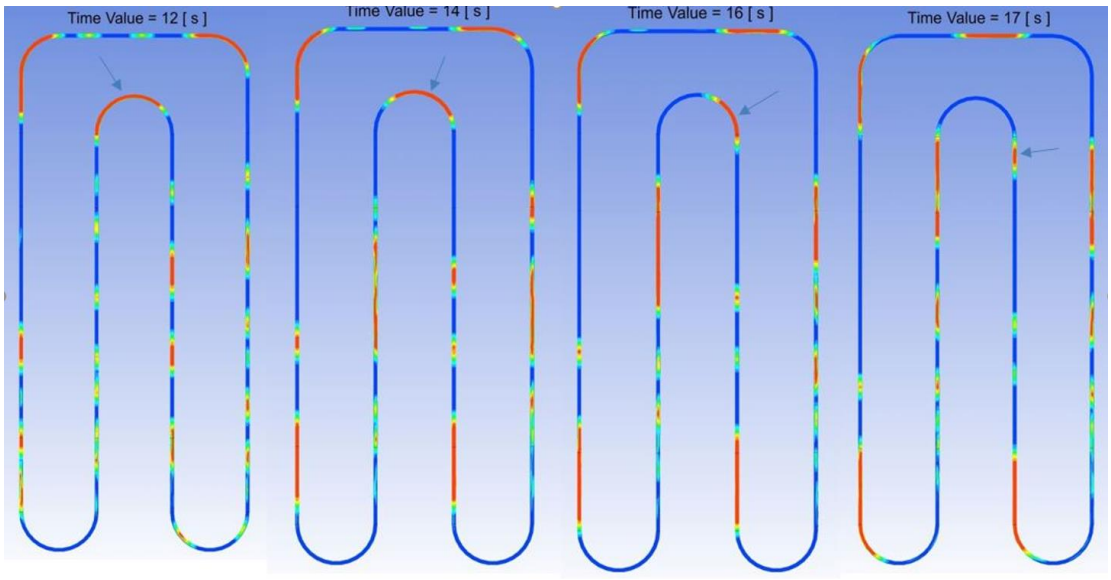

Figure 6-6: The shrinkage of vapor plugs for 2turn - 50mm PHP with 136 w/m^2 heat flux and 70% fill ratio

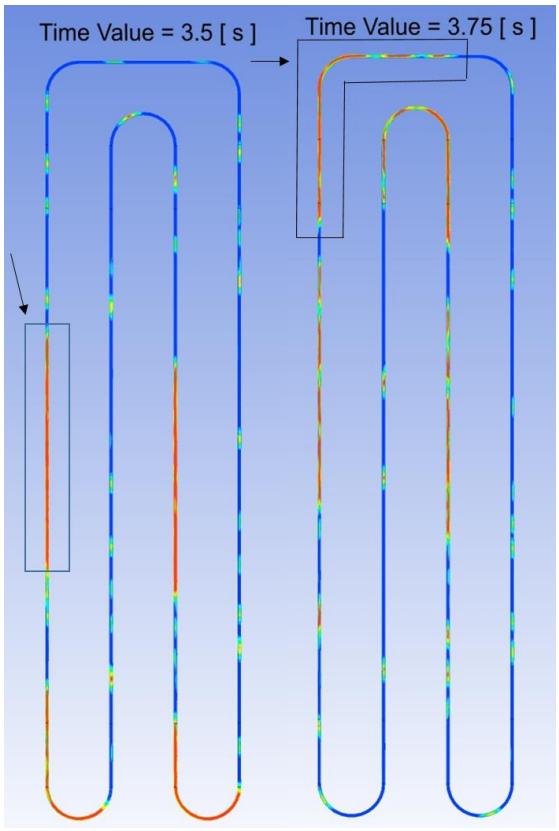

Figure 6-7: The breakdown of vapor plugs for 2turn - 100mm PHP with 85 w/m^2 heat flux and 70% fill ratio (clockwise flow)

The opposite mechanisms also happen in the evaporator. When small vapor plugs pass through the evaporator, bubbles grow bigger with time. An example can be seen in Figure 6-8.

Also, multiple vapor plugs can merge as they pass through the evaporator. An example is shown in Figure 6-9.

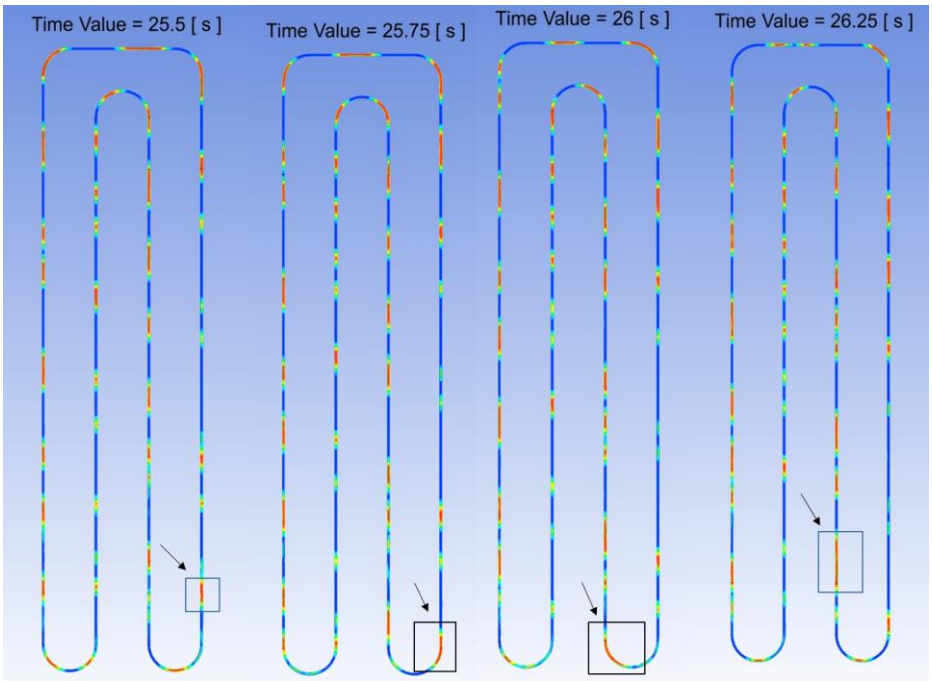

Figure 6-8: The growth of vapor plugs for 2turn - 100mm PHP with 85 w/m^2 heat flux and 70% fill ratio (clockwise flow)

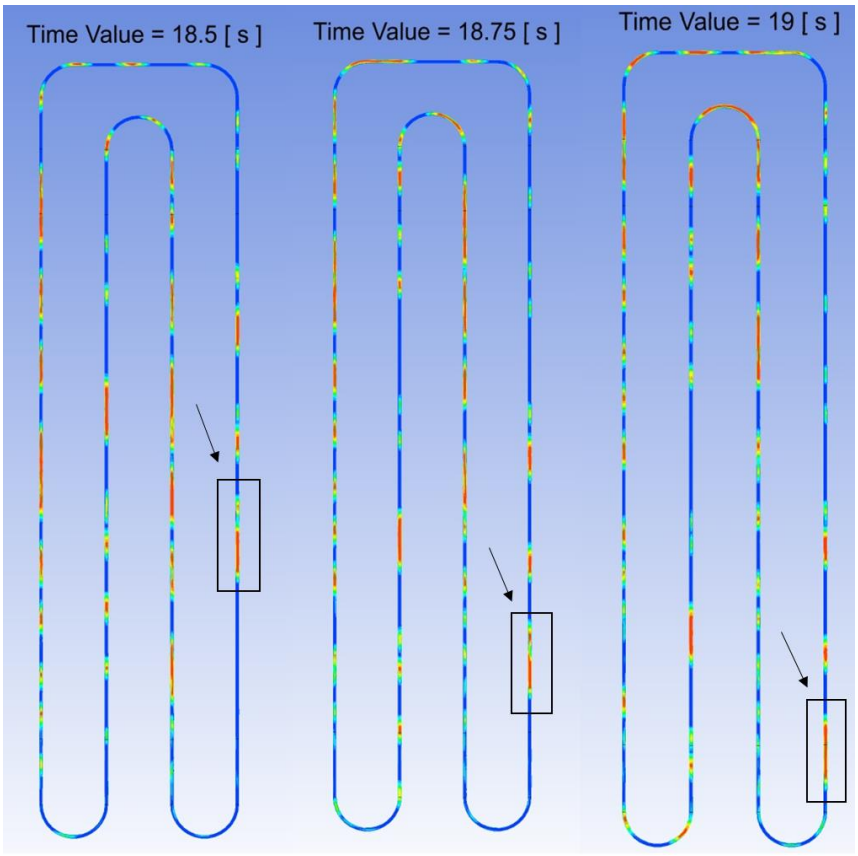

Figure 6-9: The merging of two vapor plugs for 2turn - 100mm PHP with 85 w/m^2 heat flux and 70% fill ratio (clockwise flow)

6.4 Steady state flow pattern among different geometries

After PHPs of different geometries, but at the same fill ratio, reach steady state, the slug/plug flow patterns are similar among different geometries as shown in Figure 6-10 and Figure 6-11. Additionally, the lengths of vapor plugs are similar among different geometries as shown in the length marks in Figure 6-10 and Figure 6-11. All PHP cases of different geometries reach steady state after 30 seconds.

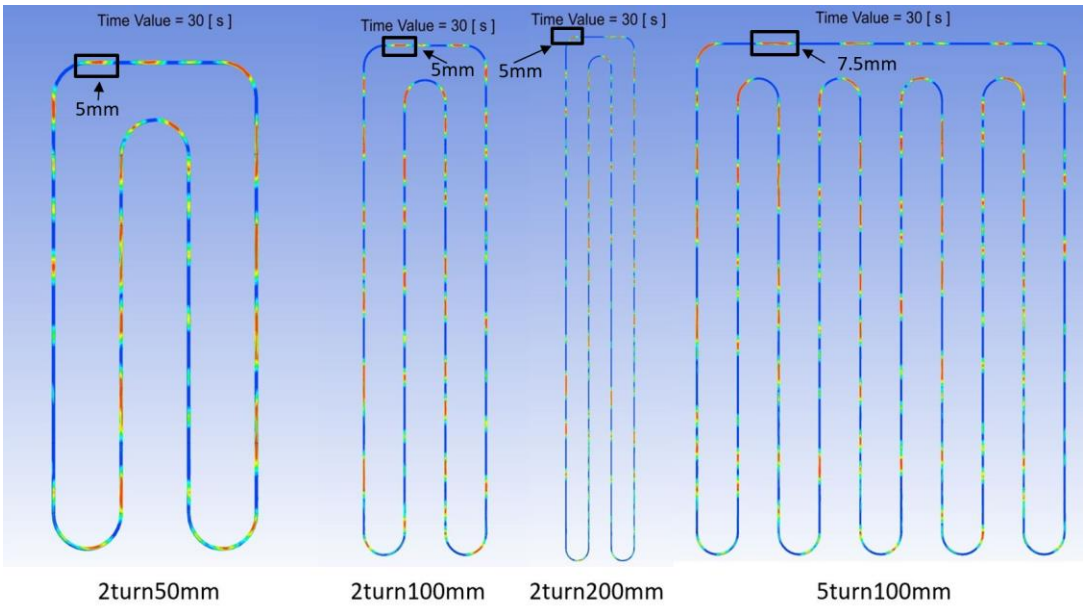

Figure 6-10: Steady state VOF visualization of different geometries with 85 w/m^2 heat flux and 70% fill ratio, part 1

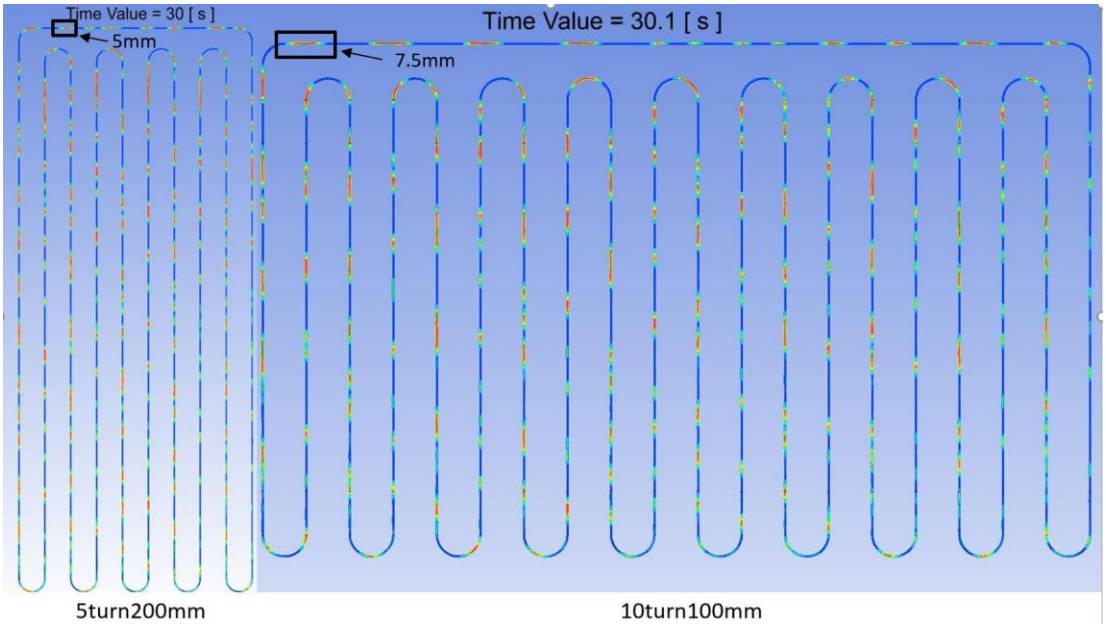

Figure 6-11 Steady state VOF visualization of different geometries with 85 w/m^2 heat flux and 70% fill ratio, part 2

However, when the heat flux is as high as $227 w/m^2$, the fill ratio is as low as 70%, and the geometry is as large as 10turn – 100mm, annular flow mixed with slug/plug flow can be observed as shown in Figure 6-12.

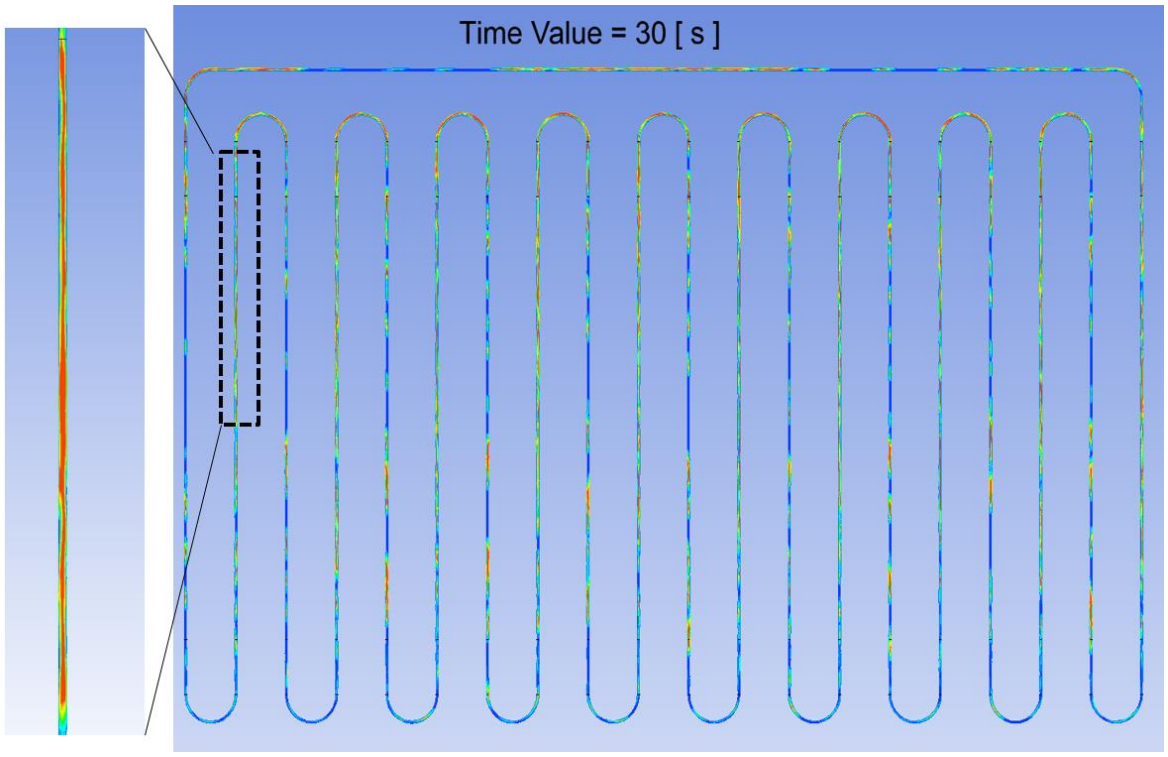

Figure 6-12 Steady state VOF visualization of 10turn – 100mm with 227 w/m^2 heat flux and 70% fill ratio

The rest of the VOF visualization results with other heat flux and geometry parameters are attached in Appendix 14.7.

6.5 Steady state flow pattern changes with fill ratio

VOF contour plots of 5turn 100 mm PHPs with different fill ratios are illustrated in Figure 6-13. With a 70% fill ratio, the liquid slugs/plugs are almost equal in length and number in the two sets of alternating tubes. In contrast, with a 90% fill ratio, there are almost pure liquid

slugs in the set of tubes where the flow is leaving the condenser while slug/plug flow exists in the set of tubes where the flow is leaving the evaporator.

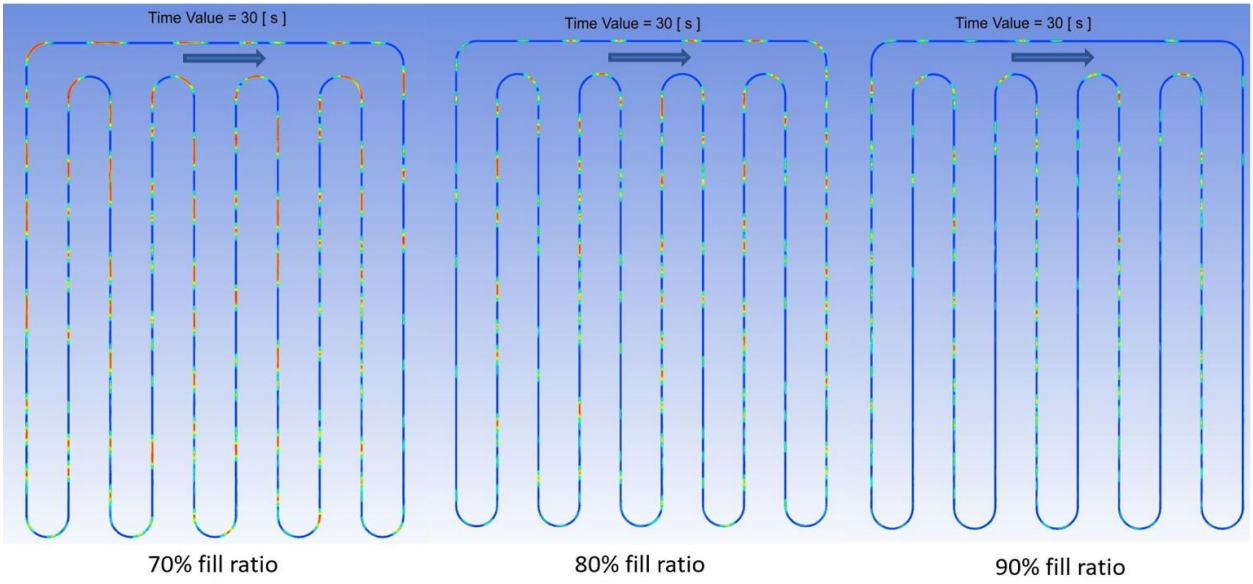

Figure 6-13: Steady state VOF visualization of different geometries with an applied heat flux of 85 w/m^2

6.6 Steady state flow pattern changes with heat flux

VOF contour plots of the 5turn - 100 mm PHPs with different heat flux values were visualized as illustrated in Figure 6-14. Here the heat flux has little influence on the flow pattern.

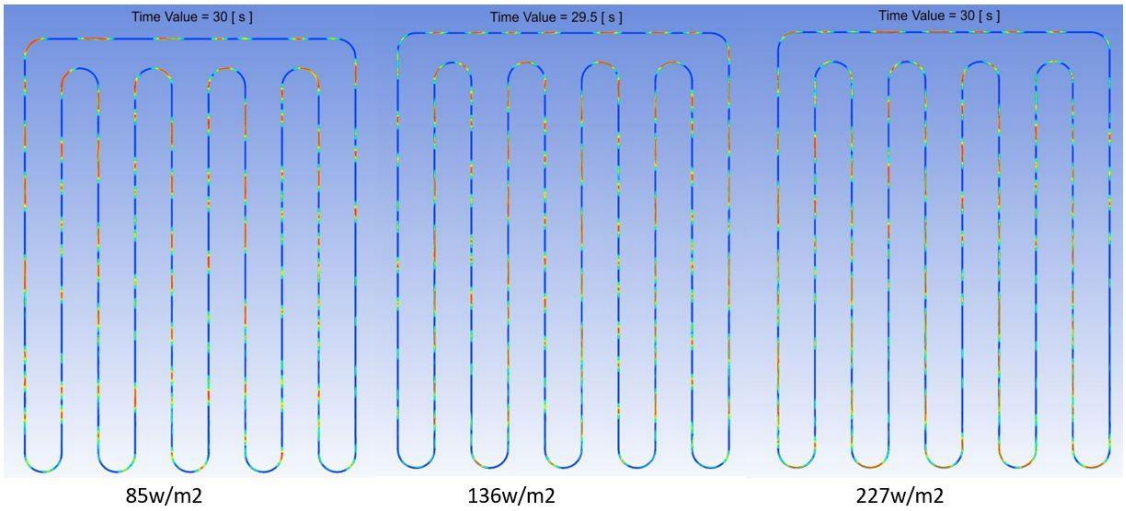

Figure 6-14: Steady state VOF visualization of different geometries with a 70% fill ratio

6.7 Reynolds number

For pipe flow, when the Reynolds number is lower than 2300, the flow is laminar. As a result, the maximum Reynolds number of different geometries and thermal conditions were plotted to check if the maximum Reynolds number is less than 2300. It is an effective way to validate the choice of laminar flow as the viscous model.

6.7.1 Maximum Reynold number

The Reynolds number is plotted throughout the PHP domain after it reaches steady state as shown in Figure 6-15. The maximum Reynolds number is shown at the top of the legend. As in Figure 6-15, for the 2 turn – 50 mm PHP with a 70% fill ratio and an applied heat flux of 85 w/m^2 in the evaporator, the maximum Reynolds number is 986.9.

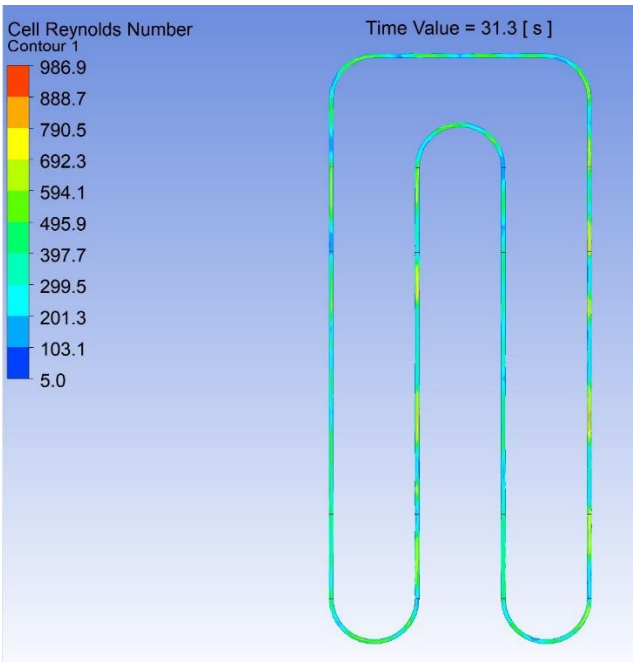

Figure 6-15: Re number visualization for the 2 turn – 50 mm PHP with a 70% fill ratio and an applied heat flux of $85 w/m^2$ in the evaporator

After plotting all the Reynolds number values after flow reaches steady state, it is found that the maximum Reynold number varies between 581-1867. In conclusion, the flow in all the PHPs simulated in the study is laminar. However, because the Reynolds number in this study is dangerously close to 2300, a helium PHP might on occasion reach a point where the flow becomes turbulent with other geometries and thermal conditions.

6.7.2 Reynolds number versus geometry

Figure 6-16, Figure 6-17, and Figure 6-18 plot the Reynolds number versus total tubing length with different values of applied heat flux. It can be observed that at the highest value of applied heat flux, a steady increase of the Reynolds is associated with the increase of tube length. At the smaller values of heat flux, the dependence of Reynolds number on length or fill ratio is

not clear. It can be observed that the longer tubing length contributes to larger Reynolds numbers. At the same time, the fill ratio and applied heat flux do not produce a noticeable influence on the Reynolds number. As a result, it is possible that in a PHP geometry with a total length larger than 4000 mm turbulent flow could exist in a helium PHP.

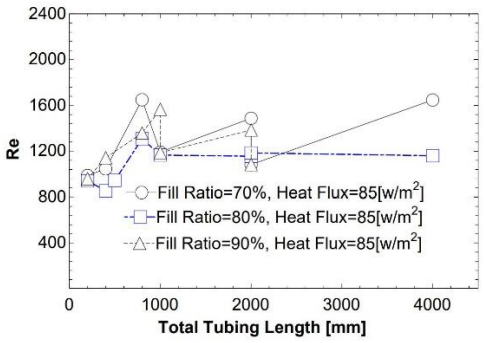

Figure 6-16: Re number versus total tubing length with 85 w/m^2

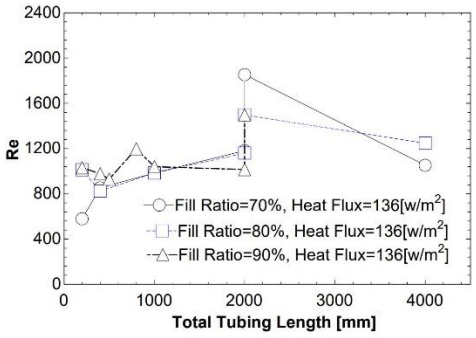

Figure 6-17: Re number versus total tubing length with 136 w/m^2

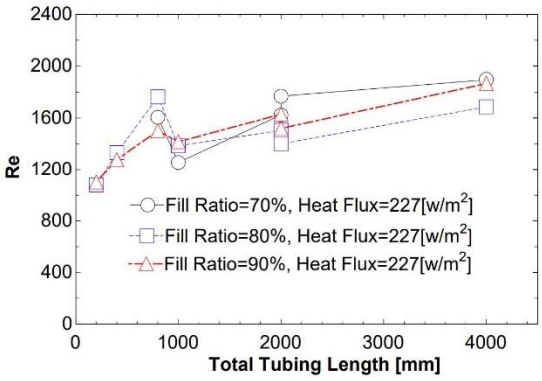

Figure 6-18: Re number versus total tubing length with 227w/m^2

7 Circulatory pattern

7.1 Category method

In all the different geometry, heat load, and fill ratio conditions that were investigated, the flow inside the PHP develops into one-directional circulatory flow within 30 seconds of simulation. Among the circulatory flows, most of them have a steady velocity value and are defined as steady circulatory flow such as shown in Figure 7-1 The rest display changing velocity values and are defined as pulsating circulatory flow as shown in Figure 7-2.

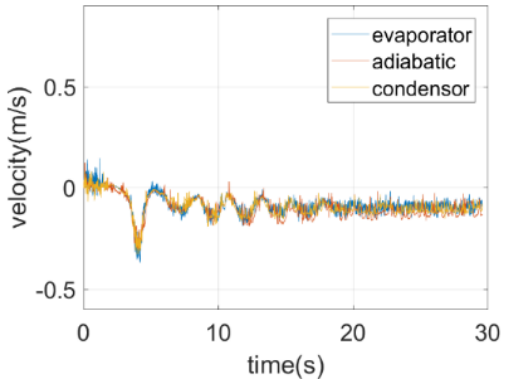

Figure 7-1: Velocity profile of 10 turn – 200 mm PHP with an 80% fill ratio and an applied heat flux of 136 w/m^2

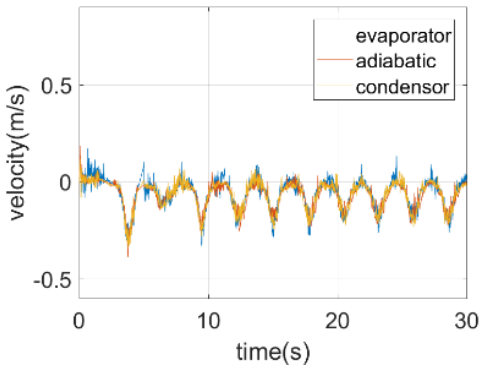

Figure 7-2: Velocity profile of 10 turn – 200 mm PHP with a 70% fill ratio and an applied heat flux of 136 w/m^2

The difference between steady circulatory flow and pulsating circulatory flow is that the steady circulatory flow as shown in Figure 7-1 does not have a distinct frequency characteristic, while pulsating circulatory flow as shown in Figure 7-2 displays a distinct frequency.

Furthermore, for steady circulatory flow, the velocity variation can be as low as the velocity variations that are shown in Figure 7-1. On the other hand, the velocity variation can also be large as shown in Figure 7-3. It is easy to mistake the flow conditions shown in Figure 7-3 as pulsating circulatory flow. However, upon careful examination of the velocity profile between 25-26 seconds, as shown in Figure 7-4, we can see that the velocities at different sections are not oscillating at the same time. In contrast, for the cases considered as pulsating circulatory flow, the velocities at different sections all pulsate at the same time as shown in Figure 7-2.

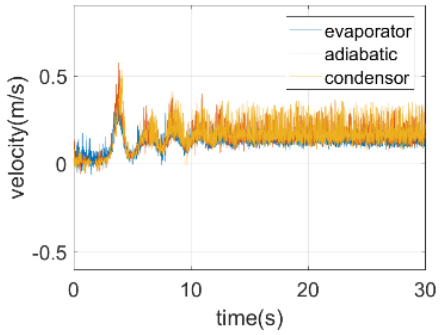

Figure 7-3: Velocity profile of 2turn - 200mm PHP with 80% fill ratio and 85 w/m^2 heat flux

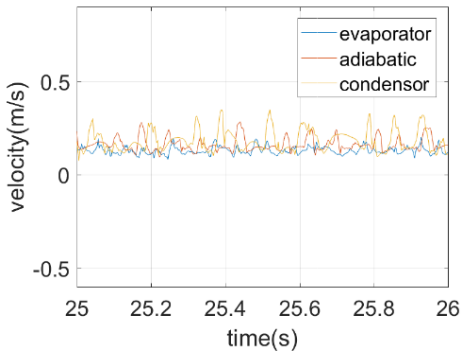

Figure 7-4: Enlarged view of the 25-26 second region from Figure 75.

To simplify the process of categorizing the circulatory pattern, only the frequency of the velocity oscillation is considered.

7.2 Trend observation

Using the method of categorizing described above, the circulatory pattern results associated with various conditions are listed in Table 7-1.

Table 7-1: Circulatory pattern according to different heat load, fill ratio, and geometry

70% fill ratio and 85 w/m^2				80% fill ratio and 85 w/m^2				90% fill ratio and 85 w/m^2			
	2turn	5turn	10turn		2turn	5turn	10turn		2turn	5turn	10turn
50mm	Steady			50mm	Steady	Steady		50mm	Steady		Steady
100mm	Pulsating	Pulsating	Steady	100mm	Steady	Steady	Steady	100mm	Steady	Steady	Steady
200mm	Pulsating	Steady	Steady	200mm	Steady	Steady	Steady	200mm	Steady	Steady	
70% fill ratio and 136 w/m^2				80% fill ratio and 136 w/m^2				90% fill ratio and 136 w/m^2			
	2turn	5turn	10turn		2turn	5turn	10turn		2turn	5turn	10turn
50mm	Pulsating			50mm	Steady		Pulsating	50mm	Steady	Steady	
100mm	Steady	Steady	Steady	100mm	Pulsating	Steady	Steady	100mm	Steady	Steady	Steady
200mm		Pulsating	Pulsating	200mm		Steady	Steady	200mm	Steady	Steady	
70% fill ratio and 227 w/m^2				80% fill ratio and 227 w/m^2				90% fill ratio and 227 w/m^2			
	2turn	5turn	10turn		2turn	5turn	10turn		2turn	5turn	10turn
50mm				50mm	Steady	Steady		50mm	Steady	Steady	Steady
100mm	Pulsating	Pulsating	Pulsating	100mm	Steady	Steady	Steady	100mm	Steady	Steady	Steady
200mm	Steady	Steady	Pulsating	200mm	Steady	Steady	Steady	200mm	Steady	Steady	Steady

As illustrated in Table 7-1, at 70% fill ratio, there are 10 cases of PHPs with pulsating circulatory flow. When the fill ratio is increased to 80%, there are only 2 cases of PHPs with pulsating circulatory flow. Furthermore, at the 90% fill ratio, all PHPs exhibit steady circulatory flow. In conclusion, higher flow ratios lead to a higher possibility of steady circulatory flow.

On the other hand, with applied heat flux values of $85w/m^2$, $136w/m^2$, and 227 w/m^2 the PHPs display 3 cases, 4 cases, and 4 cases of pulsating circulatory flow respectively. As a result, the value of heat flux has an unclear effect on the circulatory pattern.

To investigate how the turn number and total length influence the circulatory pattern, the table was listed in an alternate way with an emphasis on turn number and total length as shown in Table 7-2.

Table 7-2: Circulatory pattern with turn number and total length

2turn and 50mm				5turn and 50mm				10turn and 50mm			
	70%	80%	90%		70%	80%	90%		70%	80%	90%
85 w/m^2	Steady	Steady	Steady	85 w/m^2		Steady		85 w/m^2			Steady
136 w/m^2	Pulsating	Steady	Steady	136 w/m^2			Steady	136 w/m^2		Pulsating	
227 w/m^2		Steady	Steady	227 w/m^2		Steady	Steady	227 w/m^2			Steady
2turn and 100mm				5turn and 100mm				10turn and 100mm			
	70%	80%	90%		70%	80%	90%		70%	80%	90%
85 w/m^2	Pulsating	Steady	Steady	85 w/m^2	Pulsating	Steady	Steady	85 w/m^2	Steady	Steady	Steady
136 w/m^2	Steady	Pulsating	Steady	136 w/m^2	Steady	Steady	Steady	136 w/m^2	Steady	Steady	Steady
227 w/m^2	Pulsating	Steady	Steady	227 w/m^2	Pulsating	Steady	Steady	227 w/m^2	Pulsating	Steady	Steady
2turn and 200mm				5turn and 200mm				10turn and 200mm			
	70%	80%	90%		70%	80%	90%		70%	80%	90%
85 w/m^2	Pulsating	Steady	Steady	85 w/m^2	Steady	Steady	Steady	85 w/m^2	Steady	Steady	
136 w/m^2			Steady	136 w/m^2	Pulsating	Steady	Steady	136 w/m^2	Pulsating	Steady	
227 w/m^2	Steady	Steady	Steady	227 w/m^2	Steady	Steady	Steady	227 w/m^2	Pulsating	Steady	Steady

Counting all the circulatory patterns for 2 turns, the table displays 5 cases of pulsating circulatory flow. At the same time, 3 cases and 4 cases of pulsating circulatory flow exist for 5 turns and 10 turns, respectively. On the other hand, for PHPs with total lengths of 50 mm, 100 mm, and 200 mm, 2 cases, 6 cases and 4 cases of pulsating circulatory flow exist, respectively. From the above observations, it is reasonable to conclude that the number of turns and the total length do not have a significant effect on the circulatory pattern.

In summary, of all the parameters investigated, the fill ratio is the only parameter that has a consistent effect on the circulatory pattern. All the velocity profiles associated with the various conditions are available in Appendix 14.8 for a more detailed examination.

8 Averaged velocities and frequencies

8.1 Averaged velocities

For the non-pulsating flow cases, all velocities come to steady state roughly after 25 seconds. Consequently, the velocities between 25 seconds and 30 seconds are averaged and reported (below) as the average velocities of the PHPs.

8.1.1 The effect of total length on velocity

The influence of PHP length on velocity is displayed in Figure 8-1, Figure 8-2 , and Figure 8-3. As displayed there, the trends vary with the different fill ratios. Among all the fill ratios, the 80 percent fill ratio data displays the most unclear trends. For the 70% fill ratio cases, the average velocity increases as the length increases from 50 mm to 100 mm for each value of heat flux. As the length continues to increase, the effect of the increasing length on velocity is uncertain as well. For the 90% fill ratio cases, the velocity increases from 100mm to 200mm for each value of heat flux. Between the length of 50 mm and 100 mm, there is no consistent dependence of the velocity on length.

8.1.2 The effect of turn number on velocity

At the 70% fill ratio, it can be observed from Figure 8-4 that the velocity first increases with the increasing turn number, and then decreases with the increasing turn number. In other words, there exists an optimum number of turns producing the maximum velocity. The model results display that the PHP with 5 turns produces the maximum velocities. Such a result is consistent with the experimental data of Li, Li, Xu [9] in which an 8-turn PHP produced a better conductance per turn than a 48-turn PHP. The issue is of great significance for applications where one might ask whether it is better to use six individual 8-turn PHPs or one 48-turn PHP to

transfer heat. Figure 4-4 suggests that for helium PHPs with a 70% fill ratio, the optimum number of turns is five.

From Figure 8-5 for the 80% fill ratio cases, it can be observed that there is an optimum value for $227 w/m^2$. However, the trend is unclear for the other two values of heat flux. From Figure 8-6 for the 90% fill ratio cases, although velocities decrease with turn number for 85 w/m^2 and $136 w/m^2$, the trend is unclear for $227 w/m^2$. In summary, while an optimum number of turns exists for a fill ratio of 70%, no consistent optimum turn numbers exist for the 80% and 90% fill ratio cases.

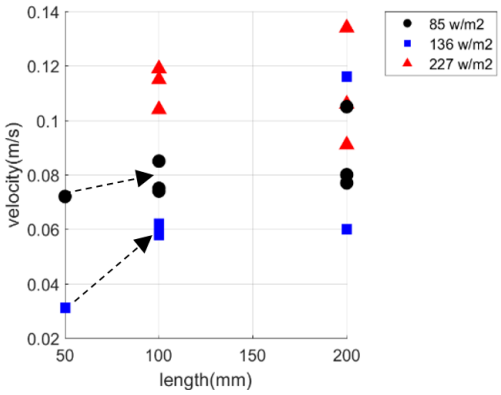

Figure 8-1: Velocity vs. length with a 70% fill ratio

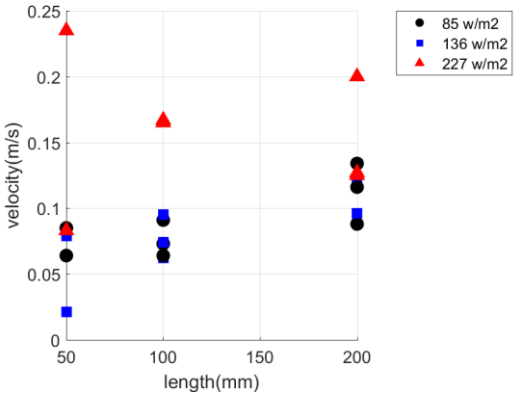

Figure 8-2: Velocity vs. length for an 80% fill ratio

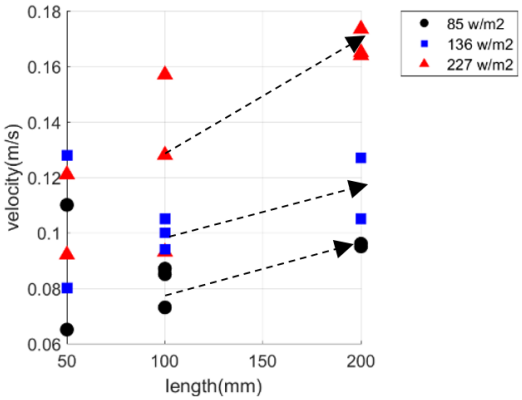

Figure 8-3: Velocity vs. length for 90% fill ratio

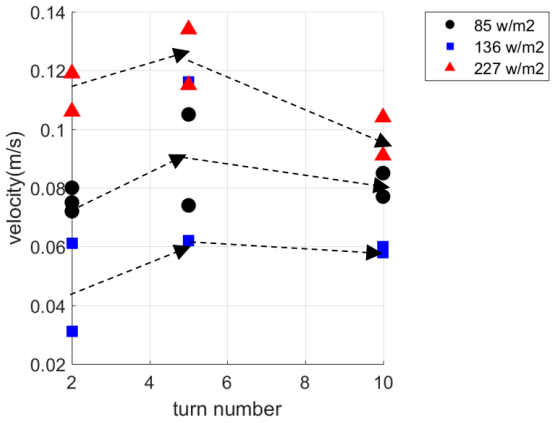

Figure 8-4: Velocity vs. number of turns for 70% fill ratio

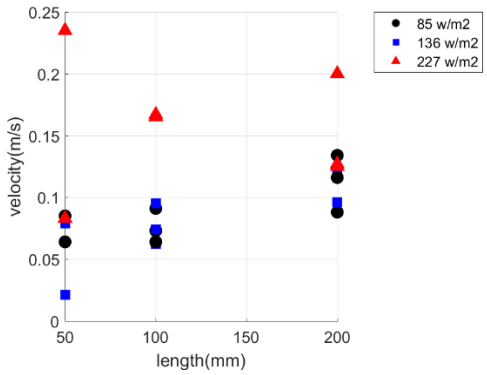

Figure 8-5: Velocities vs number of turns for 80%

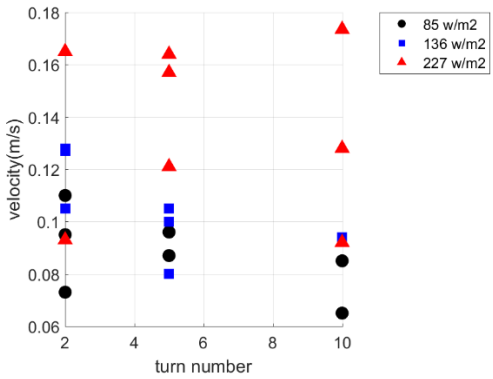

Figure 8-6: Velocities vs number of turns for 90%

8.1.3 The effect of total tube length on velocity

Figure 8-7, Figure 8-8, and Figure 8-9 display the velocity as a function of total tube length at different fill ratios. The total tube length is calculated by multiplying the number of turns by the length of all three sections (evaporator, adiabatic, condenser) in one turn. For the 70% fill ratio, an optimum tube length exists at which the velocity is maximized. 1000 mm is the optimum total tube length that produces the highest velocities for the 70% fill ratio cases. For both the 80% and 90% fill ratio cases, although the velocities increase slightly at the lower heat flux values, they consistently decrease with increasing total tubing length at the highest heat flux value of $227 w/m^2$.

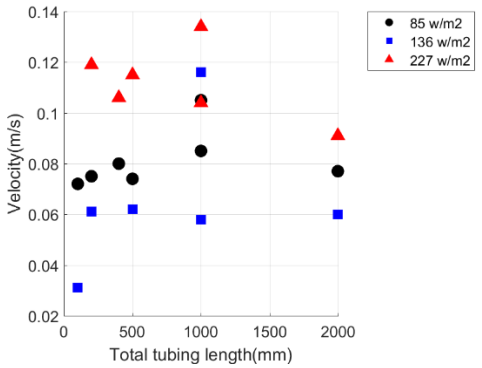

Figure 8-7: Total tubing length vs Velocities for 70%

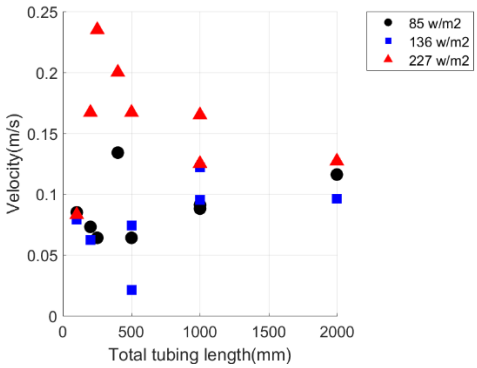

Figure 8-8: Total tubing number vs Velocities for 80%

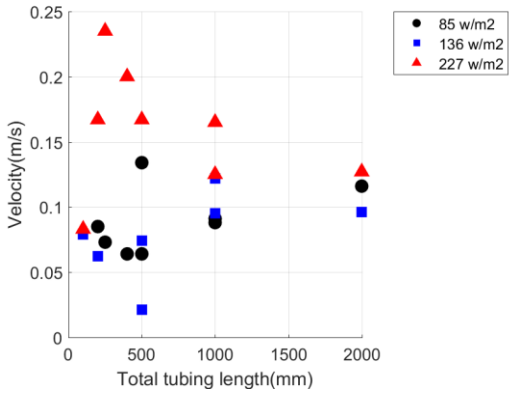

Figure 8-9: Total tubing number vs Velocities for 90%

8.1.4 The effect of heat flux on velocity

Figure 8-1 to Figure 8-9 all consistently demonstrate that velocities increase with heat flux. In helium PHP experiments, effective thermal conductivity also increases with heat flux. The results of the model strongly suggest that the experimentally observed increase of effective thermal conductivity is due to the increase of flow velocity. Figure 8-10 displays the experimental data from Li, Li, Xu [9] demonstrating the increase of effective thermal

conductivity with heat load (heat flux). The results from the numerical model included in the graph demonstrate the strong correlation between the velocity and thermal conductivity dependence on the heat load (heat flux).

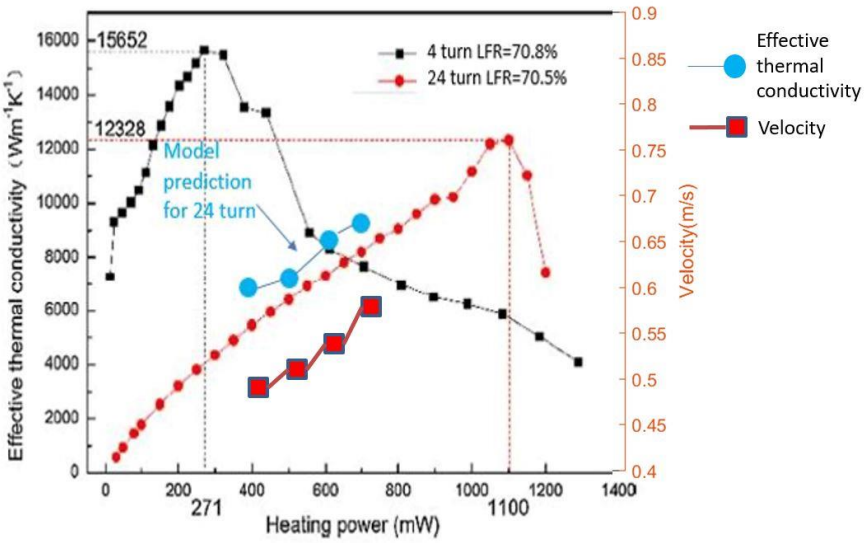

Figure 8-10: Effective thermal conductivity/velocity model results versus heating power

8.1.5 The effect of fill ratio on velocity

The data in Figure 8-11 display that the highest velocities are obtained with the 80% fill ratio.

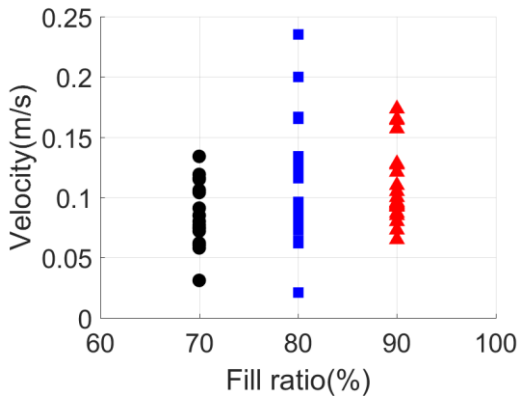

Figure 8-11: Fill ratio Vs Velocities

8.2 Frequencies

Data from the various cases displaying pulsating flow allow further characterization in terms of the pulsating frequency and the dependence of those frequency values on the other parameters such as PHP length, applied heat flux, fill ratio. In this section the trends of frequency vs. the other parameters are explored and described using the associated FFT results.

The results in Table 7-1 show that few cases with 80% and 90% fill ratios displayed pulsating flow. Further, after plotting the FFT results for different fill ratios, it is found that the 80% and 90% fill ratios have multiple peaks in the FFT plot. Consequently, there are no definite frequencies for these cases.

On the other hand, the frequencies found for the various pulsating flow cases with a 70% fill ratio are definitive and plotted in a 3D scatter plot as shown in Figure 8-12. Here the frequency values are plotted as a function of PHP length and number of turns. A 2D projection of the same information onto the frequency – PHP length plane (including all the various turn number cases) is shown in Figure 8-13. Two trends can be observed from this figure. First,

longer lengths lead to smaller frequencies or slower oscillations. Second, higher evaporator heat flux leads to higher frequencies or faster oscillations. In contrast, as shown in the 2D projection onto the frequency – turn number plane shown in Figure 8-14 (which includes all the various length cases), the number of turns does not have a direct impact on frequencies.

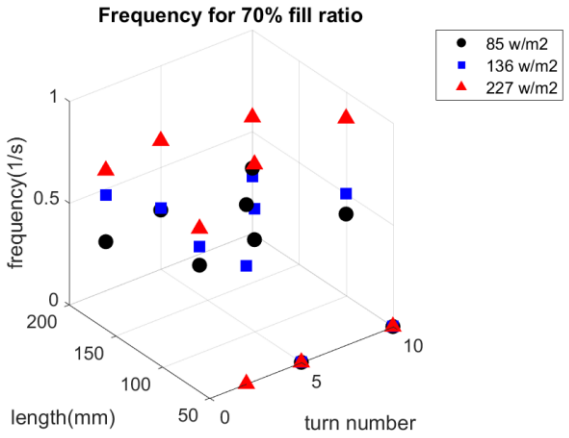

Figure 8-12: Frequencies vs turn number vs length for 70% fill ratio

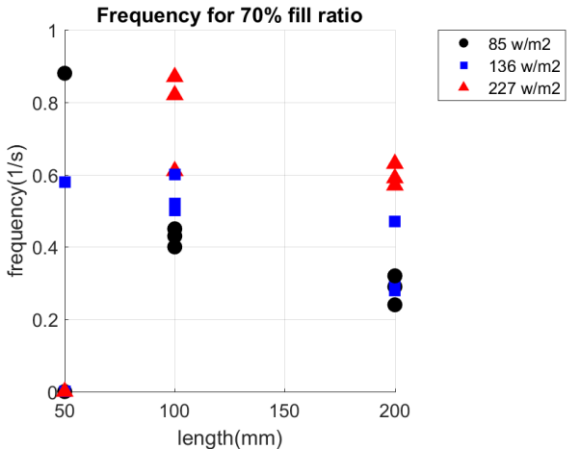

Figure 8-13: Frequencies vs length for 70% fill ratio

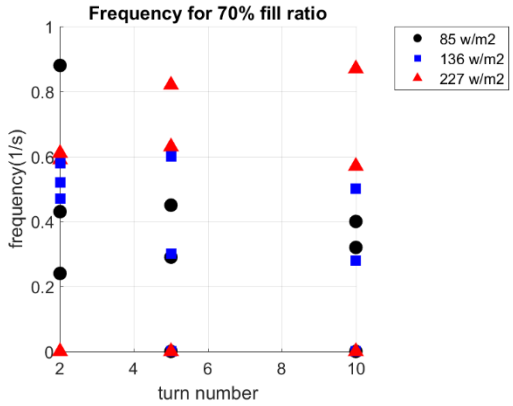

Figure 8-14: Frequencies vs turn number for 70% fill ratio

9 Thermal results

9.1 Thermal performance results

The raw modeling results detailing the evaporator temperature as dependent on turn number, total length, heat flux and fill ratio are attached in the appendix14.9. The present chapter presents thermal features of the modeling results including the influence of the copper blocks at the evaporator and condenser ends, the flow-boiling heat transfer characteristics, trends of the evaporator temperature, and the PHP's overall effective thermal conductivity.

9.2 Heat transfer to boiling helium in a tube physics validation

To assist verifying the numerical model, it is important to verify that the physics happening in the simulation matches with what is observed in experiments. One of the physical mechanisms that can be verified in the numerical model is the physics of heat transfer to boiling helium in a tube.

9.2.1 Experimental result for heat transfer to boiling helium in a tube

The relationship between heat flux q and temperature difference ΔT for heat transfer to boiling helium in a tube is explored in experiments by Ogata and Sato [27]. The experiment process is as such: The test tube is a straight stainless-steel tube that is electrically heated. Liquid helium was cooled before being forced to flow vertically upwards through the tube. Four temperature sensors were placed on the experiments. Two of the temperature sensors were placed at two locations on the outside wall of the tube section, two temperature sensors were placed to measure the fluid temperature at the inlet and at the outlet of the tube. Fluid temperature T_b is calculated from the heat load balance along the test section, while the wall temperature T_w is calculated from the temperature drop between the outside and inside of the

tube wall. ΔT is defined as the temperature difference between T_w and T_b. The schematic plot of the experiment is shown in Figure 9-1. The resulting q versus ΔT plot is shown in Figure 9-2. Interestingly, when the heat flux decreases, the q versus ΔT curve takes a new path compared to when the heat flux increases.

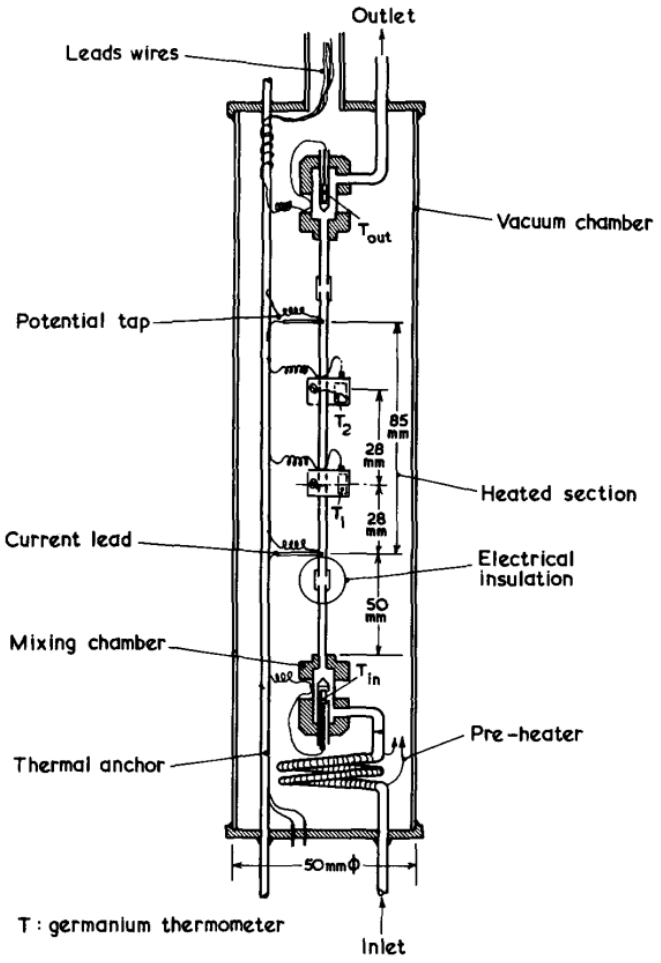

Figure 9-1: Boiling helium in a tube experiment

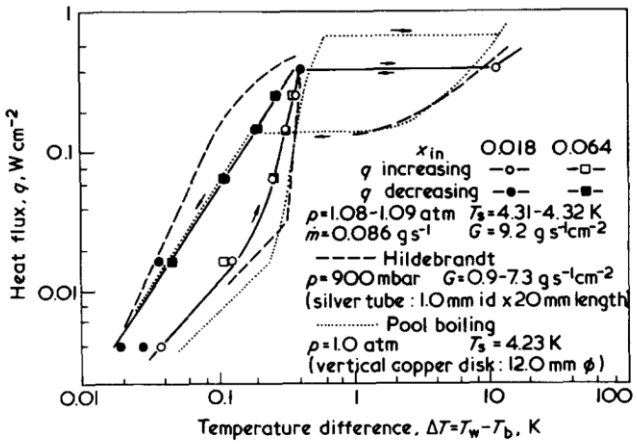

Figure 9-2: Q versus ΔT plot for boiling helium in a tube

9.2.2 Comparing simulation results with experimental results

In the PHP simulation, the surface heat flux applied to the evaporator surface and the resulting temperature difference between the evaporator surface-averaged temperature and the volume-averaged temperature are used to generate a q versus ΔT plot that is like Figure 9-2. Such a plot is overlaid on top of Figure 9-2 for comparison in Figure 9-3. The three star-shaped markers in Figure 9-3 represents the numerical values of q and ΔT at three different heat flux levels for the 2-turn 200 mm PHP with a 70% fill ratio. The model-based markers closely follow the experimental curve. Thus, the physics related to pool boiling heat transfer in a vertical tube is accurately captured by the PHP simulation. Because the wall temperature is lower than the fluid temperature in the condenser, the experimental results of the pool boiling do not apply to the condenser.

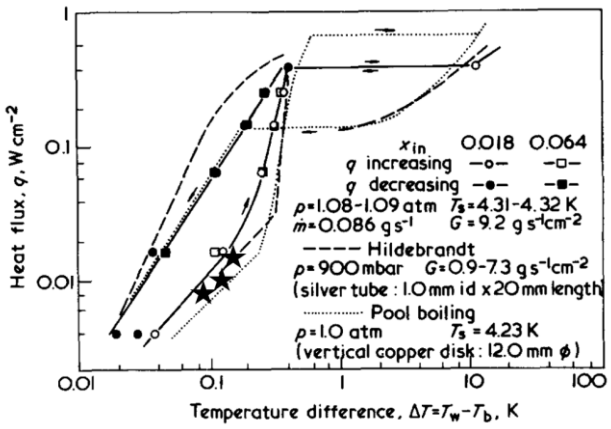

Figure 9-3: Q versus ΔT plot for 2turn - 200mm PHP with 70% fill ratio

Furthermore, simulation results with different geometries and heat flux values were used to explore whether the above conclusion is universal. Figure 9-4 and Figure 9-5 plot the similar overlay plots for the 10-turn100 mm PHP and the 5-turn100 mm PHP, respectively. Both plots showing the star-shaped markers reflect that the numerical results follow closely to the experimental curve.

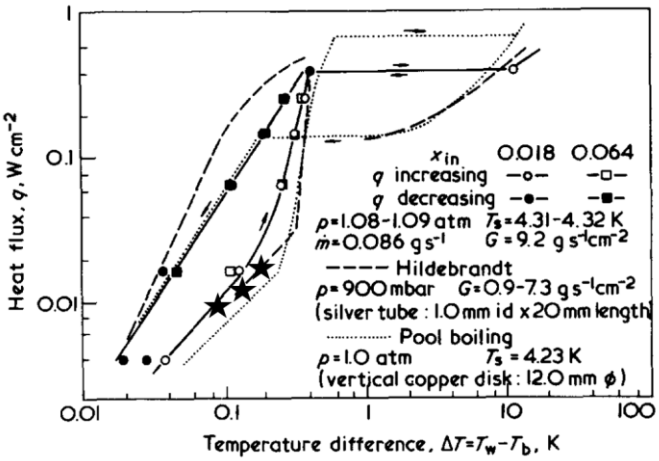

Figure 9-4: Q versus ΔT plot for the 10 turn – 100 mm PHP with a 70% fill ratio

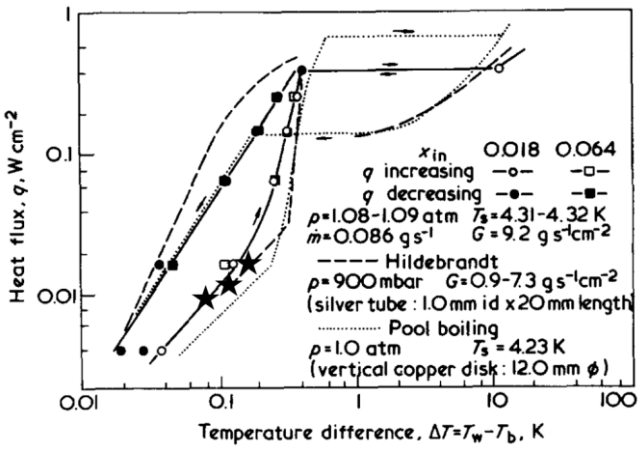

Figure 9-5: Q versus ΔT plot for the 5 turn –100 mm PHP with a 90% fill ratio

9.3 Temperature distribution inside of vapor plugs

Figure 9-6 is the VOF plot for the 2 turn 50mm PHP at a 70% fill ratio with an applied heat flux of 85 w/m^2 after the PHP reaches steady state. Area 1 and Area 2 highlight two vapor plugs in the evaporator and condenser sections, respectively. In Figure 9-7, Area 1 is enlarged, and the corresponding VOF contour plot and temperature contour plot are displayed on the left- and right-hand side, respectively. On the right-hand plot, a thin layer of high temperature is visible near the wall. Moving toward the center of the tube, the temperature decreases, reaching a minimum at the center of the tube. The left-hand plot reveals that the same thin layer of high temperature near the wall and the decreasing temperature toward the center corresponds to a vapor plug in the VOF contour plot.

In Figure 9-8, the opposite features can be observed. Here a thin layer of low temperature exists near the wall, and temperatures increase from the wall to the center of the tube. Again, the VOF plot on the left reveals that the strong temperature gradient near the wall is associated with the presence of a vapor plug.

Additionally, it can be observed that a thin layer of high temperatures only exists in the evaporator while a thin layer of low temperatures only exists in the condenser. In both cases, it happens in the area where there is a vapor plug. There are no thin layers of either high or low temperatures in the adiabatic section, even though the vapor plugs are present there as well.

The reason for the existence of the thin layer of either high or low temperature is that when there is heat flux coming in or out of the wall, the lower heat transfer coefficient of a vapor plug causes a high temperature gradient near the wall.

The same phenomenon can also be observed for PHPs of other geometries and thermal conditions.

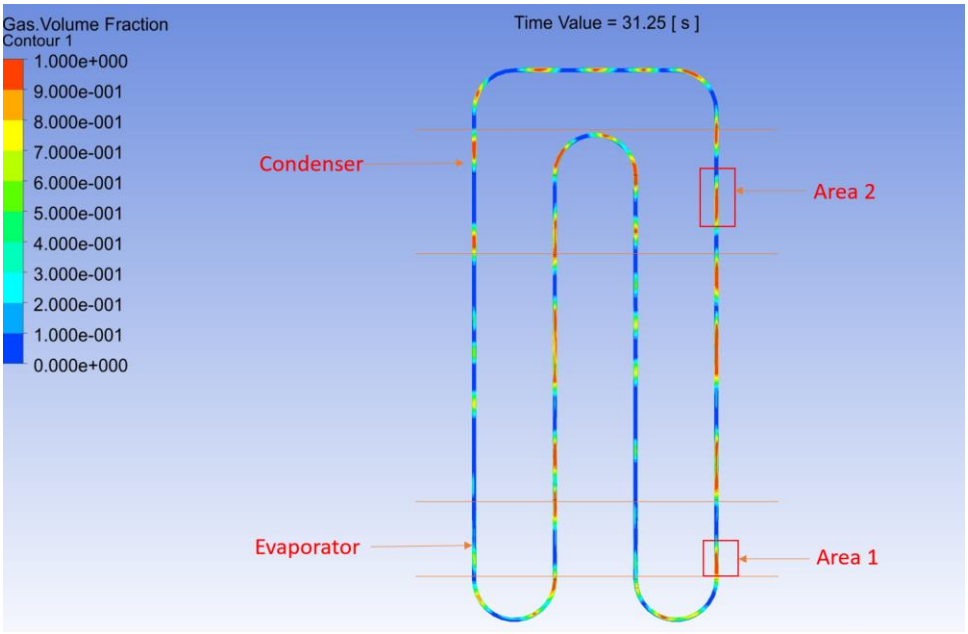

Figure 9-6: VOF contour plot for 2turn - 50 mm PHP (85 w/m^2 heat flux and 70% fill ratio) after the PHP reached steady state

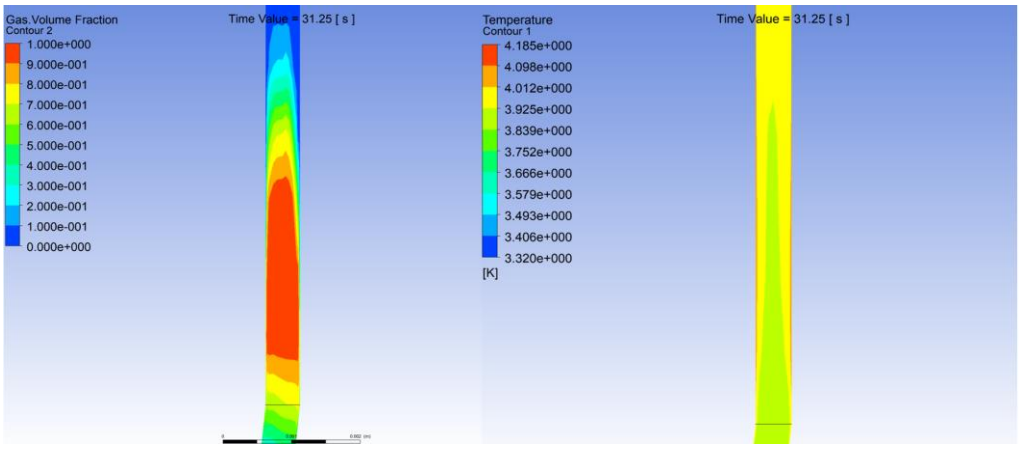

Figure 9-7: VOF contour plot (left) vs temperature contour plot (right) at Area 1

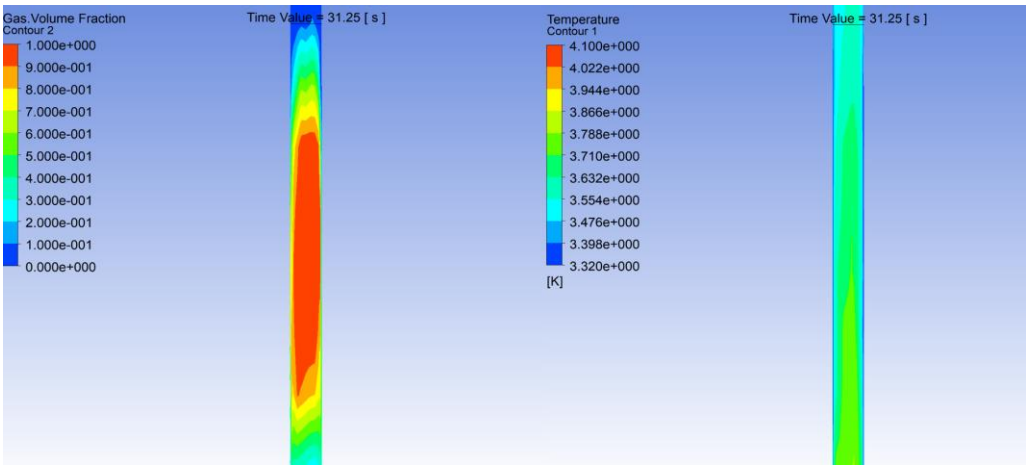

Figure 9-8: VOF contour plot (left) vs temperature contour plot (right) at Area 2

9.4 Copper block

It is common for experimental studies of cryogenic PHPs to use a copper block thermally connected to all the tubes in the evaporator section to apply heat, and a similar copper block thermally connected to all the tubes in the condenser to remove the heat from the PHP and deposit it into the cold sink. Without the copper block in the numerical simulation, the oscillating amplitude of the temperature profile is much larger than that observed in the experimental data.

However, by adding the copper block to the model, the temperature oscillation magnitude matches the experimental data. The very large thermal diffusivity of the copper block in the 4 K range thermally connects all the tubes. The comparison plots, without and with the copper block, are shown in Figure 9-9.

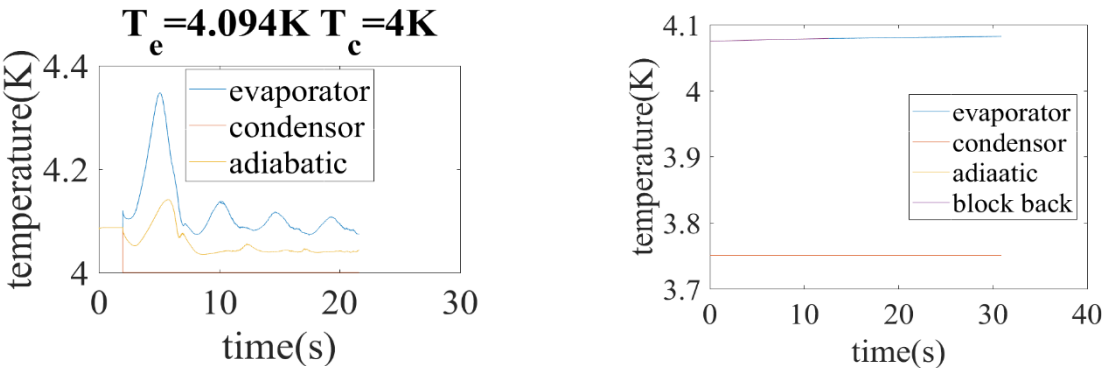

Figure 9-9: Temperature plot versus time without block and with block

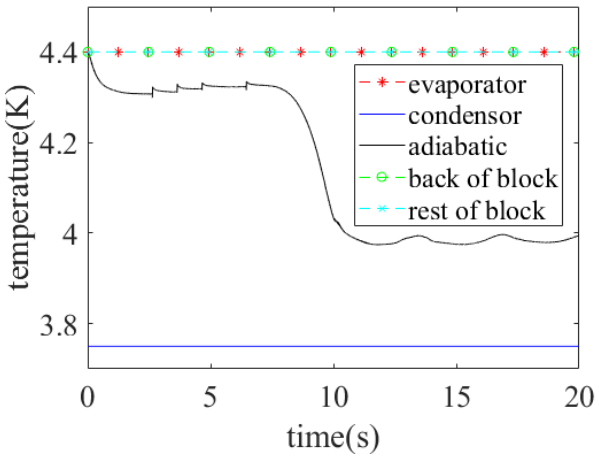

Figure 9-10: Temperature plot versus time with detailed block and evaporator information

After the copper block was added, the block temperature is identical to the wall temperature of evaporator as shown in Figure 9-10. This is due to the high thermal diffusivity of the copper. The “block back” in the plot means the back of the copper block on which surface the heat is applied. The “rest of block” in the plot means the rest of the copper block except the back of the copper block.

A comparison study was performed to investigate the reason the copper block smooths the temperature. The heat capacity of the copper block was artificially reduced by 1000 times and the result showed that the temperature oscillation amplitude was still small. In conclusion, the heat capacity does not contribute to smoothing out the temperature. Instead, the high thermal diffusivity of copper at cryogenic temperature is the reason. Specifically, the thermal diffusivity of RRR100 copper at $4K$ is $0.72m^2/s$. This value of thermal diffusivity was used to calculate the thermal wave propagation time which is 0.85 ms for 1m. It is interesting to point out that the thermal diffusivity of the copper at the room temperature is $0.0001137m^2/s$. This difference indicates that the copper might not have the same smoothing effect for room-temperature PHPs.

By adding the copper block to the PHP tubes, the thermal boundary condition on the evaporator tubes changes from uniform heat flux to spatially dependent heat flux boundary conditions which can be seen in Figure 9-11. The corresponding volume fraction plot is shown in Figure 9-12. These two plots together show that a small heat flux occurs in the regions that contain gaseous helium, while a large heat flux occurs in the regions that contain liquid helium. Moreover, with the added copper block, the evaporator temperature of the PHP takes 20 seconds to increase by 0.01K in the simulation. This observation matches the experimental data by Fonseca [14] as shown in Figure 9-13. This figure records the transition between different heat

flux/condenser temperature pairs. Steep temperature increases occur to the right of the white vertical dotted line in response to corresponding step increases in the applied heat. The experimental time for the complete temperature increase corresponds to a few minutes. Several minutes in simulation time requires a few months in computational time. Thus, the transition is unrealistic to model.

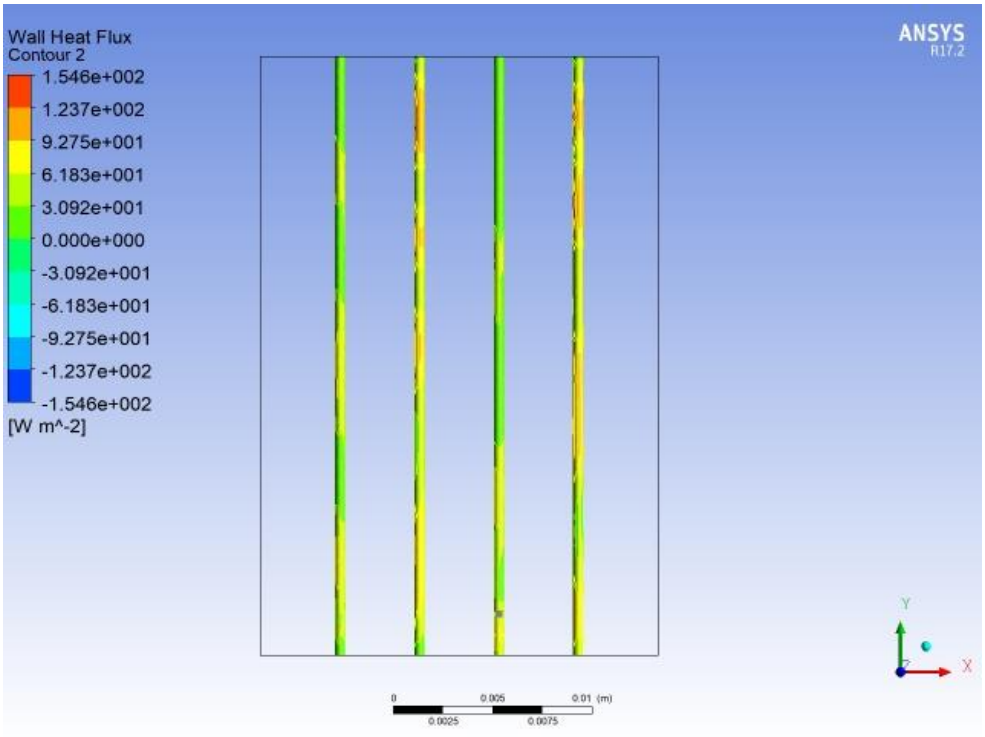

Figure 9-11: Heat flux on the wall of evaporator

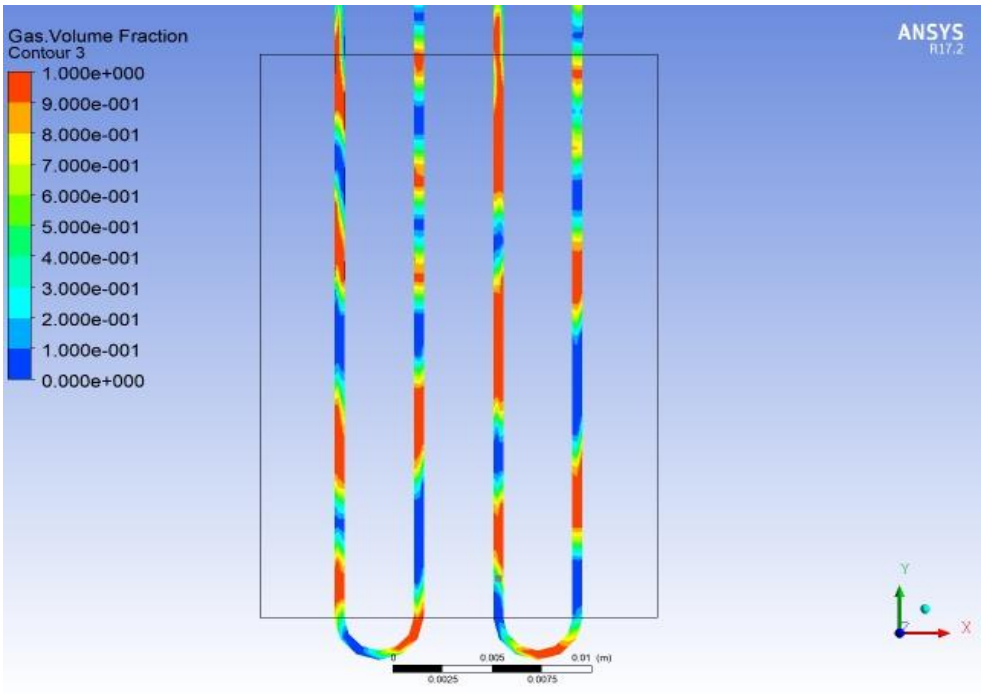

Figure 9-12: Corresponding gas VOF plot

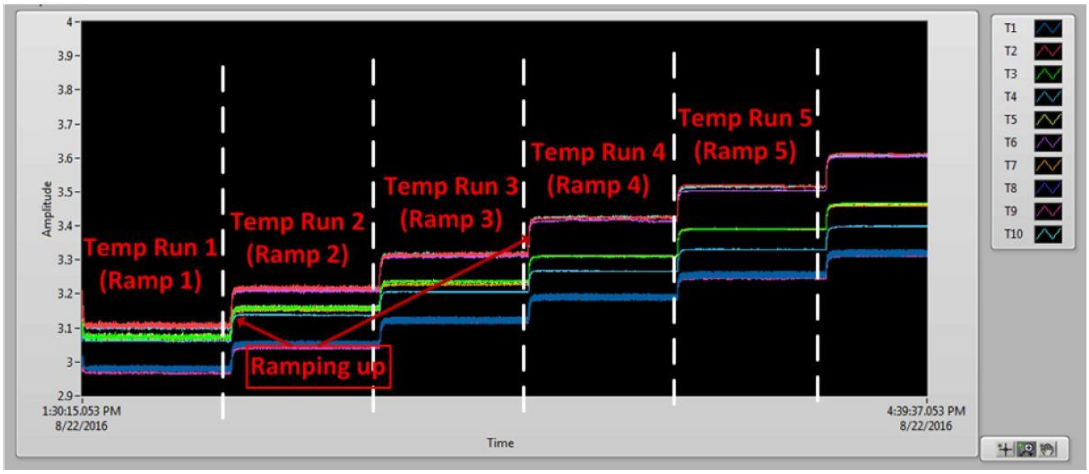

Figure 9-13: Temperature profile versus time and heat flux in experimental data

9.5 Time evolution of temperature contour plot

PHPs of all the various geometries and thermal conditions follow a similar time evolution in the temperature contour plot. An example of such a time evolution process for the 2 turn 50

mm PHP at a 70% fill ratio and with an 85 w/m^2 evaporator heat flux is displayed in Figure 9-14 to Figure 9-19. It is indeed the same geometry and thermal conditions as the time evolution of the VOF plot in Figure 6-1 to Figure 6-5. Additionally, both contour plots are plotted at the same time values. As a result, the reader can compare the temperature contour plots shown here to the VOF plots in section.

From 0-0.75 seconds in Figure 9-14 areas of high temperature are shown in the evaporator section while areas of low temperature are shown in the condenser section. This is because heat is added in and taken out of these respective sections.

From the 1 second to 1.75 second period in Figure 9-15, both areas of high temperature in the evaporator and areas of low temperature are moving counterclockwise. Comparing these to the corresponding VOF contour plots, both vapor plugs and liquid slugs are moving in this direction and moving with roughly the same velocity.

From the 2 second to 15 second period shown in Figure 9-16, Figure 9-17, and Figure 9-18, the temperature begins to smooth out and the variations inside of the domain diminish. At around 15-17 seconds, the temperature reaches steady state and the variations inside the domain remain constant as shown in Figure 9-19. Notice in Figure 9-19 that the temperatures of adjacent tubes in the adiabatic region alternate from warm to cool due to the unidirectional flow respectively leaving the evaporator or condenser.

Figure 9-20 plots the average temperature at different sections with respect to time. In this plot, it can also be clearly observed that the temperature of the adiabatic section begins to become steady at about 15-17 seconds.

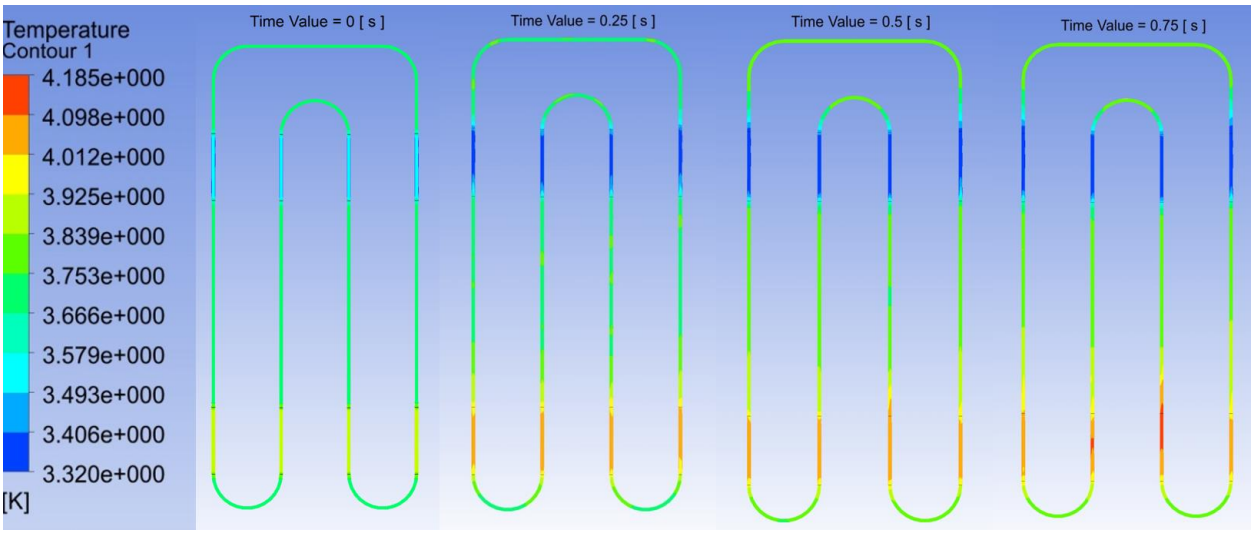

Figure 9-14: Time evolution of temperature plots for 2 turn – 50 mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 0-0.75 second

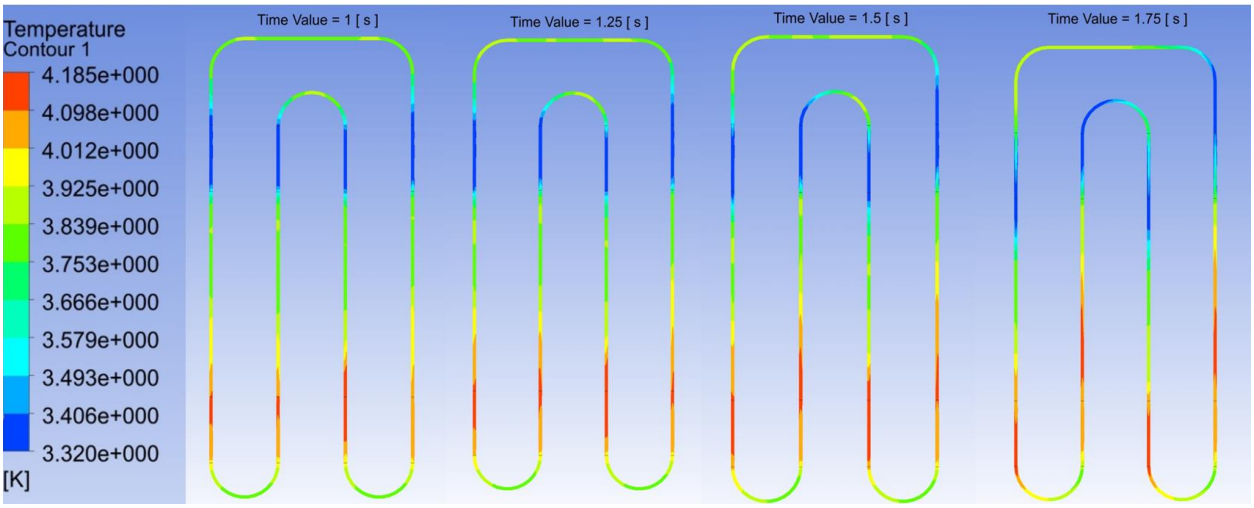

Figure 9-15: Time evolution of temperature plots for 2 turn – 50 mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 1-1.75 second

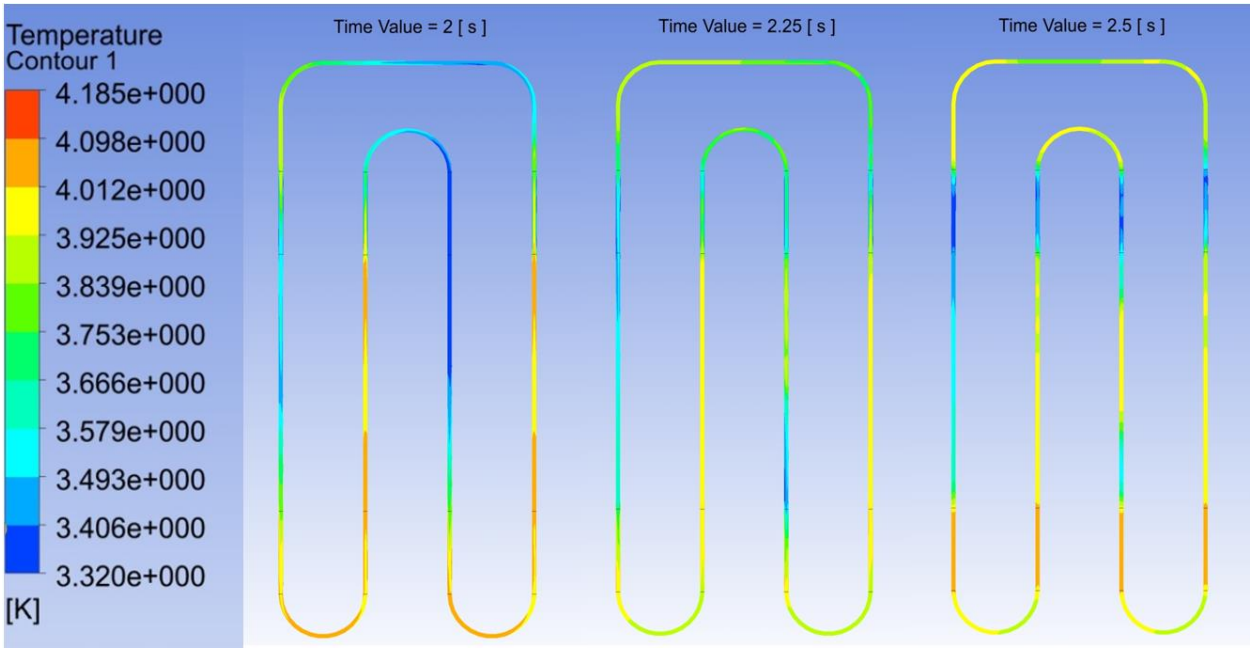

Figure 9-16: Time evolution of temperature plots for 2 turn – 50 mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 2-2.5 second

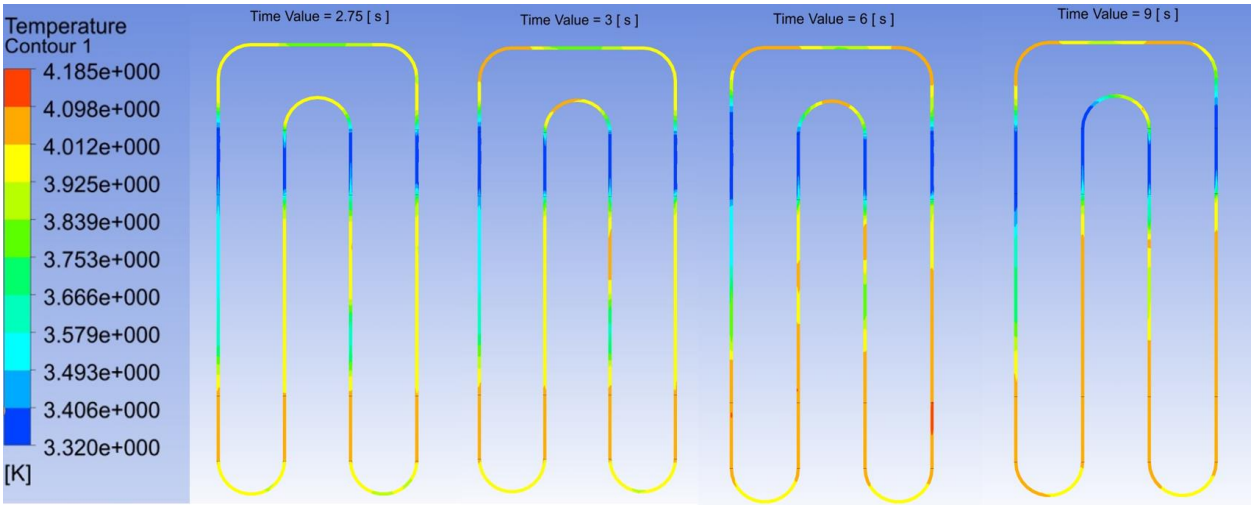

Figure 9-17: Time evolution of temperature plots for 2 turn – 50 mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 2.75-9 second

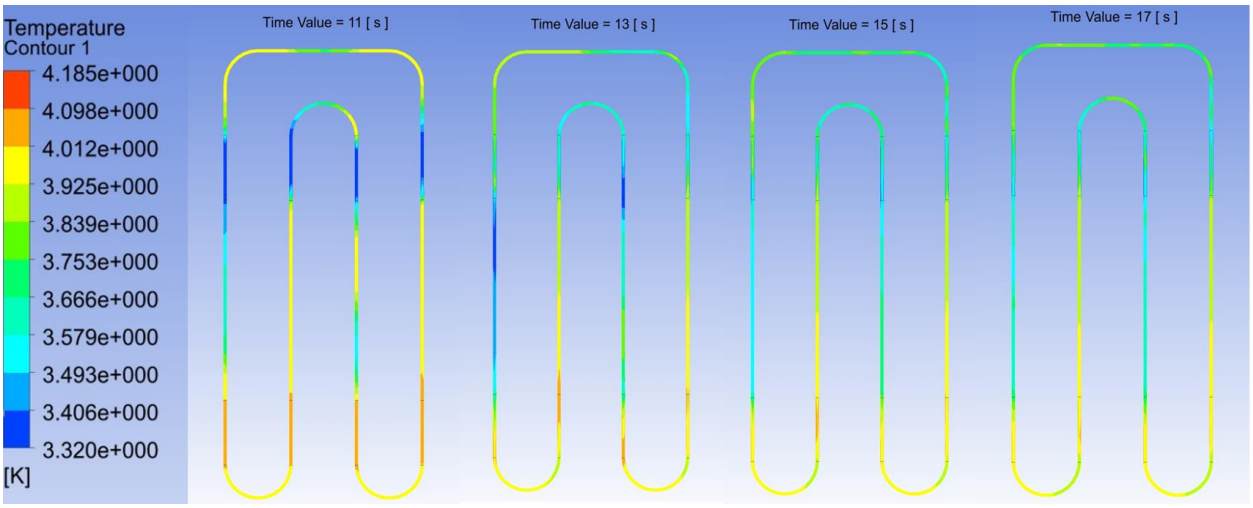

Figure 9-18: Time evolution of temperature plots for 2 turn – 50 mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 11-17 second

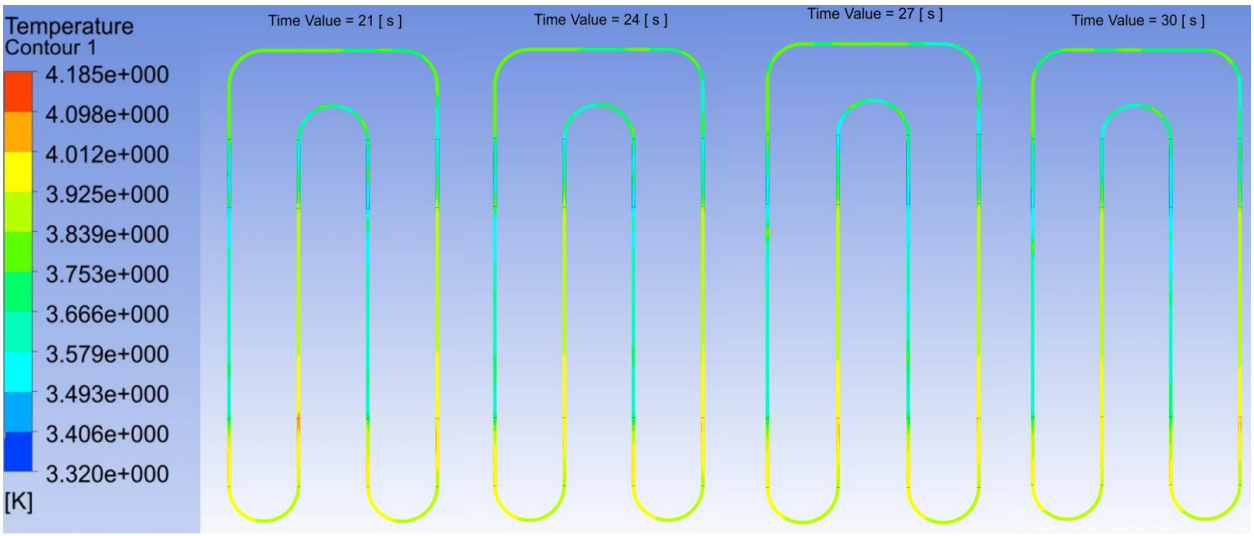

Figure 9-19: Time evolution of temperature plots for 2 turn – 50 mm PHP (85 w/m^2 heat flux and 70% fill ratio) from 21-30 second

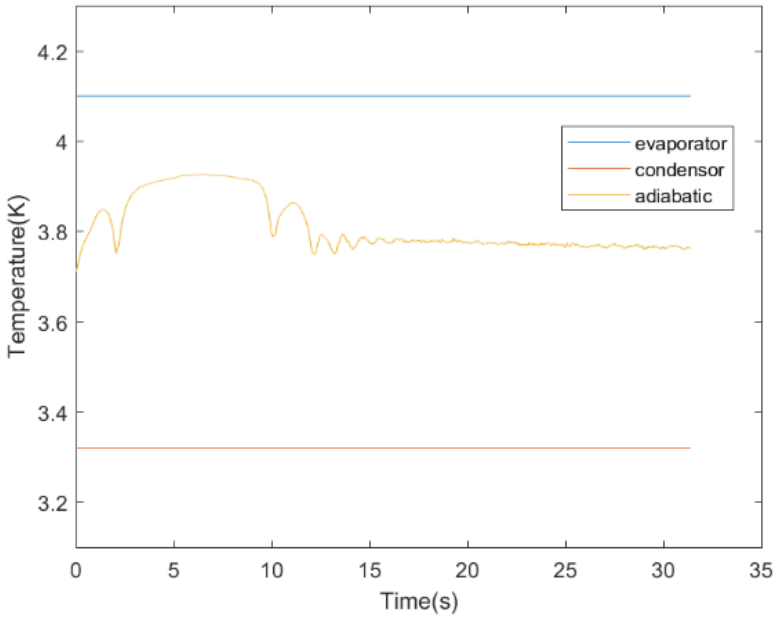

Figure 9-20: Time vs Wall-Averaged temperature 2 turn – 50 mm PHP (85 w/m^2 heat flux and 70% fill ratio)

9.5.1 Wall area-averaged temperature vs volume averaged temperature

The volume-averaged temperature versus time is shown in Figure 9-21 for a 2-turn 50-mm PHP. Comparing it to Figure 9-20 which shows the wall-averaged temperature versus time, it can be concluded that the value of the volume-averaged temperatures for the evaporator and for the condenser are between the wall–averaged temperatures and the adiabatic temperature. Thus, much of the thermal resistance between the evaporator and the condenser occurs at the walls of those components. This feature is consistent with recent experimental reports [11] demonstrating that only a small portion of the thermal resistance between the evaporator and condenser can be attributed to the thermal resistance through the fluid.

Finally, note that the volume-averaged temperature has more oscillation than the wall-averaged temperature.

In Figure 9-21, it can also be observed that each of the sections experiences oscillations and steady state conditions at around the same time.

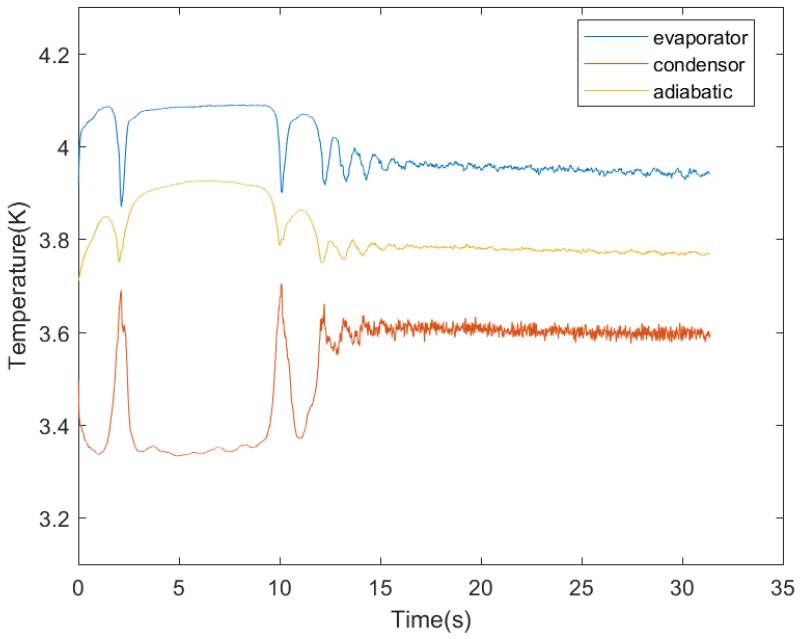

Figure 9-21: Volume-Averaged temperature vs Time for the 2 turn – 50 mm PHP (85 w/m^2 heat flux and 70% fill ratio)

9.6 Steady state temperature contour plots among different geometries

The steady state temperature contour plots among different geometries are shown in Figure 9-22 and Figure 9-23. The evaporator is the hottest part of the PHP, and the condenser is the coldest part of the temperature domain. It can be observed that the temperature is high and low in the alternating tubes of the adiabatic section of the PHPs. It can also be seen that the temperature of the horizontal tube on top is between the temperatures of the two sets of alternative tubes of the adiabatic section.

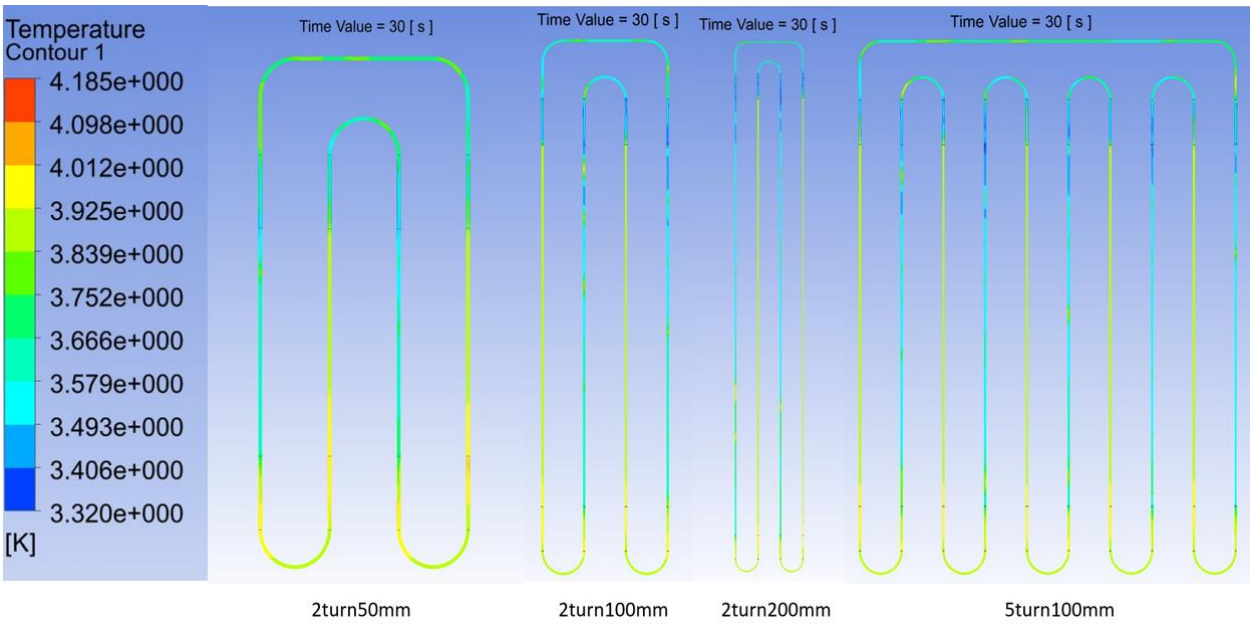

Figure 9-22: Steady state temperature visualization of different geometries with 85 w/m^2 heat flux and 70% fill ratio, part 1

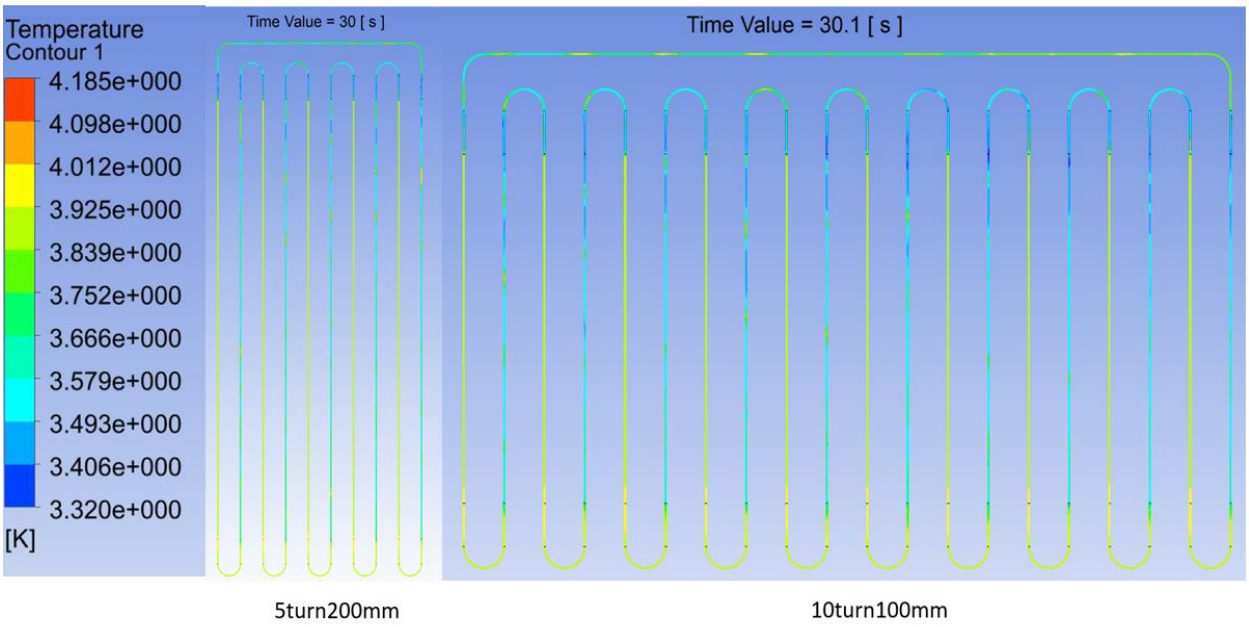

Figure 9-23: Steady state temperature visualization of different geometries with 85 w/m^2 heat flux and 70% fill ratio, part 2

9.7 Evaporator temperature trends

Among the various geometries with different turn number and length, the evaporator temperature remains nearly constant for a given heat flux and fill ratio. As demonstrated in Table 9-1, with a heat flux of 85.94 w/m^2 and a 70% fill ratio, most evaporator temperatures of different geometries are 4.0 K.

Table 9-2 and Table 9-3, the same trend applies for a heat flux of 136.5 $\mathrm{w/m^2}$and a 70% fill ratio as well as for a heat flux of 227.5 $\mathrm{w/m^2}$and a 70% fill ratio, with most evaporator temperatures for both conditions holding constant at 4.7 K.

Table 9-1: Evaporator temperature for heat flux 85.94 w/m^2 and 70% fill ratio

	2turn	**5turn**	**10turn**
50mm	4.1K	NAN	NAN

100mm	4.0K	4.0K	4.0K
200mm	4.0K	4.0K	4.0K

Table 9-2: Evaporator temperature for 136.5 w/m^2 and 70% fill ratio

	2turn	**5turn**	**10turn**
50mm	4.8K	NAN	NAN
100mm	4.7K	4.7K	4.7K
200mm	4.7K	4.5K	4.7K

Table 9-3: Evaporator temperature for 227.5 w/m^2 and 70% fill ratio

	2turn	**5turn**	**10turn**
50mm	NAN	NAN	NAN
100mm	4.4K	4.7K	4.9K
200mm	4.7K	4.7K	4.7K

9.8 Effective thermal conductivity trends

Values of effective thermal conductivity are calculated by the following equation:

$$k_{eff} = \frac{QL_{adia}}{NA_c(T_{eva} - T_{cond})}$$

where Q is the heat applied to evaporator, L_{adia} is the adiabatic length, N is the number of tubes, A_c is the cross-sectional area of the tube, T_{eva} is the wall temperature of the evaporator, and T_{cond} is the wall temperature of the condenser.

For a given fill ratio and geometry, effective thermal conductivity increases with evaporator heat flux as shown in Figure 9-24.

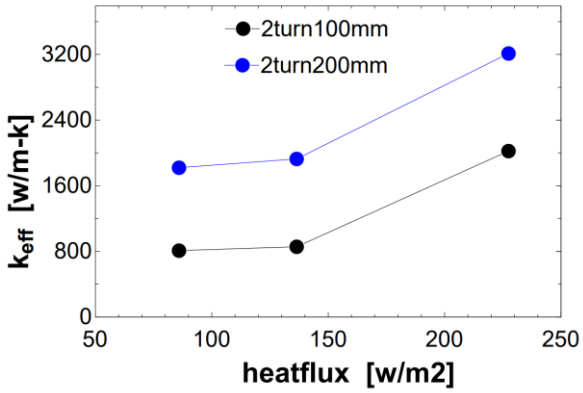

Figure 9-24: Effective thermal conductivities at a 70% fill ratio with 2 - turns

The above figure also shows that the effective thermal conductivity increases with total length. The trend is not only valid for the 2-turn geometry, but is also valid for both the 5-turn and 10-turn cases as shown in Figure 9-25 and Figure 9-26.

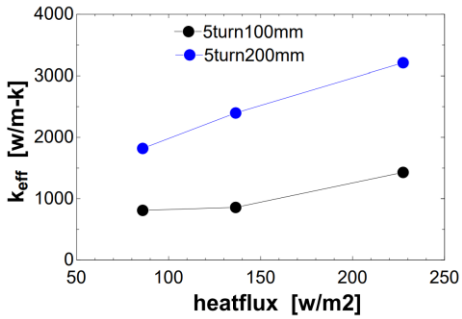

Figure 9-25: Effective thermal conductivities with a 70% fill ratio and 5 turns

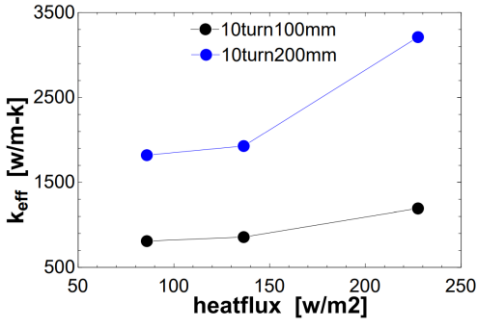

Figure 9-26: Effective thermal conductivities with a 70% fill ratio with 10 turns

By contrast, values of effective thermal conductivity are not influenced by the number of turns as illustrated in Figure 9-27 and Figure 9-28. This trend is similar to that displayed by the

experimental results by Li, Li, & Xu [9] as shown in Figure 9-29. The heat load per turn represented in the x-axis is equivalent to a heat flux. Notice that PHPs with different turn numbers have similar effective thermal conductivity when plotted against the heat load per turn or heat flux.

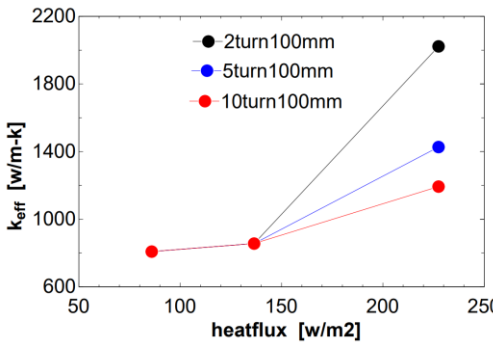

Figure 9-27: Effective thermal conductivities at 70% fill ratio with 100mm

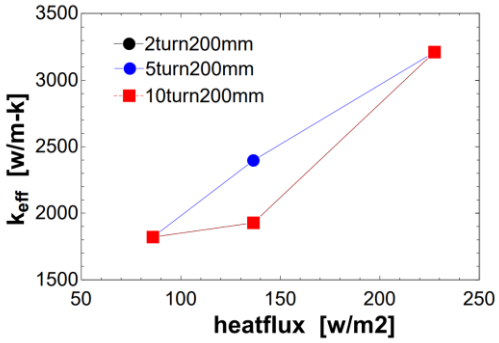

Figure 9-28: Effective thermal conductivities at 70% fill ratio with 200mm

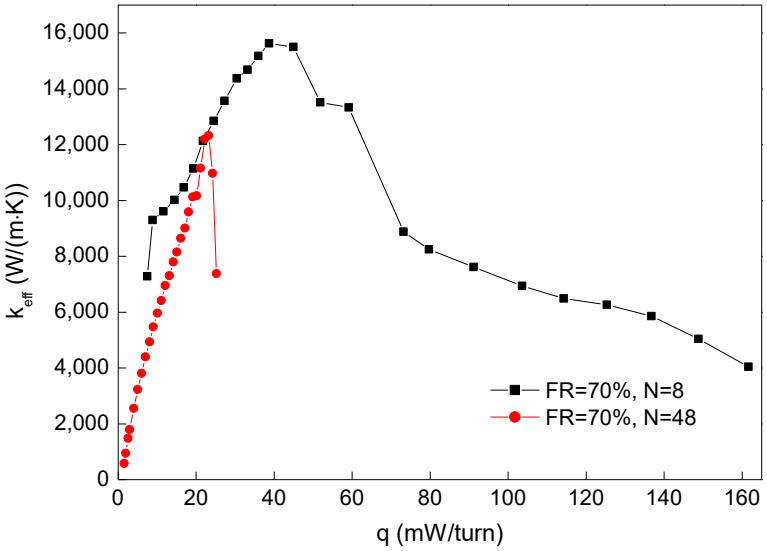

Figure 9-29: Effective thermal conductivity versus heat load per turn

Furthermore, the fill ratio has a varying influence on the effective thermal conductivity of the PHP depending on the heat flux that is applied. Figure 9-30, Figure 9-31, and Figure 9-32 show that, with most heat flux and geometry pairs, the fill ratio has a positive effect on the effective thermal conductivity. In contrast, Figure 9-32 shows that at $85 w/m^2$ with the 10turn – 100 mm geometry, the fill ratio has a minimal effect on the effective thermal conductivity. In most cases, a higher fill ratio leads to higher values of effective thermal conductivity.

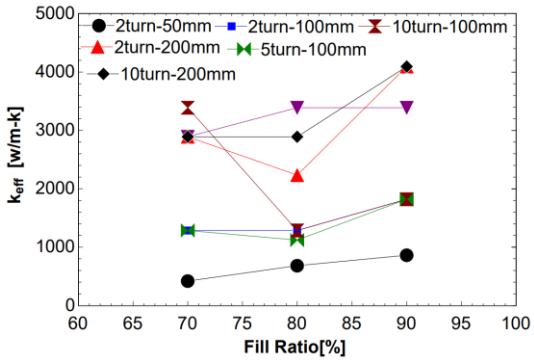

Figure 9-30: Effective thermal conductivities with 85 w/m^2

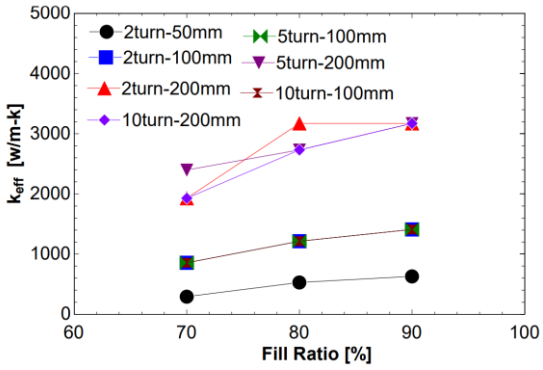

Figure 9-31: Effective thermal conductivities with 136.5w/m^2

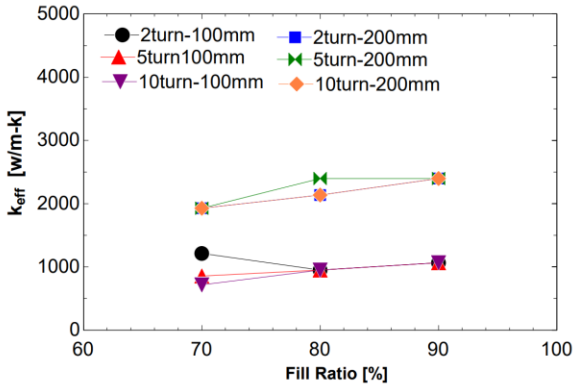

Figure 9-32: Effective thermal conductivities with 227 w/m^2

10 Pressure

10.1 Pressure plot versus time

Pressure waves travel fast inside of the PHP. As a result, the pressure change throughout the domain is small compared to the pressure change with time. The volume-averaged pressure of the domain changes from the starting guess pressure to a value that is close to the saturation pressure corresponding to the temperature in the adiabatic region. In fact, after the system reaches equilibrium, the pressure change with time is relatively small compared to the displayed variations before it reaches equilibrium. All this can be observed from various PHPs that were modeled. In this chapter, a 10 turn – 100 mm PHP with a 70% fill ratio and 85 w/m^2 heat flux is analyzed because the pressure wave is most clear when visualized with the large geometry. All the above conclusions are demonstrated with this example. Moreover, similar phenomena can be seen with other PHPs.

The volume-averaged pressure for the three sections of the 10 turn 100 mm PHP can be seen in Figure 10-1. Firstly, the variations between sections are much smaller than the pressure variations with time.

Secondly, at the end of 30 seconds, the volume averaged adiabatic pressure has a value of 73722 Pa. The corresponding saturation temperature at pressure 73722 Pa for helium is 3.90 K. Not surprisingly, the adiabatic temperature of this PHP at 30 second is 3.96 K. Consequently, it can be concluded that the saturation temperature associated with the adiabatic pressure is close to the adiabatic temperature of the PHP after the system reaches steady state.

Thirdly, the variation in pressure after the PHP reaches steady state is much smaller than the variations in the beginning.

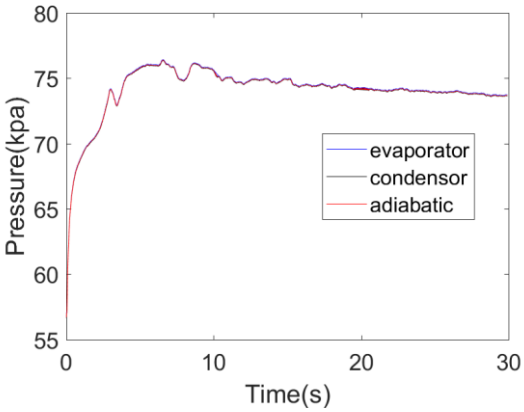

Figure 10-1: Volume-averaged pressure plot of the three sections for the 10 turn – 100 mm PHP with a 70% fill ratio and an applied heat flux of 85 w/m^2

In addition, by enlarging the pressure plot within the 25-30 seconds region as shown in Figure 10-2, both triangular and saw-tooth oscillations can be observed. The frequency of the pressure wave can be calculated, and it is 2.2 Hz in this case.

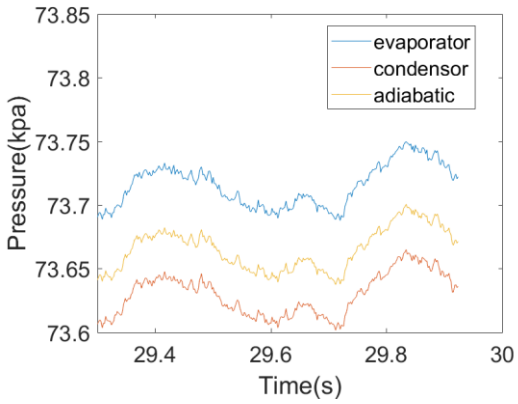

Figure 10-2: Volume-averaged pressure plot of three sections from 25-30 second (10turn - 100mm PHP with 70% fill ratio and 85 w/m^2heat flux)

10.2 Pressure visualization plot

The pressure is initialized uniformly inside of the PHP. However, after as short as 0.25 seconds, pressure layers in the domain can be observed due to the force of gravity. The pressure at bottom layer is larger than the pressure in the top layer as shown in Figure 10-3.

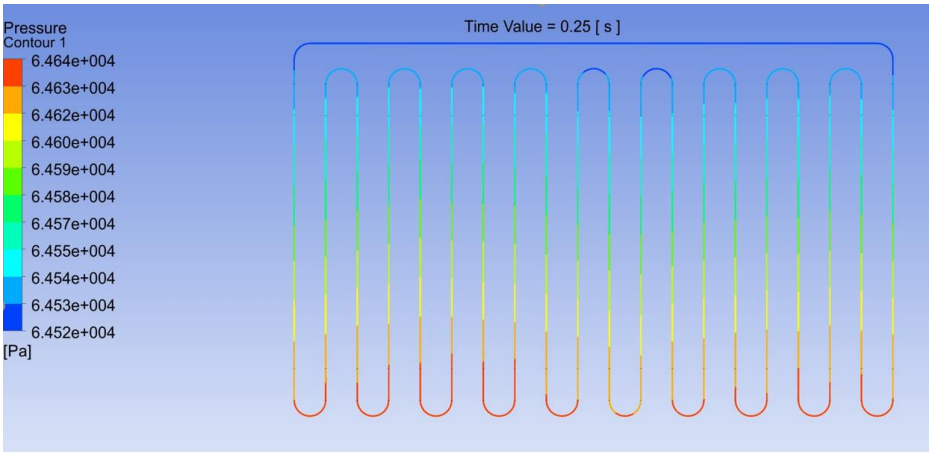

Figure 10-3: Pressure visualization plot at 0.25 second (10turn - 100mm PHP with 70% fill ratio and 85 w/m^2heat flux)

Figure 10-4 shows the pressure visualization plot after the PHP reaches steady state at 29.3 seconds. After the PHP reaches equilibrium, the pressure is still highest at the bottom and lowest on the top. The values are plotted with smooth contours to clearly see the pressure variation and gradients inside of the PHP.

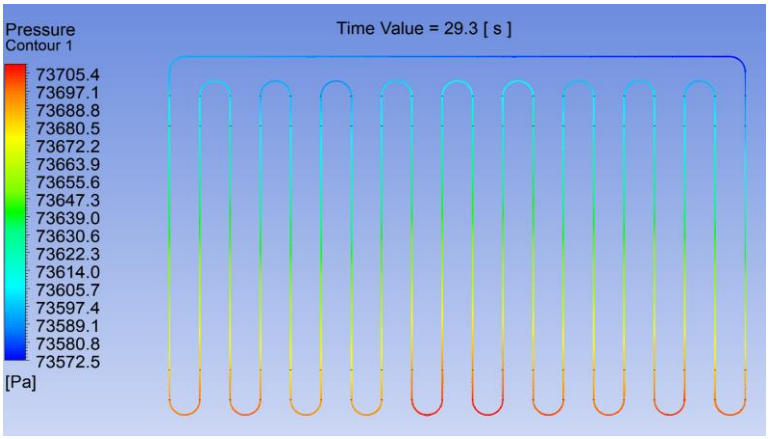

Figure 10-4: Pressure visualization plot at 29.3 seconds for the 10 turn – 100 mm PHP with a 70% fill ratio and applied heat flux of 85 w/m^2)

Figure 10-5, Figure 10-6, Figure 10-7, and Figure 10-8 show the visualization of pressure from 29.3 to 29.9 seconds. It is shown with sharper contours compared to the figure above so that the pressure wave can be clearly visualized. In these plots, the interface of different colors represents the change of pressure. Spatially varying sinusoidal waves can be clearly observed from these plots. Some of the sine waves are small in amplitude as shown in Figure 10-5, while others are large as shown in Figure 10-8. Some of the sine waves are not complete from the left side of the PHP to the right side due to being on top of the PHP and running out of space as shown in Figure 10-6.

Additionally, it can also be seen that from 29.3 to 29.5 seconds, a line of constant pressure moves from the bottom to the top of the PHP. The same behavior can be observed from 29.7 to 29.9 seconds.

The pressure versus time plot from 29.3 to 29.9 second as shown in Figure 10-9 displays another version of the same data. The pressures in all three sections are rising from 29.3 to 29.5 seconds and are rising again from 29.7 to 29.9 second.

To summarize, pressure waves of varying amplitude are found to extend across the entire PHP and travel from bottom to the top and from turn to turn.

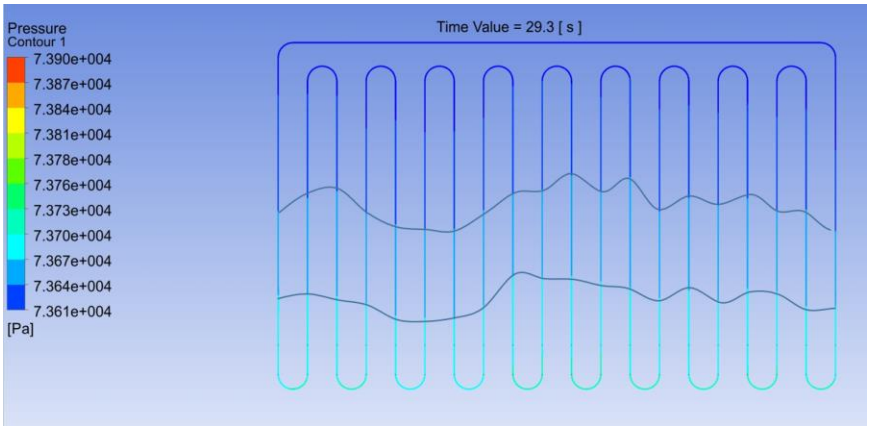

Figure 10-5: Pressure visualization plot at 29.3 second with pressure waves (10turn - 100mm PHP with 70% fill ratio and 85 w/m^2heat flux)

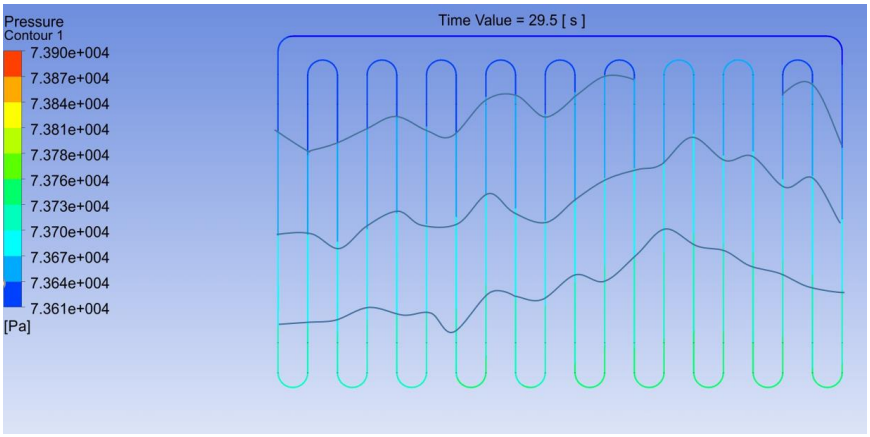

Figure 10-6: Pressure visualization plot at 29.5 second with pressure waves (10turn - 100mm PHP with 70% fill ratio and 85 w/m^2 heat flux)

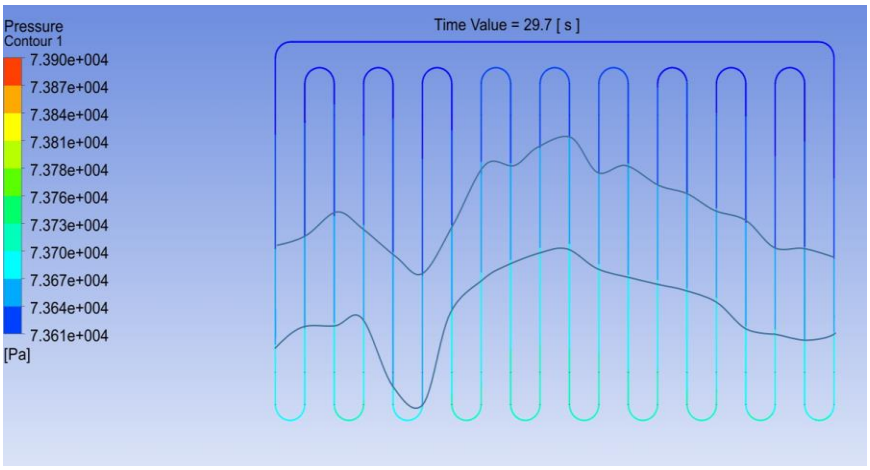

Figure 10-7: Pressure visualization plot at 29.7 second with pressure waves (10turn - 100mm PHP with 70% fill ratio and 85 w/m^2heat flux)

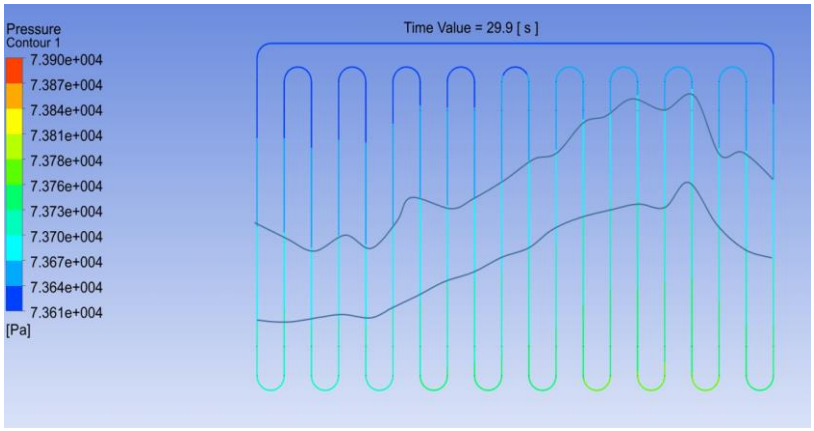

Figure 10-8: Pressure visualization plot at 29.9 second with pressure waves (10turn - 100mm PHP with 70% fill ratio and 85 w/m^2 heat flux)

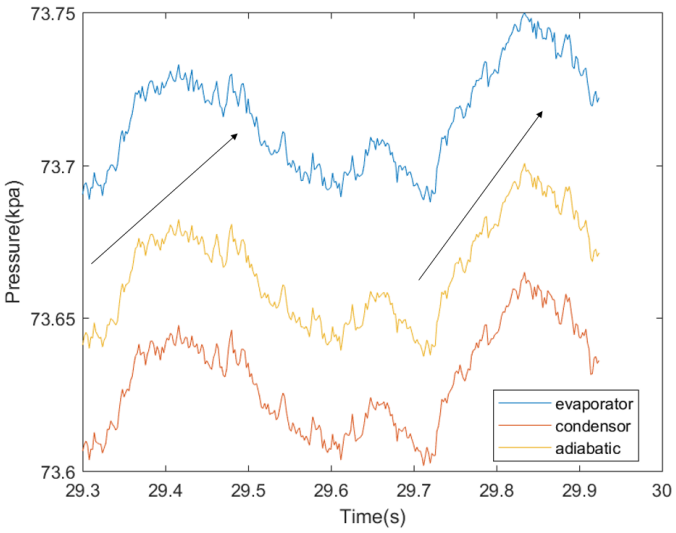

Figure 10-9: Volume-averaged pressure plot of three sections from 25-30 seconds (10 turn –100 mm PHP with a 70% fill ratio and 85 $w/m^2$of applied heat flux)

11 Mass transfer rate plot

11.1 Mass transfer rate versus time

Mass transfer refers to both the evaporation and the condensation that occur in a PHP. In the numerical model, the volume-averaged mass transfer rate for the evaporator is defined in the following equations:

$$masstransfer_{evaporator} = \frac{\sum masstransfer_{cell} \cdot Volume_{cell}}{Volume_{evaporator}} \qquad 11\text{-}1$$

The mass transfer rate for each cell in the evaporator is defined in Equation 11-1. The same definition applies to the condenser. The plot of mass transfer rate versus time is shown in Figure 11-1. The system reaches steady state after roughly 15 seconds and the evaporation and condensation rates remain constant thereafter.

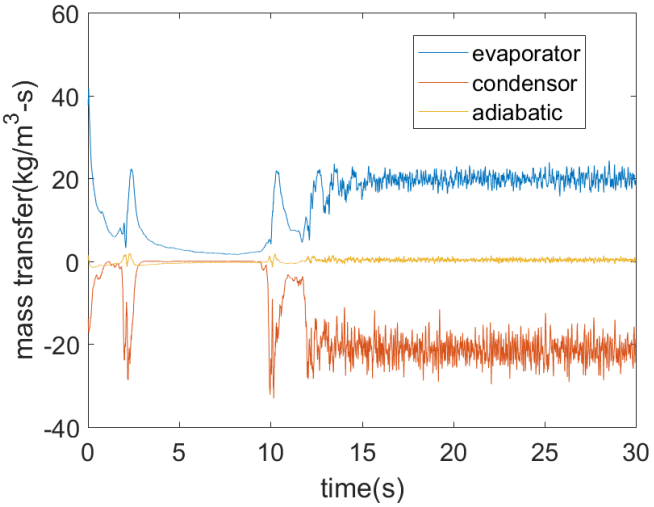

Figure 11-1: Mass transfer rate versus time for 2turn - 50mm PHP (70% fill ratio and 85w/m^2)

11.2 Mass transfer rate versus heat flux

The mass transfer rate versus heat flux is plotted in Figure 11-2 to investigate the influence of heat flux on mass transfer rate. The mass transfer rate in the plot is averaged over all the geometries and fill ratios, but all for the same heat flux level. It can be observed from the plot that the higher the heat flux, the larger the mass transfer rate. The percentage of the total heat transfer due to latent heat is also plotted using the right-hand-axis of the figure. Even though the mass transfer rate increases with heat flux, the percentage of the total heat due to latent heat decreases with increasing heat flux. A possible explanation is that the velocity increases with heat flux, and thus increases the rate of heat transfer via sensible heat.

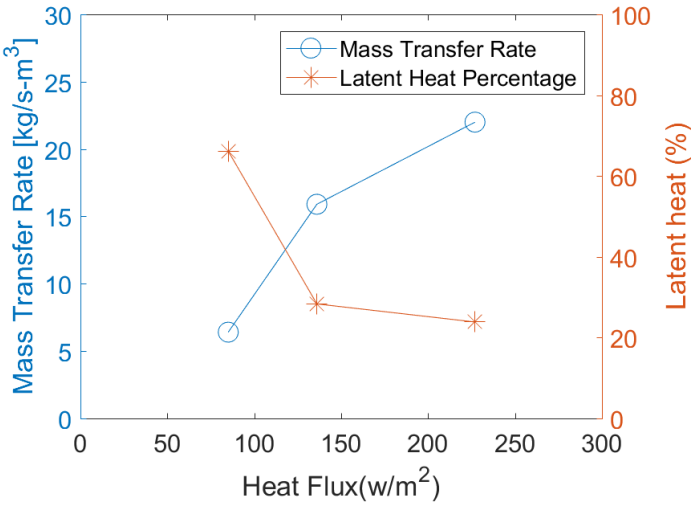

Figure 11-2: Mass transfer rate (Left)/Latent heat percentage (Right) versus heat flux for 5turn – 100mm PHPs

11.3 Mass transfer rate versus fill ratio

The mass transfer rate versus fill ratio is plotted in Figure 11-3. It can be concluded from these results that higher fill ratios correspond to slightly lower mass transfer rates. The corresponding percentages of latent heat were plotted on the right-hand axis.

A higher fill ratio generally leads to a higher effective thermal conductivity as shown in Figure 9-30, Figure 9-31, and Figure 9-32. Furthermore, a higher fill ratio leads to a lower evaporator temperature, and a lower evaporator temperature leads to a lower mass transfer rate as shown in Equation 2-17. Finally, as shown in Figure 11-3, lower mass transfer rate naturally leads to a lower percentage of latent heat.

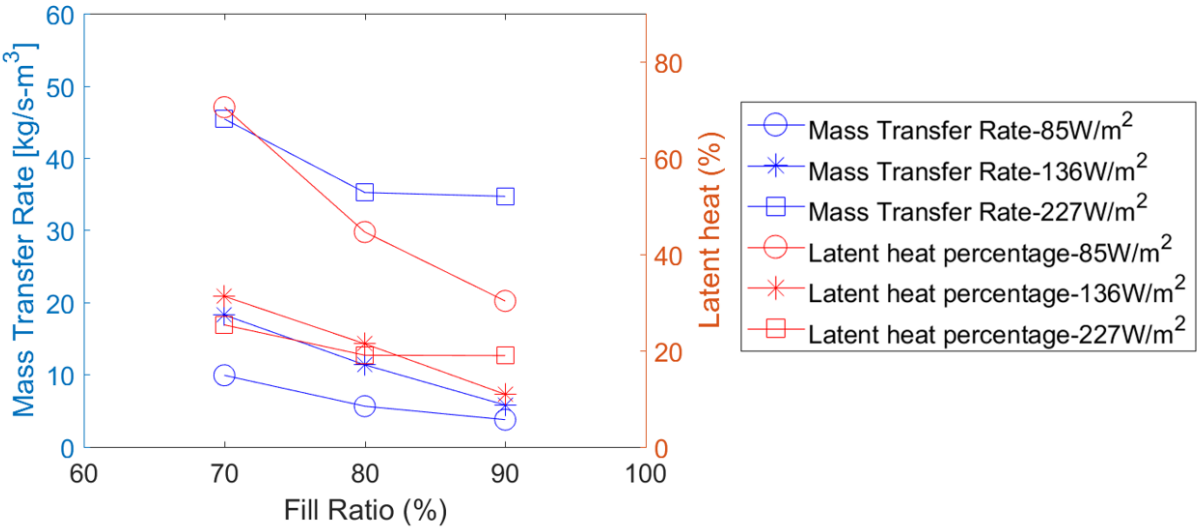

Figure 11-3 Mass transfer rate (Left)/Latent heat percentage (Right) versus fill ratio for 5turn – 200mm PHPs

11.4 Mass transfer rate visualization

Figure 11-4 shows a contour plot of the mass transfer rate inside of the PHP. Most of the mass transfer occurs near the wall as might be expected noting that evaporation typically occurs near a surface. It is also true that the temperature of the fluid near the wall is higher than the temperature of the fluid near the centerline.

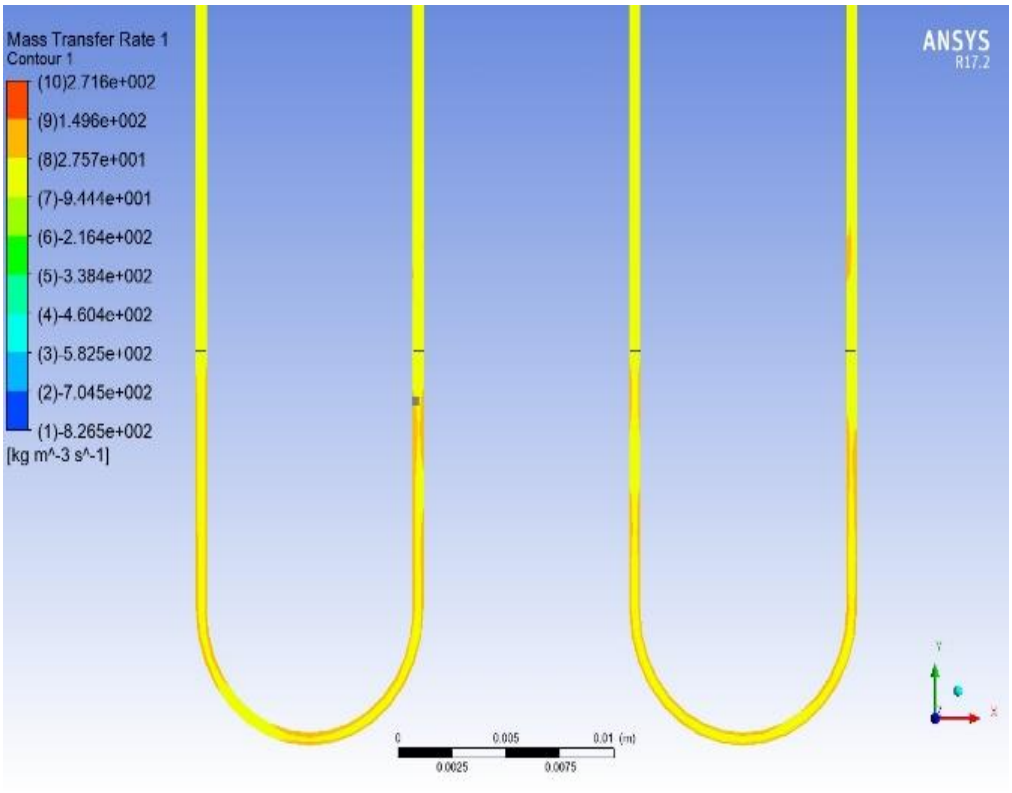

Figure 11-4: Mass transfer rate contour plot of evaporator section

12 Conclusion and future work

12.1 Conclusion

After verifying the model with experimental data, a parametric study was carried out to investigate the effects of the number of turns, total length, fill ratio, and evaporator heat flux on the flow pattern, circulatory pattern, velocity, frequency, and thermal conductivity of a helium PHPs. The results can be summarized below:

- The numerical start-up process of PHPs and the associated shrinkage, breakdown, growth and merger of vapor plugs can be clearly visualized.
- The flow throughout the parametric study is laminar

- The flow pattern for most geometries, fill ratio and heat flux level are slug/plug flow. However, for 70% fill ratio, 227 w/m^2 and for the 10 turn-100 mm geometry, the flow pattern changed to annular flow mixed with slug/plug flow.
- A higher fill ratio leads to longer liquid slugs.
- Two circulatory patterns exist: steady circulatory flow and pulsating circulatory flow.
- The fill ratio is the only parameter that has a significant effect on the circulatory pattern. Higher fill ratios lead to a higher possibility of steady circulatory flow.
- A higher heat flux produces a higher velocity, a higher frequency of pulsating flow, and a higher effective thermal conductivity. Longer lengths result in lower frequencies.
- The evaporator temperature varies little among different geometries with the same evaporator heat and fill ratio. However, the larger the applied heat flux, the larger the temperature variation there is among geometries.
- The effective thermal conductivity increases with increasing evaporator heat flux and total length. In contrast, the number of turns has very little influence on the effective thermal conductivity, and the fill ratio has a mixed effect on the effective thermal conductivity.
- As the heat flux increases, so does the mass transfer rate. At the same time, larger values of heat flux result in a lower percentage of the total heat being transferred via latent heat.
- The higher the fill ratio, the lower the mass transfer rate, and the lower the percentage of latent heat.

12.2 Future work

12.2.1 Improve temperature range

For this 3D model, the investigated temperature range is limited by the temperature range of the experimental data that was used to generate and validate the Lee model frequency correlation. In the future, with more experimental data, it will be very useful to extend the temperature range further, even beyond helium's critical temperature.

12.2.2 Explain some of the phenomenon discovered by Diego's paper

After the temperature is extended, the temperature excursions that are present in Fonseca's experimental data [1] as shown in Figure 12-1 and Figure 12-2 should be explored.

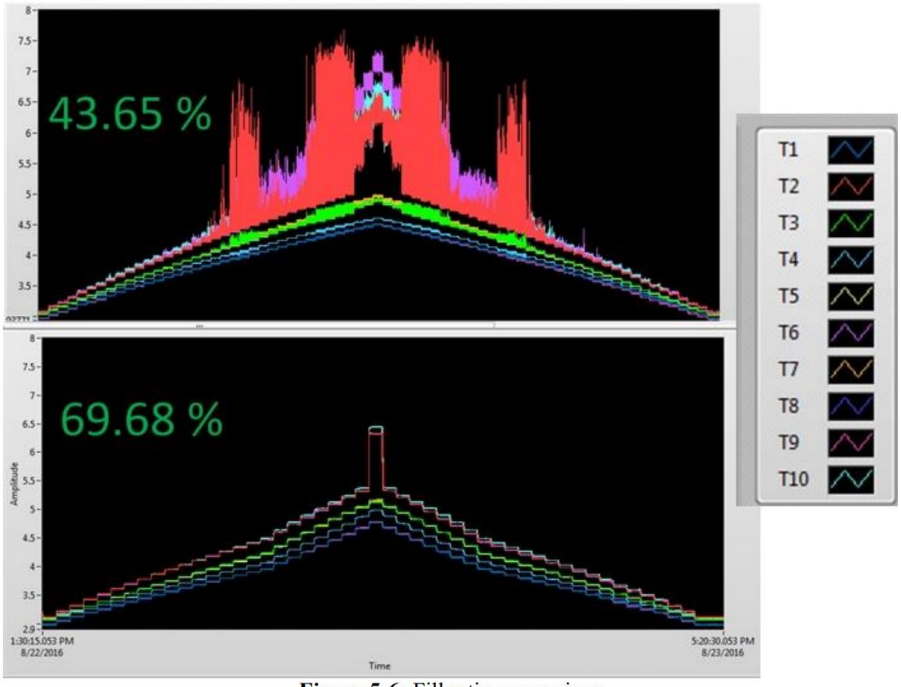

Figure 5-6: Fill ratio comparison

Figure 12-1: Temperature versus heat load plots for 43% and 70% fill ratio

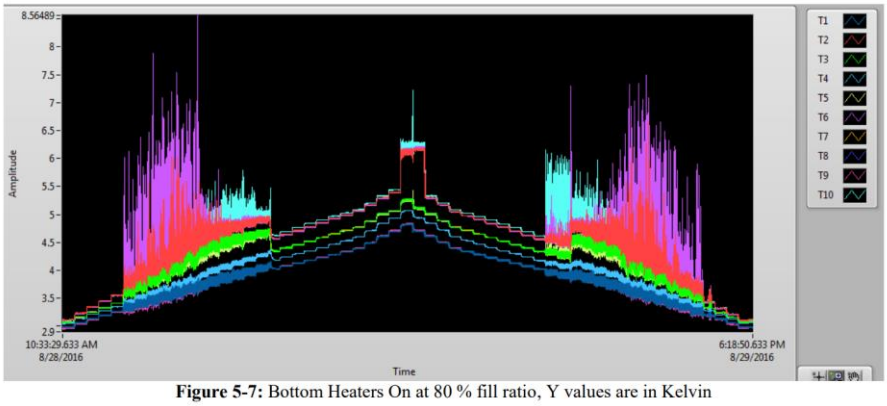

Figure 5-7: Bottom Heaters On at 80 % fill ratio, Y values are in Kelvin

Figure 12-2: Temperature versus heat load plots for 80% fill ratio

12.2.3 Investigate additional parameters

As presented in various experimental reports (see for example [5] and [6]), the magnitude of gravity and the tilt angle of a PHP also influences the performance of the PHP. The future work should investigate these two additional parameters.

12.2.4 Develop a design tool

Currently, the development of the fluent model could provide a design tool to predict the effective thermal conductivity given certain geometry and thermal conditions. Due to the limitations of the HPC system and ANSYS, different tasks are performed with different software. ANSYS is used to define the geometry and mesh while the HPC is used for applying physical settings and running the simulation.

Four steps are envisioned in using the model for design purposes. First, geometry parameters such as length, number of turns, and bending radius are entered into the ANSYS workbench to create the geometry and appropriate mesh. Second, the evaporator, adiabatic and condenser

sections are separated in Fluent, and a case file is exported. Third, the case file is transferred into an HPC system, and the geometry parameters and thermal boundary conditions are entered into a journal file. Finally, the simulation is run with various values of applied heat flux at the evaporator and fixed temperatures at the condenser to obtain the corresponding effective thermal conductivity of the PHP.

The opportunity is offered for future attempts to predict with this design tool. However, ANSYS requires a significant time investment to master, and a High-Performance Computing center is a rare resource that few can obtain.

13 Reference

[1] D. Fonseca, "Experimental Characterization of Cryogenic Helium Pulsating Heat Pipes," PHD thesis, Dept. Mech., UW-Madison, Madison,WI, 2016.

[2] B. Mueller, "The performance of pulsating heat pipes as thermal switches in redundant cryocooler systems," PHD thesis, Dept. Mech., UW-Madison, madison,WI, 2021.

[3] K. Tang, "The study of heat transfer mechanism and modeling of cryogenic PHP," M.S. Thesis, Dept.Mech, University of Shanghai for Science and Technology, Shanghai, 2016.

[4] L. D. Fonseca, F. Miller and J. Pfotenhauer, "A helium based pulsating heat pipe for superconducting magnets," in *AIP Conference*, 2014.

[5] T. Mito, K. Natsume, N. Yanagi, H. Tamura, T. Tamada, K. Shikimachi, N. Hirano and S. Nagaya, "Development of highly effective cooling technology for a superconducting magnet using cryogenic OHP," *IEEE Transactions on applied superconductivity*, vol. 20, no. 3, pp. 2023-2026, 2010.

[6] F. Bonnet, P. Gully and V. Nikolayev, "Development and test of a cryogenic pulsating heat pipe and a pre-cooling system," in *AIP Conference Proceedings*, Grenoble, France, 2012.

[7] X. Han, H. Ma, A. Jiao and J. K. Critser, "Investigations on the heat transport capability of a cryogenic oscillating heat pipe and its application in achieving ultra-fast cooling rates for cell vitrification cryopreservation," *Cryobiology*, vol. 56, no. 3, pp. 195-203, 6 2008.

[8] D. Xu, L. Li and H. Liu, "Experimental investigation on the thermal performance of helium based cryogenic pulsating heat pipe," *Experimental Thermal and Fluid Science*, vol. 70, p. 61, 2016.

[9] M. Li, L. Li and D. Xu, "Effect of number of turns and configurations on the heat transfer performance of helium cryogenic pulsating heat pipe," *Cryogenics*, vol. 96, pp. 159-165, 1 12 2018.

[10] M. Li, D. Xu and L. Li, "Effect of filling ratio and orientation on the performance of a multiple turns," *Cryogenics*, vol. 100, p. 62, 2019.

[11] J. M. pfotenhauer, X. Sun, A. Berryhill and C. B. Shoemaker, "The influence of aspect ratio on the thermal performance of a cryogenic pulsating heat pipe," *Applied Thermal Engineering*, vol. 196, p. 117322, 2021.

[12] D. Xu, H. Liu, L. Li, R. Huang and W. Wang, "Theoretical research of helium pulsating heat pipe," in *IOP Conference Series: Materials Science and Engineering*, 2016.

[13] M. Li, R. Huang, D. Xu and L. Li, "Theoretical analysis of start-up power in helium pulsating heat pipe," in *IOP Conference Series: Materials Science and Engineering*, 2017.

[14] X. Sun, J. Pfotenhauer, B. Jiao, L. D. Fonseca, D. Han and Z. Gan, "Investigation on the temperature dependence of filling ratio in cryogenic pulsating heat pipes," *International Journal of Heat and Mass Transfer,* vol. 126, p. 237, 2018.

[15] R. Senjaya and T. Inoue, "Oscillating heat pipe simulation considering bubble generation Part I: Presentation of the model and effects of a bubble generation," *International Journal of Heat and Mass Transfer*, vol. 60, pp. 816-824, 1 1 2013.

[16] B. Fadhl, L. C. Wrobel and H. Jouhara, "Modelling of the thermal behaviour of heat pipes," in *WIT Transactions on Engineering Sciences*, 2014.

[17] J. Wang, H. Ma, Q. Zhu, Y. Dong and K. Yue, "Numerical and experimental investigation of pulsating heat pipes with corrugated configuration," *Applied Thermal Engineering*, vol. 102, pp. 158-166, 5 6 2016.

[18] Q. Li, C. Wang, Y. Wang, Z. Wang, H. Li and C. Lian, "Study on the effect of the adiabatic section parameters on the performance of pulsating heat pipes," *Applied Thermal Engineering*, vol. 180, p. 115813, 2020.

[19] J. V. Suresh and P. Bhramara, "CFD Analysis of Multi turn Pulsating Heat pipe," in *5th International Conference of Materials Processing and Characterization*, 2017.

[20] S. M. Pouryoussefi and Y. Zhang, "Analysis of chaotic flow in a 2D multi-turn closed-loop pulsating heat pipe," *Applied Thermal Engineering*, vol. 126, pp. 1069-1076, 2017.

[21] D.-T. Vo, H.-T. Kim, J. Ko and K.-H. Bang, "An experiment and three-dimensional numerical simulation of pulsating heat pipes," *International Journal of Heat and Mass Transfer*, vol. 150, p. 119317, 2020.

[22] S. R. Sagar, H. B. Naik and M. B. Mehta, "Numerical study of liquid nitrogen based pulsating heat pipe for cooling superconductors," *International Journal of Refrigeration,* vol. 122, p. 33, 2021.

[23] B. Y. Tong, T. N. Wong and K. T. Ooi, "Closed-loop pulsating heat pipe," *Applied Thermal Engineering,* vol. 21, pp. 1845-1862, 2001.

[24] S. Khandekar, P. Charoensawan, M. Groll and P. Terdtoon, "Closed loop pulsating heat pipes - Part B: Visualization and semi-empirical modeling," *Applied Thermal Engineering,* vol. 23, no. 16, pp. 2021-2033, 11 2003.

[25] J. L. Xu, Y. X. Li and T. N. Wong, "High speed flow visualization of a closed loop pulsating heat pipe," *International Journal of Heat and Mass Transfer,* vol. 48, no. 16, pp. 3338-3351, 7 2005.

[26] D. Fonseca, *Experimental data,* Madison,WI: Private communication, 2016.

[27] H. Ogata and S. Sato, "Forced convection heat transfer to boiling helium," in *5th International Cryogenic Engineering Conference*, 1974.

[28] L. D. Fonseca, J. Pfotenhaur and F. Miller, "Results of a three evaporator cryogenic helium Pulsating Heat Pipe," *International Journal of Heat and Mass Transfer,* vol. 120, p. 1275, 2018.

14 Appendix

14.1 Fitting equation for the properties of liquid helium

14.1.1 Dynamic viscosity fitting equation for liquid helium

The dynamic viscosity fitting equation for liquid helium is as follows:

$If\ 0K < T < 5.19K$

$$\mu = -0.0000782584 + 0.000138319T - 0.0000968005T^2 + 0.0000360241T^3 - 0.00000751592T^4 + 8.31703e^{-7}T^5 - 3.81009e^{-8}T^6 \quad (14\text{-}1)$$

$If\ 5.19K < T < 5.195K$

$$\mu = 5.491829216863793e^{-4} - 1.053629996309962e^{-4}\,T \quad (14\text{-}2)$$

$If\ 5.195K < T < 20K$

$$\mu = 0.0000116538 - 0.00000549234T + 0.00000118711T^2 - 1.28832e^{-7}T^3 + 7.62738e^{-9}T^4 - 2.34454e^{-9}T^5 + 2.92996e^{-12}T^6 \quad (14\text{-}3)$$

14.1.2 Thermal conductivity fitting equation for liquid helium

The thermal conductivity fitting equation for liquid helium is as follows:

$If\ 0K < T < 5.19K$

$$k = -0.306281 + 0.563803T - 0.414948T^2 + 0.162265T^3 - 0.0352837T^4 + 0.00403937T^5 - 0.000190419T^6 \quad \text{14-4}$$

$If\ 5.19K < T < 5.195K$

$$k = 5.19\ 5.195\ 3.500059943140276\ - 0.6712639582158 53T \quad \text{14-5}$$

$If\ 5.195K < T < 20K$

$$k = 0.0730414\ - 0.0345273T + 0.00760244T^2 - 0.000831771T^3 + 0.0000494594T^4 - 0.00000152407 + 1.90745e^{-8}\ T^6 \quad \text{14-6}$$

14.1.3 Specific heat fitting equation for liquid helium

The specific heat fitting equation for liquid helium is as follows:

$If\ 0K < T < 5.19K$

$$C_p = 32689573.4643358 - 66508629.4253465T + 56935839.3712276T^2 - 26557842.1608542T^3 + 7281835.10684757T^4 - 1172011.55244442T^5 + 102342.250014116\ T^6 - 3730.755676609 20T^7 \quad \text{14-7}$$

$If\ 5.19K < T < 5.195K$

$$C_p = 99615500.3459828 \ - 19169989.1692256T \quad \text{14-8}$$

$If\ 5.195K < T < 20K$

$$\begin{aligned} C_p = {} & 2496664.31269277 - 1523696.73701256T \\ & + 388629.259937345T^2 - 53602.0584051713T^3 \\ & + 4322.45346762991T^4 - 204.080759736468 \\ & + 5.23180754346943\ T^6 \\ & - 0.0562682358720253T^7 \end{aligned} \quad \text{14-9}$$

14.2 Fitting equation for the properties of gaseous helium

14.2.1 Dynamic viscosity fitting equation for gaseous helium

$If\ 0K < T < 5.19K$

$$\begin{aligned} \mu = {} & 0.0000349735 \ - 0.0000643549T + 0.000048861T^2 \\ & - 0.0000193895T^3 + 0.000004263T^4 \\ & - 4.92636e^{-7}T^5 + 2.34099e^{-8}T^6 \end{aligned} \quad \text{14-10}$$

$If\ 5.19K < T < 5.195K$

$$\mu = 1.269866951968433e^{-4} - 2.409327364648499e^{-5}T \quad \text{14-11}$$

$If\ 5.195K < T < 20K$

$$\mu = 0.0000116538 - 0.00000549234T + 0.00000118711T^2 - 1.28832e^{-7}T^3 + 7.62738e^{-9}T^4 - 2.34454\text{E} - 10T^5 + 2.92996e^{-12}T^6 \quad \text{14-12}$$

14.2.2 Thermal conductivity fitting equation for gaseous helium

$If\ 0K < T < 5.19K$

$$k = 0.206783 - 0.385134T + 0.295949T^2 - 0.118552T^3 + 0.0262788T^4 - 0.00305919T^5 + 0.000146361T^6 \quad \text{14-13}$$

$If\ 5.19K < T < 5.195K$

$$k = 0.792076562945634 - 0.149993101087611T \quad \text{14-14}$$

$If\ 5.195K < T < 20K$

$$k = 0.0730414 - 0.0345273T + 0.00760244T^2 - 0.000831771T^3 + 0.0000494594T^4 - 0.00000152407 + 1.90745e^{-8}\,T^6 \quad \text{14-15}$$

14.2.3 Specific heat fitting equation for gaseous helium

$If\ 0K < T < 2K$

$$C_p = 5443 \tag{14-16}$$

$If\ 2K < T < 5K$

$$C_p = 2457740 - 4826080\mathrm{T} + 3891230T^2 - 1646040T^3 + 385520T^4 - 47434.5T^5 + 2398.05T^6; \tag{14-17}$$

$If\ 5K < T < 20K$

$$C_p = 47223 \tag{14-18}$$

14.2.4 Enthalpy fitting equation for gaseous helium

$$If\ T > 1.84826 + 0.000073856P - 1.41996\mathrm{e}^{-9}P^2 + 1.74229\mathrm{e}^{-14}P^3 - 1.16493\mathrm{e}^{-19}P^4 + 3.92289\mathrm{e}^{-25}P^5 - 5.20498\mathrm{e}^{-31}P^6$$

$$\begin{aligned} h = {} & 1711 + 7850T - 0.2309P - 706.7\,T^2 + 0.07732TP \\ & - 4.825\mathrm{e}^{-7}P^2 + 81.29\,T^3 - 0.01014\,T^2P \\ & + 1.202\mathrm{e}^{-7}TP^2 - 3.837\mathrm{e}^{-13}P^3 - 4.172\,T^4 \\ & + 0.0005727T^3P - 1.022\mathrm{e}^{-8}T^2P^2 + 1.24\mathrm{e}^{-13}TP^3 \\ & - 1.995\mathrm{e}^{-18}P^4 + 0.07837\,T^5 - 1.161\mathrm{e}^{-5}T^4P \\ & + 2.721\mathrm{e}^{-10}\,T^3P^2 - 4.46\mathrm{e}^{-15}T^2P^3 + 1.567\mathrm{e}^{-21}TP^4 \\ & + 3.373\mathrm{e}^{-24}P^5 \end{aligned} \tag{14-19}$$

$$If\ T < 1.84826 + 0.000073856\mathrm{P} - 1.41996\mathrm{e}^{-9}P^2 + 1.74229\mathrm{e}^{-14}P^3 - 1.16493\mathrm{e}^{-19}P^4 + 3.92289\mathrm{e}^{-25}P^5 - 5.20498\mathrm{e}^{-31}P^6$$

$$h = -260985 + 497724T - 374435T^2 + 149788T^3 - 33320.6T^4 + 3904.86T^5 - 188.789T^6; \quad 14\text{-}20$$

14.3 Fluent Journal file code

```
;~~~ mesh and basic setup ~~
file read-case "1.cas"
define models energy yes yes yes
define models unsteady-1st-order? yes

;~~~ geo info ~~
(rp-var-define 'diameter 0.5e-3 'real #f)
(rp-var-define 'turns 5 'integer #f)
(rp-var-define 'eva_length 10e-3 'real #f)
(rp-var-define 'cond_length 10e-3 'real #f)
(rp-var-define 'adia_length 80e-3 'real #f)
(rp-var-define 'bend_radius 5e-3 'real #f)
(Define v_distance (+ (Rpgetvar 'bend_radius) 3e-3))
(Define lengthfortube (* -1 (+ (Rpgetvar 'eva_length) (Rpgetvar 'adia_length) (Rpgetvar
'cond_length))))
(Define distance (* (Rpgetvar 'bend_radius) 2))

;~~~ thermal info ~~
(rp-var-define 'T_eva 4.9 'real #f)
(rp-var-define 'T_cond 3.682 'real #f)
(rp-var-define 'heat 0.0209 'real #f)
(Define T_eva (Rpgetvar 'T_eva))
(display T_eva)
(rp-var-define 'fill 0.5 'real #f)
(rp-var-define 'guessvelocity 0 'real #f)

;~~~ solver setup ~~
(rp-var-define 'save_frequency 100 'integer #f)
(rp-var-define 'time_step 0.0025 'real #f)
(rp-var-define 'iteration_num 70 'integer #f)
(rp-var-define 'time 100 'real #f)
(rp-var-define 'timestep_num 320000 'integer #f)
```

```
(rp-var-define 'percentage 1.0 'real #f)

;~~~ derived geometry info ~~
(Define wall_thickness (* (Rpgetvar 'bend_radius) 0.7 2))
(display wall_thickness)
(Define area_eva_wall (* 2 (Rpgetvar 'turns) (Rpgetvar 'eva_length) 3.14 (+ (Rpgetvar 'diameter)
(* wall_thickness 2))))
(display area_eva_wall)
(Define area_eva_tube (* 2 (Rpgetvar 'turns) (Rpgetvar 'eva_length) 3.14  (Rpgetvar 'diameter)))
(display area_eva_tube)
(Define volume_cond (* 2 (Rpgetvar 'turns) (Rpgetvar 'cond_length) (* 0.25 (expt (Rpgetvar
'diameter) 2) 3.14)))
(display volume_cond)
(Define volume_eva (* 2 (Rpgetvar 'turns) (Rpgetvar 'eva_length) (* 0.25 (expt (Rpgetvar
'diameter) 2) 3.14)))
(display volume_eva)

;~~~ derived heat info ~~
(Define T_adia (* (+ (Rpgetvar 'T_eva) (Rpgetvar 'T_cond)) 0.5))
(display T_adia)
(Define P_adia (+ -15136 (* T_adia 22060.9) (* (expt T_adia 2) -13537.6) (* (expt T_adia 3)
3514.45)))
(display P_adia)
(Define flux_eva_wall (* (/ (Rpgetvar 'heat) area_eva_wall)))
(display flux_eva_wall)
(Define flux_eva_tube (* (/ (Rpgetvar 'heat) area_eva_tube)))
(display flux_eva_tube)

;~~~ eva frequency ~~
(Define evarate (+ -1.36137 (* 0.224081 flux_eva_tube) (* -0.00202183 flux_eva_tube
flux_eva_tube) (* 0.00000715905 flux_eva_tube flux_eva_tube flux_eva_tube)))
(rp-var-define 'evarate evarate 'real #f)

;~~~ cond frequency ~~
(Define density_vap (+ 1904.5 (* (Rpgetvar 'T_cond) -3315.1) (* (expt (Rpgetvar 'T_cond) 2)
2381.99) (* (expt (Rpgetvar 'T_cond) 3) -904.754) (* (expt (Rpgetvar 'T_cond) 4) 191.787) (*
(expt (Rpgetvar 'T_cond) 5) -21.5117) (* (expt (Rpgetvar 'T_cond) 6) 0.999131)))
(display density_vap)
(Define density_liq (+ -2819.01 (* (Rpgetvar 'T_eva) 5415.12) (* (expt (Rpgetvar 'T_eva) 2) -
4057.26) (* (expt (Rpgetvar 'T_eva) 3) 1598.36) (* (expt (Rpgetvar 'T_eva) 4) -349.525) (* (expt
(Rpgetvar 'T_eva) 5) 40.2272) (* (expt (Rpgetvar 'T_eva) 6) -1.90564)))
(display density_liq)
(Define mass_transfer_rate (* evarate (- 1 (Rpgetvar 'fill)) density_liq volume_eva (/ (- T_eva
T_adia) T_adia)))
```

```
(display mass_transfer_rate)
(Define condrate (/ (/ mass_transfer_rate 32) (* (Rpgetvar 'fill) density_vap volume_cond (/ (+
T_adia (* -1 (Rpgetvar 'T_cond))) T_adia))))
(display evarate)
(display condrate)

;~~~ gravity ~~
define operating-conditions gravity yes 0 -9.8 0

;~~~ flow condition ~~
define models viscous laminar y

;~~~ operating condition ~~
define operating-conditions operating-pressure 0
define operating-conditions operating-density? yes 1.225
define operating-conditions operating-temperature 4

;~~~ material property ~~~
define/user-defined/compiled-functions compile "libudf" yes "heliumnew.c" ""
define/user-defined/compiled-functions/load "libudf"
define materials copy fluid helium
define materials copy fluid helium-liquid
/define/materials change-create aluminum copper yes constant 9048 yes constant 0.09944 yes
constant 642.3 yes
define/user-defined/real-gas-models user-defined-real-gas-model yes "libudf"

;~~~ density ~~~
/define/materials/change-create helium-liquid helium-liquid yes piecewise-polynomial 3 0 5.19 7
-2819.01 5415.12 -4057.26 1598.36 -349.525 40.2272 -1.90564 5.19 5.195 2
4.179557149826600e+04 -8.036690442825819e+03 5.195 20 7 1105.21 -548.42 112.74 -
12.0871 0.710936 -0.0217583 0.000271052 no no no no no no no no

;~~~ viscosity ~~~
/define/materials/change-create helium-liquid helium-liquid
no no no yes piecewise-polynomial 3 0 5.19 7 -0.0000782584 0.000138319 -0.0000968005
0.0000360241 -0.00000751592 8.31703E-07 -3.81009E-08 5.19 5.195 2 5.491829216863793e-
04 -1.053629996309962e-04 5.195 20 7 0.0000116538 -0.00000549234 0.00000118711 -
1.28832E-07 7.62738E-09 -2.34454E-10 2.92996E-12
no no no no no

;~~~ conductivity ~~~
/define/materials/change-create helium-liquid helium-liquid no no yes piecewise-polynomial 3 0
5.19 7 -0.306281 0.563803 -0.414948 0.162265 -0.0352837 0.00403937 -0.000190419 5.19
```

5.195 2 3.500059943140276 -0.671263958215853 5.195 20 7 0.0730414 -0.0345273
0.00760244 -0.000831771 0.0000494594 -0.00000152407 1.90745E-08 no no no no no no

;~~~ cp ~~~
/define/materials/change-create helium-liquid helium-liquid
no
yes piecewise-polynomial 3 0 5.19 8 32689573.4643358 -66508629.4253465 56935839.3712276
-26557842.1608542 7281835.10684757 -1172011.55244442 102342.250014116 -
3730.75567660920 5.19 5.195 2 99615500.3459828 -19169989.1692256
5.195 20 8 2496664.31269277 -1523696.73701256 388629.259937345 -53602.0584051713
4322.45346762991 -204.080759736468 5.23180754346943 -0.0562682358720253 no no no no
no no no

;~~~ enthalpy ~~~
/define/materials/change-create helium-liquid helium-liquid
no no no no no yes 0 yes 4.23 no no

;~~~ vof method and define phases ~~~
define models multiphase model vof
define phases phase-domain phase-1 liquid yes helium-liquid
define phases phase-domain phase-2 gas yes real-gas-fluid

;~~~ surface tension~~~
solve/set/expert no no no no no
/define/phases/interaction-domain 1 yes 2 3 evaporation-condensation yes evarate condrate yes
polynomial 7 1.84826 0.000073856 -1.41996E-09 1.74229E-14 -1.16493E-19 3.92289E-25 -
5.20498E-31 yes yes yes no yes polynomial 7 0.000405262 -0.0000720673 0.0000486274 -
0.0000321566 0.00000806969 -9.61759E-07 4.71380E-08

;~~~ boundary conditions ~~~
/define/boundary-conditions/wall wall-evaporator mixture wall_thickness no 0 no no no
flux_eva_wall yes no no no 174 no 1
define/boundary-conditions/wall wall-condensor mixture 0 no 0 no yes temperature no (Rpgetvar
'T_cond) no no no no 174 no 1
define/boundary-conditions/wall wall-adiabatic mixture 0 no 0 no no no 0 no no no no 174 no 1

;~~~ cell zone ~~~
solve initialize set-defaults mixture temp (Rpgetvar 'T_eva)
solve initialize set-defaults gas mp (Rpgetvar 'fill)
solve initialize set-defaults mixture pressure P_adia

;~~~ initialization ~~~
/solve/initialize/initialize-flow
(Define T_eva_volume (* (+ (Rpgetvar 'T_eva) T_adia) 0.5))

```
(display T_eva_volume)
(Define T_cond_volume (* (+ (Rpgetvar 'T_cond) T_adia) 0.5))
(display T_cond_volume)
/adapt/mark-inout-hexahedron yes no (* (Rpgetvar 'bend_radius) -1) (Rpgetvar 'bend_radius)
lengthfortube 0 -10 10
/solve/patch/mixture () hexahedron-r0 () y-velocity ok (Rpgetvar 'guessvelocity)
(do ((i 1 (+ i 2))(x 1 (+ x 1))) ((> i (- (* 2 (Rpgetvar 'turns)) 2)))
(ti-menu-load-string (format #f "/adapt/mark-inout-hexahedron yes no (+ (Rpgetvar
'bend_radius) (* ~a distance)) (+ (Rpgetvar 'bend_radius) (* (+ ~a 1) distance)) lengthfortube 0 -
10 10\n" i i))
(ti-menu-load-string (format #f "/solve/patch/mixture () hexahedron-r~a () y-velocity ok
(Rpgetvar 'guessvelocity)\n" x))
(ti-menu-load-string (format #f "adapt/delete-register hexahedron-r~a\n" x))
)
(do ((i 0 (+ i 2))(x (Rpgetvar 'turns) (+ x 1))) ((> i (- (* 2 (Rpgetvar 'turns)) 2)))
(ti-menu-load-string (format #f "/adapt/mark-inout-hexahedron yes no (+ (Rpgetvar
'bend_radius) (* ~a distance)) (+ (Rpgetvar 'bend_radius) (* (+ ~a 1) distance)) lengthfortube 0 -
10 10\n" i i))
(ti-menu-load-string (format #f "/solve/patch/mixture () hexahedron-r~a () y-velocity ok (* -1
(Rpgetvar 'guessvelocity))\n" x))
(ti-menu-load-string (format #f "adapt/delete-register hexahedron-r~a\n" x))
)
/solve/patch/mixture evaporator () () temperature T_eva_volume
/solve/patch mixture adiabatic () () temperature T_adia
/solve/patch mixture condensor () () temperature T_cond_volume
adapt/mark-boundary-cells/wall-evaporator () 1
adapt/mark-boundary-cells/wall-condensor () 1
(do ((i (* 2 (Rpgetvar 'turns)) (+ i 1))) ((> i (+ (* 2 (Rpgetvar 'turns)) 0)))
(ti-menu-load-string (format #f "/solve/patch/mixture () boundary-r~a () temperature (Rpgetvar
'T_eva)\n" i))
)
(do ((i (+ 1 (* 2 (Rpgetvar 'turns))) (+ i 2))) ((> i (+ (* 2 (Rpgetvar 'turns)) 1)))
(ti-menu-load-string (format #f "/solve/patch/mixture () boundary-r~a () temperature (Rpgetvar
'T_cond)\n" i))
)

;~~~ for horizontal tubes ~~~
(do ((i (+ 2 (* 2 (Rpgetvar 'turns))) (+ i 1))) ((> i (+ (* 2 (Rpgetvar 'turns)) 2)))
(ti-menu-load-string (format #f "/adapt/mark-inout-hexahedron yes no -100 100 v_distance 100 -
10 10\n"))
(ti-menu-load-string (format #f "/solve/patch/mixture () hexahedron-r~a () x-velocity ok
(Rpgetvar 'guessvelocity)\n" i))
)
```

```
;~~~ for turns ~~~
(do ((i 0 (+ i 2))(x (+ 3 (* 2 (Rpgetvar 'turns))) (+ x 1))) ((> x (+ (* 3 (Rpgetvar 'turns)) 1)))
(ti-menu-load-string (format #f "/adapt/mark-inout-hexahedron yes no (+ (Rpgetvar
'bend_radius) (* ~a distance)) (+ (Rpgetvar 'bend_radius) (* (+ ~a 2) distance)) 0 (Rpgetvar
'bend_radius) -10 10\n" i i))
(ti-menu-load-string (format #f "/solve/patch/mixture () hexahedron-r~a () x-velocity ok (* -1
(Rpgetvar 'guessvelocity))\n" x))
(ti-menu-load-string (format #f "adapt/delete-register hexahedron-r~a\n" x))
)
(do ((i -1 (+ i 2))(x (+ 2 (* 3 (Rpgetvar 'turns))) (+ x 1))) ((> x (+ (* 4 (Rpgetvar 'turns)) 1)))
(ti-menu-load-string (format #f "/adapt/mark-inout-hexahedron yes no (+ (Rpgetvar
'bend_radius) (* ~a distance)) (+ (Rpgetvar 'bend_radius) (* (+ ~a 2) distance)) -100
lengthfortube -10 10\n" i i))
(ti-menu-load-string (format #f "/solve/patch/mixture () hexahedron-r~a () x-velocity ok (* -1
(Rpgetvar 'guessvelocity))\n" x))
(ti-menu-load-string (format #f "adapt/delete-register hexahedron-r~a\n" x))
)

;~~~ for side vertical tube ~~~
(do ((i (+ 2 (* 4 (Rpgetvar 'turns))) (+ i 1))) ((> i (+ (* 4 (Rpgetvar 'turns)) 2)))
(ti-menu-load-string (format #f "/adapt/mark-inout-hexahedron yes no  (* (Rpgetvar
'bend_radius) -1) (Rpgetvar 'bend_radius) 0 v_distance -10 10\n"))
(ti-menu-load-string (format #f "/solve/patch/mixture () hexahedron-r~a () y-velocity ok
(Rpgetvar 'guessvelocity)\n" i))
)
(do ((i (+ 3 (* 4 (Rpgetvar 'turns))) (+ i 1))) ((> i (+ (* 4 (Rpgetvar 'turns)) 3)))
(ti-menu-load-string (format #f "/adapt/mark-inout-hexahedron yes no (+ (Rpgetvar
'bend_radius) (* (- (* 2 (Rpgetvar 'turns)) 2) distance)) (+ (Rpgetvar 'bend_radius) (* (+ (- (* 2
(Rpgetvar 'turns)) 1) 1) distance)) 0 v_distance -10 10\n"))
(ti-menu-load-string (format #f "/solve/patch/mixture () hexahedron-r~a () y-velocity ok (* -1
(Rpgetvar 'guessvelocity))\n" i))
)

;~~~ solver settings ~~~
solve set discretization-scheme mom 4
solve set discretization-scheme mp 16
solve set discretization-scheme pressure 13
solve set discretization-scheme temperature 0
solve set p-v-coupling 21
solve set gradient scheme no yes
solve set under-relaxation pressure 0.3
solve set under-relaxation density 0.1
solve set under-relaxation body-force 1
solve set under-relaxation mom 0.7
```

```
solve set under-relaxation cvt 0.3
solve set under-relaxation temp 0.3
solve/monitors/residual converge-criteria 3e-5  4e-4  2e-3  7e-5  5e-10

;~~~ wall average ~~~
;~~~ temperature ~~~
/solve/monitors/surface/set-monitor temp_eva "Area-Weighted Average" mixture/temperature
wall-evaporator , no no yes "temp_eva" 1 yes flow-time
/solve/monitors/surface/set-monitor temp_cond "Area-Weighted Average" mixture/temperature
wall-condensor , no no yes "temp_cond" 1 yes flow-time
/solve/monitors/surface/set-monitor temp_adia "Area-Weighted Average" mixture/temperature
wall-adiabatic , no no yes "temp_adia" 1 yes flow-time
/solve/monitors/surface/set-monitor temp-iter-eva-avg "Area-Weighted Average" mixture
temperature wall-evaporator () no yes yes "temp-iter-eva-avg" 1 no iter
/solve/monitors/surface/set-monitor temp-iter-cond-avg "Area-Weighted Average" mixture
temperature wall-adiabatic () no yes yes "temp-iter-adia-avg" 1 no iter
/solve/monitors/surface/set-monitor temp-iter-adia-avg "Area-Weighted Average" mixture
temperature wall-condensor () no yes yes "temp-iter-cond-avg" 1 no iter

;~~~ heat flux ~~~
/solve/monitors/surface/set-monitor heatflux_in "Area-Weighted Average" mixture heat-flux
wall-evaporator () no no yes "heatflux_in" 1 yes flow-time
/solve/monitors/surface/set-monitor heatflux_out "Area-Weighted Average" mixture heat-flux
wall-condensor () no no yes "heatflux_out" 1 yes flow-time
/solve/monitors/surface/set-monitor heatflux_middle "Area-Weighted Average" mixture heat-
flux wall-adiabatic () no no yes "heatflux_middle" 1 yes flow-time
/solve/monitors/surface/set-monitor heatflux_system_php "Area-Weighted Average" mixture
heat-flux wall-adiabatic wall-condensor wall-evaporator () no no yes "heatflux_system_php" 1
yes flow-time

;~~~ heat transfer ~~~
/solve/monitors/surface/set-monitor h_coeff_eva "Area-Weighted Average" mixture heat-
transfer-coef wall-evaporator () no no yes "h_coeff_eva" 1 yes flow-time
/solve/monitors/surface/set-monitor h_coeff_cond "Area-Weighted Average" mixture heat-
transfer-coef wall-condensor () no no yes "h_coeff_cond" 1 yes flow-time
/solve/monitors/surface/set-monitor h_coeff_adia "Area-Weighted Average" mixture heat-
transfer-coef wall-adiabatic () no no yes "h_coeff_adia" 1 yes flow-time

;~~~ wall integral ~~~
;~~~ heat ~~~
/solve/monitors/surface/set-monitor heat_in "Integral" mixture heat-flux wall-evaporator () no no
yes "heat_in" 1 yes flow-time
/solve/monitors/surface/set-monitor heat_out "Integral" mixture heat-flux wall-condensor () no
no yes "heat_out" 1 yes flow-time
```

/solve/monitors/surface/set-monitor heat_middle "Integral" mixture heat-flux wall-adiabatic () no no yes "heat_middle" 1 yes flow-time
/solve/monitors/surface/set-monitor heat_system_php "Integral" mixture heat-flux wall-adiabatic wall-condensor wall-evaporator () no no yes "heat_system_php" 1 yes flow-time

;~~~ volume average ~~~
;~~~ mass and vof ~~~
/solve/monitors/volume/set-monitor vof "Volume-Average" liquid/vof adiabatic condensor evaporator , no no yes "vof" 1 yes flow-time
/solve/monitors/volume/set-monitor vof-eva "Volume-Average" liquid/vof evaporator , no no yes "vof-eva" 1 yes flow-time
/solve/monitors/volume/set-monitor vof-cond "Volume-Average" liquid/vof condensor , no no yes "vof-cond" 1 yes flow-time
/solve/monitors/volume/set-monitor vof-adia "Volume-Average" liquid/vof adiabatic , no no yes "vof-adia" 1 yes flow-time
/solve/monitors/volume/set-monitor mass-eva "Volume Integral" mixture/density evaporator , no no yes "mass-eva" 1 yes flow-time
/solve/monitors/volume/set-monitor mass-cond "Volume Integral" mixture/density condensor , no no yes "mass-cond" 1 yes flow-time
/solve/monitors/volume/set-monitor mass-adia "Volume Integral" mixture/density adiabatic , no no yes "mass-adia" 1 yes flow-time

;~~~ pressure ~~~
/solve/monitors/volume/set-monitor avg-pressure_eva "Volume-Average" mixture/pressure evaporator , no no yes "avg_press_eva" 1 yes flow-time
/solve/monitors/volume/set-monitor avg-pressure_adia "Volume-Average" mixture/pressure adiabatic , no no yes "avg_press_adia" 1 yes flow-time
/solve/monitors/volume/set-monitor avg-pressure_cond "Volume-Average" mixture/pressure condensor , no no yes "avg_press_cond" 1 yes flow-time

;~~~ volume integral ~~~
;~~~ temperature ~~~
/solve/monitors/volume/set-monitor temp-volume-eva "Volume-Average" mixture/temperature evaporator , no no yes "temp_volume-eva" 1 yes flow-time
/solve/monitors/volume/set-monitor temp-volume-cond "Volume-Average" mixture/temperature condensor , no no yes "temp_volume-cond" 1 yes flow-time
/solve/monitors/volume/set-monitor temp-volume-adia "Volume-Average" mixture/temperature adiabatic , no no yes "temp_volume-adia" 1 yes flow-time

;~~~ mass transfer ~~~
/solve/monitors/volume/set-monitor masstransfer_aida "Volume-Average" mixture/mt-rate-1 adiabatic , no no yes "mass_transfer_aida" 1 yes flow-time
/solve/monitors/volume/set-monitor masstransfer_cond "Volume-Average" mixture/mt-rate-1 condensor , no no yes "mass_transfer_cond" 1 yes flow-time

```
/solve/monitors/volume/set-monitor masstransfer_eva "Volume-Average" mixture/mt-rate-1
evaporator , no no yes "mass_transfer_eva" 1 yes flow-time

;~~~ mass and vof ~~~
/solve/monitors/volume/set-monitor mass "Volume Integral" mixture/density adiabatic
condensor evaporator , no no yes "mass" 1 yes flow-time
/solve/monitors/volume/set-monitor mass-iter "Volume Integral" mixture/density adiabatic
condensor evaporator , no yes yes "mass-iter" 1 no iter

;~~~ mass transfer ~~~
/solve/monitors/volume/set-monitor masstransfer_aida_total "Volume Integral" mixture/mt-rate-
1 adiabatic , no no yes "mass_transfer_aida_total" 1 yes flow-time
/solve/monitors/volume/set-monitor masstransfer_cond_total "Volume Integral" mixture/mt-rate-
1 condensor , no no yes "mass_transfer_cond_total" 1 yes flow-time
/solve/monitors/volume/set-monitor masstransfer_eva_total "Volume Integral" mixture/mt-rate-1
evaporator , no no yes "mass_transfer_eva_total" 1 yes flow-time
/solve/monitors/volume/set-monitor masstransfer_combined "Volume Integral" mixture/mt-rate-
1 adiabatic condensor evaporator , no no yes "mass_transfer_total_combined" 1 yes flow-time
/define/custom-field-functions define "h_fg" "mt_rate_1*(gas_enthalpy-liquid_enthalpy)"
/solve/monitors/volume/set-monitor h_fg_eva "Volume Integral" mixture/h_fg evaporator , no
no yes "h_fg_eva" 1 yes flow-time
/solve/monitors/volume/set-monitor h_fg_cond "Volume Integral" mixture/h_fg condensor , no
no yes "h_fg_cond" 1 yes flow-time
/solve/monitors/volume/set-monitor h_fg_adia "Volume Integral" mixture/h_fg adiabatic , no no
yes "h_fg_adia" 1 yes flow-time

;~~~ volume integral ~~~
/solve/monitors/volume/set-monitor system-internal-energy "Mass Integral" mixture/internal-
energy adiabatic condensor evaporator , no no yes "internal_energy" 1 yes flow-time
/solve/monitors/volume/set-monitor system-enthalpy "Mass Integral" mixture/total-enthalpy
adiabatic condensor evaporator , no no yes "total_enthalpy" 1 yes flow-time
/solve/monitors/volume/set-monitor system-total-energy "Mass Integral" mixture/total-energy
adiabatic condensor evaporator , no no yes "total_energy" 1 yes flow-time

(do((x 0 (+ x distance))(i 1 (+ i 1))) ((> i (* (Rpgetvar 'turns) 2)))
(ti-menu-load-string (format #f "surface/point-surface/cond-point-tube~a ~a (* -0.5 (Rpgetvar
'cond_length)) 0\n" i x))
(ti-menu-load-string (format #f "surface/point-surface/adia-point-tube~a ~a (- (* -1 (Rpgetvar
'cond_length))  (* 0.5 (Rpgetvar 'adia_length))) 0\n" i x))
(ti-menu-load-string (format #f "surface/point-surface/eva-point-tube~a ~a (- (* -1 (Rpgetvar
'cond_length))  (* 1 (Rpgetvar 'adia_length)) (* 0.5 (Rpgetvar 'eva_length))) 0\n" i x))
```

```
(ti-menu-load-string (format #f "/solve/monitors/surface/set-monitor
yvelocity_cond_inner_tube~a \"Area-Weighted Average\" mixture y-velocity cond-point-tube~a
() no no yes \"yvelocity_cond_inner-tube~a\" 1 yes flow-time\n" i i i))
(ti-menu-load-string (format #f "/solve/monitors/surface/set-monitor
yvelocity_adia_inner_tube~a \"Area-Weighted Average\" mixture y-velocity adia-point-tube~a ()
no no yes \"yvelocity_adia_inner-tube~a\" 1 yes flow-time\n" i i i))
(ti-menu-load-string (format #f "/solve/monitors/surface/set-monitor yvelocity_eva_inner_tube~a
\"Area-Weighted Average\" mixture y-velocity eva-point-tube~a () no no yes
\"yvelocity_eva_inner-tube~a\" 1 yes flow-time\n" i i i))
)

;~~~ solve ~~~
/file/autosave/data-frequency (Rpgetvar 'save_frequency)
/solve/set/time-step (Rpgetvar 'time_step)
/file/write-case php1.cas
/solve/dual-time-iterate (Rpgetvar 'timestep_num)  (Rpgetvar 'iteration_num)
```

14.4 UDF file code to define gaseous helium properties

```
/************************************************************************/
/* User Defined Real Gas Model :                                        */
/* For Ideal Gas Equation of State                                      */
/*                                                                      */
/************************************************************************/
#include "udf.h"
#include "stdio.h"
#include "ctype.h"
#include "stdarg.h"
#include "math.h"

#define MW 4.0026       /* molec. wt. for single gas (Kg/Kmol) */
#define RGAS (UNIVERSAL_GAS_CONSTANT/MW)

static int  (*usersMessage)(char *,...);
static void (*usersError)(char *,...);

DEFINE_ON_DEMAND(I_do_nothing)
{
  /* This is a dummy function to allow us to use */
  /* the Compiled UDFs utility                   */
}


void IDEAL_error(int err, char *f, char *msg)
{
```

```
  if (err)
    usersError("IDEAL_error (%d) from function: %s\n%s\n",err,f,msg);
}

void IDEAL_Setup(Domain *domain, cxboolean vapor_phase, char *filename,
                 int  (*messagefunc)(char *format, ...),
                 void (*errorfunc)(char *format, ...))
{
  /* Use this function for any initialization or model setups*/
  usersMessage = messagefunc;
  usersError   = errorfunc;
  usersMessage("\nLoading Real-Ideal Library: %s\n", filename);
}

#define p00  -1.702
#define p10  1.072
#define p01 0.0004251
#define p20 -0.212
#define p11 -0.0001344
#define p02 1.669e-09
#define p30 0.0158
#define p21 1.843e-05
#define p12 -4.429e-10
#define p03 9.402e-16
#define p40 -0.0003884
#define p31 -1.115e-06
#define p22 3.985e-11
#define p13 -4.921e-16
#define p04 1.168e-20
#define p41 2.412e-08
#define p32 -1.097e-12
#define p23 1.807e-17
#define p14 -7.272e-23
#define p05 -1.86e-26

double IDEAL_density(cxboolean vapor_phase,double Temp, double press, double yi[])
{
double r;
double x;
double y;
x=Temp;
y=press;
```

```
if(x>1.84826+0.000073856*y -1.41996*pow(10,-9)*pow(y,2)+1.74229*pow(10,-14)*pow(y,3)-
1.16493*pow(10,-19)*pow(y,4)+3.92289*pow(10,-25)*pow(y,5)-5.20498*pow(10,-
31)*pow(y,6))

{ r=p00 + p10*x + p01*y + p20*pow(x,2) + p11*x*y + p02*pow(y,2) + p30*pow(x,3) +
p21*pow(x,2)*y + p12*x*pow(y,2) + p03*pow(y,3) + p40*pow(x,4) + p31*pow(x,3)*y +
p22*pow(x,2)*pow(y,2) + p13*x*pow(y,3) + p04*pow(y,4)  + p41*pow(x,4)*y +
p32*pow(x,3)*pow(y,2) + p23*pow(x,2)*pow(y,3) + p14*x*pow(y,4) + p05*pow(y,5);

}
else
{
r=0.498078 + 0.000164067*y - 7.54221e-11*pow(y,2) - 1.45519e-16*pow(y,3) + 6.63976e-
21*pow(y,4);}

  return r;                   /* (Kg/m^3) */
}

double IDEAL_specific_heat(double Temp, double density, double P, double yi[])
{


double cp;
if(Temp >0 && Temp <2)
cp=5443;
if(Temp >2 && Temp <5)
cp=2457740-4826080*Temp+3891230*pow(Temp,2)-
1646040*pow(Temp,3)+385520*pow(Temp,4)-47434.5*pow(Temp,5)+2398.05*pow(Temp,6);
if(Temp>5 && Temp<10)
cp=47223;
  return cp;                   /* (J/Kg/K) */
}

#define ph00 1711
#define ph10 7850
#define ph01 -0.2309
#define ph20 -706.7
#define ph11 0.07732
#define ph02 -4.825e-07
#define ph30 81.29
#define ph21 -0.01014
#define ph12 1.202e-07
#define ph03 -3.837e-13
#define ph40 -4.172
```

```
#define ph31 0.0005727
#define ph22 -1.022e-08
#define ph13 1.24e-13
#define ph04 -1.995e-18
#define ph50 0.07837
#define ph41 -1.161e-05
#define ph32 2.721e-10
#define ph23 -4.46e-15
#define ph14 1.567e-21
#define ph05 3.373e-24

double IDEAL_enthalpy(double Temp, double density, double P, double yi[])
{
double h;
double x;
double y;
x=Temp;
y=P;
if(x>1.84826+0.000073856*y-1.41996*pow(10,-9)*pow(y,2)+1.74229*pow(10,-14)*pow(y,3)-
1.16493*pow(10,-19)*pow(y,4)+3.92289*pow(10,-25)*pow(y,5)-5.20498*pow(10,-
31)*pow(y,6))
{h=ph00 + ph10*x + ph01*y + ph20*pow(x,2) + ph11*x*y + ph02*pow(y,2) + ph30*pow(x,3)
+ ph21*pow(x,2)*y + ph12*x*pow(y,2) + ph03*pow(y,3) + ph40*pow(x,4) + ph31*pow(x,3)*y
+ ph22*pow(x,2)*pow(y,2) + ph13*x*pow(y,3) + ph04*pow(y,4) + ph50*pow(x,5) +
ph41*pow(x,4)*y + ph32*pow(x,3)*pow(y,2) + ph23*pow(x,2)*pow(y,3) + ph14*x*pow(y,4) +
ph05*pow(y,5);}
else
{h=-260985+497724*x-374435*pow(x,2)+149788*pow(x,3)-
33320.6*pow(x,4)+3904.86*pow(x,5)-188.789*pow(x,6);}
  return h;                   /* (J/Kg) */
}

double IDEAL_entropy(double Temp, double density, double P, double yi[])
{
  return 0;                   /* (J/Kg/K) */
}

double IDEAL_mw(double yi[])
{
  return MW;                    /* (Kg/Kmol) */
}

double IDEAL_speed_of_sound(double Temp, double density, double P, double yi[])
{
```

```
  return 972.0;  /* m/s */
}

#define va 0.0000349735
#define vb -0.0000643549
#define vc 0.000048861
#define vd -0.0000193895
#define ve 0.000004263
#define vf -4.92636E-07
#define vg 2.34099E-08

#define va1 1.269866951968433e-04
#define vb1 -2.409327364648499e-05

#define va2 0.0000116538
#define vb2 -0.00000549234
#define vc2 0.00000118711
#define vd2 -1.28832E-07
#define ve2 7.62738E-09
#define vf2 -2.34454E-10
#define vg2 2.92996E-12

double IDEAL_viscosity(double Temp, double density, double P, double yi[])
{

double mu;
double x;
x=Temp;
if(x>0 && x<5.19)
{mu=va + vb*Temp + vc*pow(Temp,2) + vd*pow(Temp,3) + ve*pow(Temp,4) +
vf*pow(Temp,5) +vg*pow(Temp,6);
}
else if(x>5.19 && x<5.195)
{mu=va1 +vb1;
}
else
{
mu=va2 + vb2*Temp + vc2*pow(Temp,2) + vd2*pow(Temp,3) + ve2*pow(Temp,4) +
vf2*pow(Temp,5) +vg2*pow(Temp,6);
}
  return mu;                    /* (Kg/m/s) */
}

#define ka 0.206783
```

```
#define kb -0.385134
#define kc 0.295949
#define kd -0.118552
#define ke 0.0262788
#define kf -0.00305919
#define kg 0.000146361

#define ka1 0.792076562945634
#define kb1 -0.149993101087611

#define ka2 0.0730414
#define kb2 -0.0345273
#define kc2 0.00760244
#define kd2 -0.000831771
#define ke2 0.0000494594
#define kf2 -0.00000152407
#define kg2 1.90745E-08


double IDEAL_thermal_conductivity(double Temp, double density, double P,double yi[])
{
double ktc;
double x;
x=Temp;
if(x>0 && x<5.19)
{ktc=ka + kb*Temp + kc*pow(Temp,2) + kd*pow(Temp,3) + ke*pow(Temp,4) +
kf*pow(Temp,5)+ kg*pow(Temp,6);
}
else if(x>5.19 && x<5.195)
{ktc=ka1 + kb1*Temp;
}
else
{
ktc=ka2 + kb2*Temp + kc2*pow(Temp,2) + kd2*pow(Temp,3) + ke2*pow(Temp,4) +
kf2*pow(Temp,5)+ kg2*pow(Temp,6);
}
  return ktc;                   /* W/m/K */
}

double IDEAL_rho_t(double x, double density, double y, double yi[])
{
return 0;
}
double IDEAL_rho_p(double x, double density, double y, double yi[])
```

```
{
return 0;
}


double IDEAL_enthalpy_t(double Temp, double density, double P, double yi[])
{
return 0;
}

double IDEAL_enthalpy_p(double Temp, double density, double P, double yi[])
{
return 0;
}


UDF_EXPORT RGAS_Functions RealGasFunctionList =
{
  IDEAL_Setup,                /* initialize          */
  IDEAL_density,              /* density             */
  IDEAL_enthalpy,             /* enthalpy            */
  IDEAL_entropy,              /* entropy             */
  IDEAL_specific_heat,        /* specific_heat       */
  IDEAL_mw,                   /* molecular_weight    */
  IDEAL_speed_of_sound,       /* speed_of_sound      */
  IDEAL_viscosity,            /* viscosity           */
  IDEAL_thermal_conductivity,   /* thermal_conductivity */
  IDEAL_rho_t,                /* drho/dT |const p    */
  IDEAL_rho_p,                /* drho/dp |const T    */
  IDEAL_enthalpy_t,           /* dh/dT |const p      */
  IDEAL_enthalpy_p            /* dh/dp |const T      */
};
/************************************************************************/
```

14.5 UDF for Lee model

```
#include "udf.h"
#include "mem.h"
#include "sg_mphase.h"
#define condrate RP_Get_Real("condrate")

real evarate;
```

```
DEFINE_INIT(initial_evarate, d)
{

evarate=50.0;
host_to_node_real_1(evarate);

}

DEFINE_EXECUTE_AT_END(evarate_cal)
{

  real fill_eva;
  real fill_cond;
  real mass_liquid;
  real mass_gas;
  real mass_liquid_cond;
  real mass_gas_cond;
  real T_SAT;
  real tem_pressure;

  mass_liquid=0.;
  mass_gas=0.;
  mass_liquid_cond=0.;
  mass_gas_cond=0.;

fill_eva=0;
fill_cond=0;


#if !RP_HOST

  Domain *domain;
  Domain *domain_liquid;
  Domain *domain_gas;

  domain = Get_Domain(1);

  domain_liquid = Get_Domain(2);
  domain_gas = Get_Domain(3);

  Thread *thread_eva = Lookup_Thread(domain, 14);
  Thread *thread_adia = Lookup_Thread(domain, 3);
  Thread *thread_cond = Lookup_Thread(domain, 9);
```

```
  Thread *thread_eva_liquid = Lookup_Thread(domain_liquid, 14);
  Thread *thread_adia_liquid = Lookup_Thread(domain_liquid, 3);
  Thread *thread_cond_liquid = Lookup_Thread(domain_liquid, 9);

  Thread *thread_eva_gas = Lookup_Thread(domain_gas, 14);
  Thread *thread_adia_gas = Lookup_Thread(domain_gas, 3);
  Thread *thread_cond_gas = Lookup_Thread(domain_gas, 9);

  cell_t c;

  begin_c_loop_int(c, thread_eva)
  {
   tem_pressure=C_P(c, thread_eva);
  T_SAT=1.84826+0.000073856*tem_pressure -1.41996*pow(10,-
9)*pow(tem_pressure,2)+1.74229*pow(10,-14)*pow(tem_pressure,3)-1.16493*pow(10,-
19)*pow(tem_pressure,4)+3.92289*pow(10,-25)*pow(tem_pressure,5)-5.20498*pow(10,-
31)*pow(tem_pressure,6);
if (C_T(c, thread_eva) >= T_SAT)
{mass_liquid+=C_R(c,thread_eva_liquid)*C_VOLUME(c,thread_eva)*C_VOF(c,thread_eva_li
quid)*fabs(C_T(c,thread_eva)-T_SAT)/T_SAT;}
else
{mass_gas_cond+=C_VOF(c,thread_eva_gas)*C_R(c,thread_eva_gas)*C_VOLUME(c,thread_e
va)*fabs(C_T(c,thread_eva)-T_SAT)/T_SAT;}
  }
  end_c_loop_int(c, thread_eva)


 cell_t c_2;

  begin_c_loop(c_2, thread_cond)
 {
   tem_pressure=C_P(c_2, thread_cond);
   T_SAT=1.84826+0.000073856*tem_pressure -1.41996*pow(10,-
9)*pow(tem_pressure,2)+1.74229*pow(10,-14)*pow(tem_pressure,3)-1.16493*pow(10,-
19)*pow(tem_pressure,4)+3.92289*pow(10,-25)*pow(tem_pressure,5)-5.20498*pow(10,-
31)*pow(tem_pressure,6);
if (C_T(c_2, thread_cond) >= T_SAT)
{mass_liquid+=C_R(c_2,thread_cond_liquid)*C_VOLUME(c_2,thread_cond)*C_VOF(c_2,thre
ad_cond_liquid)*fabs(C_T(c_2,thread_cond)-T_SAT)/T_SAT;}
else
{mass_gas_cond+=C_VOF(c_2,thread_cond_gas)*C_R(c_2,thread_cond_gas)*C_VOLUME(c_
2,thread_cond)*fabs(C_T(c_2,thread_cond)-T_SAT)/T_SAT;}
 }
   end_c_loop(c_2, thread_cond)
```

```
 cell_t c_3;

  begin_c_loop(c_3, thread_adia)
 {

   tem_pressure=C_P(c_3, thread_adia);
   T_SAT=1.84826+0.000073856*tem_pressure -1.41996*pow(10,-
9)*pow(tem_pressure,2)+1.74229*pow(10,-14)*pow(tem_pressure,3)-1.16493*pow(10,-
19)*pow(tem_pressure,4)+3.92289*pow(10,-25)*pow(tem_pressure,5)-5.20498*pow(10,-
31)*pow(tem_pressure,6);
if (C_T(c_3, thread_adia) >= T_SAT)
{mass_liquid+=C_R(c_3,thread_adia_liquid)*C_VOLUME(c_3,thread_adia)*C_VOF(c_3,threa
d_adia_liquid)*fabs(C_T(c_3,thread_adia)-T_SAT)/T_SAT;}
else
{mass_gas_cond+=C_VOF(c_3,thread_adia_gas)*C_R(c_3,thread_adia_gas)*C_VOLUME(c_3
,thread_adia)*fabs(C_T(c_3,thread_adia)-T_SAT)/T_SAT;}
 }

   end_c_loop(c_3, thread_adia)

#endif


# if RP_NODE
  mass_liquid = PRF_GRSUM1(mass_liquid);
  mass_gas_cond = PRF_GRSUM1(mass_gas_cond);
# endif


node_to_host_real_2(mass_liquid,mass_gas_cond);

#if !RP_NODE
evarate=(mass_gas_cond*condrate)/mass_liquid;
Message("condensation rate=: %g\n", mass_gas_cond);
Message("evaporation rate=: %g\n", mass_liquid);
Message("eva_frequency=: %g\n", evarate);
Message("cond_frequency=: %g\n", condrate);
#endif

host_to_node_real_1(evarate);
}
```

```
DEFINE_MASS_TRANSFER(evaporation, cell, thread, from_index, from_species_index,
to_index, to_species_index)
{
  real m_lg;
  real T_SAT;
  real tem_pressure;
  Thread *liq = THREAD_SUB_THREAD(thread, from_index);
  Thread *gas = THREAD_SUB_THREAD(thread, to_index);
  tem_pressure=C_P(cell, thread);
  T_SAT=1.84826+0.000073856*tem_pressure -1.41996*pow(10,-
9)*pow(tem_pressure,2)+1.74229*pow(10,-14)*pow(tem_pressure,3)-1.16493*pow(10,-
19)*pow(tem_pressure,4)+3.92289*pow(10,-25)*pow(tem_pressure,5)-5.20498*pow(10,-
31)*pow(tem_pressure,6);
  m_lg = 0.;

  if (C_T(cell, thread) >= T_SAT)
    {
     m_lg = evarate*C_VOF(cell,liq)*C_R(cell,liq)*fabs(C_T(cell,thread)-T_SAT)/T_SAT;
    }
  if ((m_lg == 0. ) && (C_T(cell, thread) <= T_SAT))
    {
     m_lg = -condrate*C_VOF(cell,gas)*C_R(cell,gas)*fabs(T_SAT-C_T(cell,thread))/T_SAT;
    }

  return (m_lg);
}

DEFINE_MASS_TRANSFER(condensation, cell, thread, from_index, from_species_index,
to_index, to_species_index)
{
  real m_gl;
  real T_SAT;
  real tem_pressure;
  Thread *gas = THREAD_SUB_THREAD(thread, from_index);
  Thread *liq = THREAD_SUB_THREAD(thread, to_index);
  tem_pressure=C_P(cell, thread);
  T_SAT=1.84826+0.000073856*tem_pressure -1.41996*pow(10,-
9)*pow(tem_pressure,2)+1.74229*pow(10,-14)*pow(tem_pressure,3)-1.16493*pow(10,-
19)*pow(tem_pressure,4)+3.92289*pow(10,-25)*pow(tem_pressure,5)-5.20498*pow(10,-
31)*pow(tem_pressure,6);
  m_gl = 0.;

  if (C_T(cell, thread) >= T_SAT)
```

```
    {
      m_gl = -evarate*C_VOF(cell,liq)*C_R(cell,liq)*fabs(C_T(cell,thread)-T_SAT)/T_SAT;
    }
  if ((m_gl == 0. ) && (C_T(cell, thread) <= T_SAT))
    {
      m_gl = condrate*C_VOF(cell,gas)*C_R(cell,gas)*fabs(T_SAT-C_T(cell,thread))/T_SAT;
    }

  return (m_gl);
}
```

14.6 High performance submit file

```
#!/bin/sh
#SBATCH --partitio=pre              # default "univ", if not specified
#SBATCH --time=0-23:59:59                       # run time in days-hh:mm:ss
#SBATCH --nodes=4                                                  # require 2 nodes
#SBATCH --ntasks-per-node=16        # (by default, "ntasks"="cpus")
#SBATCH --mem-per-cpu=4000                      # RAM per CPU core, in MB (default 4
GB/core)
#SBATCH --error=0.0005.%J.err
#SBATCH --output=0.0005.%J.out
module load ansys/2019r2
module load openmpi/4.0.5-gcc930
fluent 3ddp -g -t64 -ssh -pinfiniband -slurm -i php1.jou

done
```

14.7 Summary of steady state VOF plots with different geometries

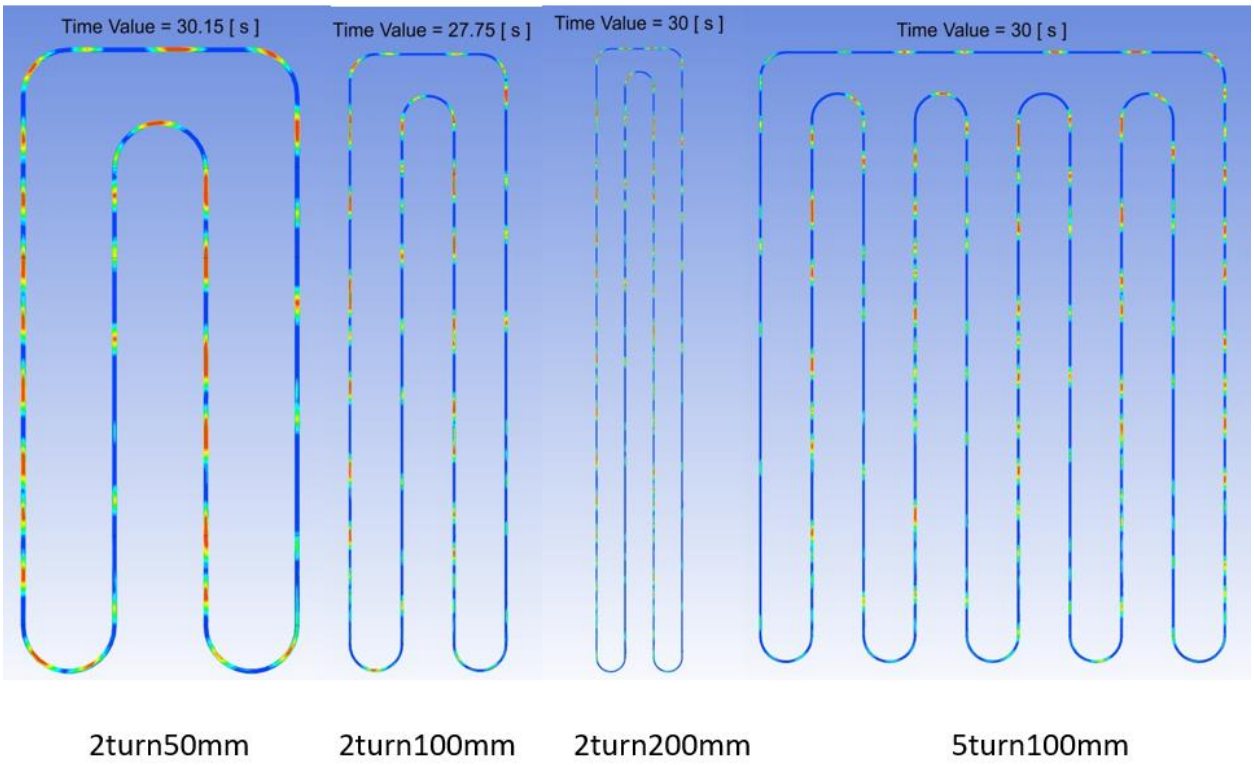

Figure 14-1: Steady state VOF visualization of different geometries with 85 w/m^2 heat flux and 80% fill ratio, part 1

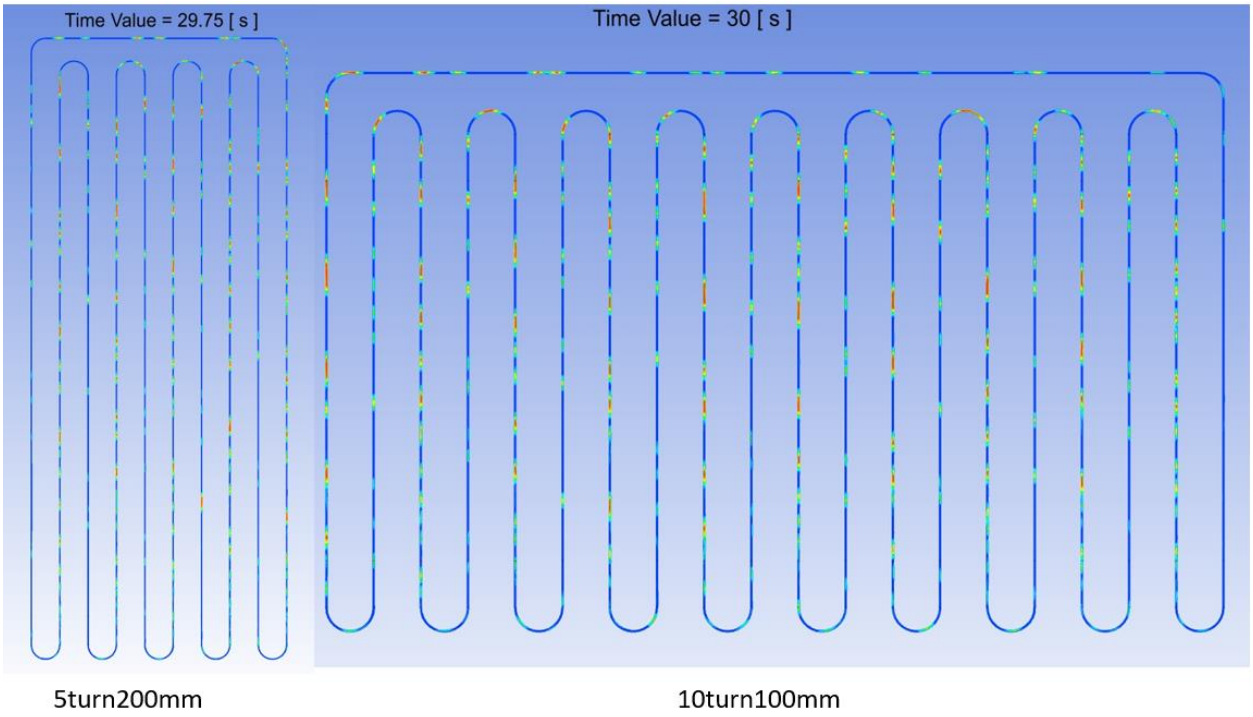

Figure 14-2: Steady state VOF visualization of different geometries with 85 w/m^2 heat flux and 80% fill ratio, part 2

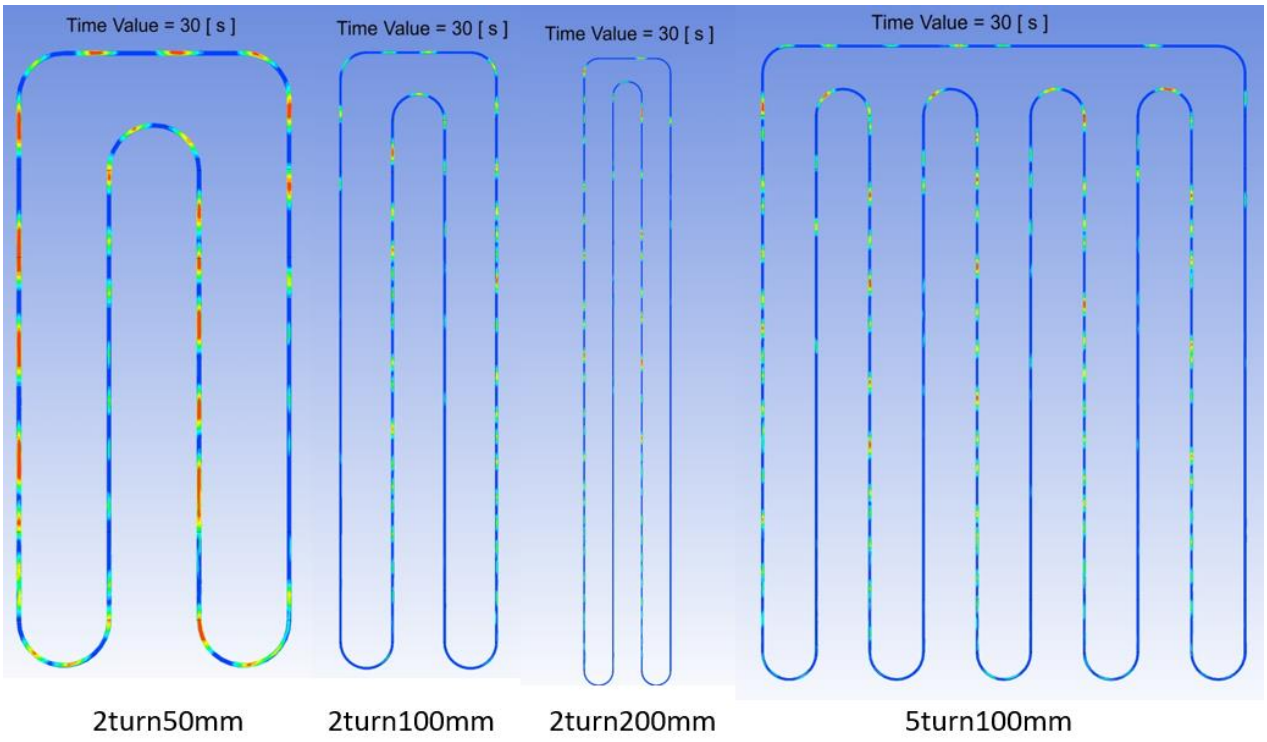

Figure 14-3: Steady state VOF visualization of different geometries with 85 w/m^2 heat flux and 90% fill ratio, part 1

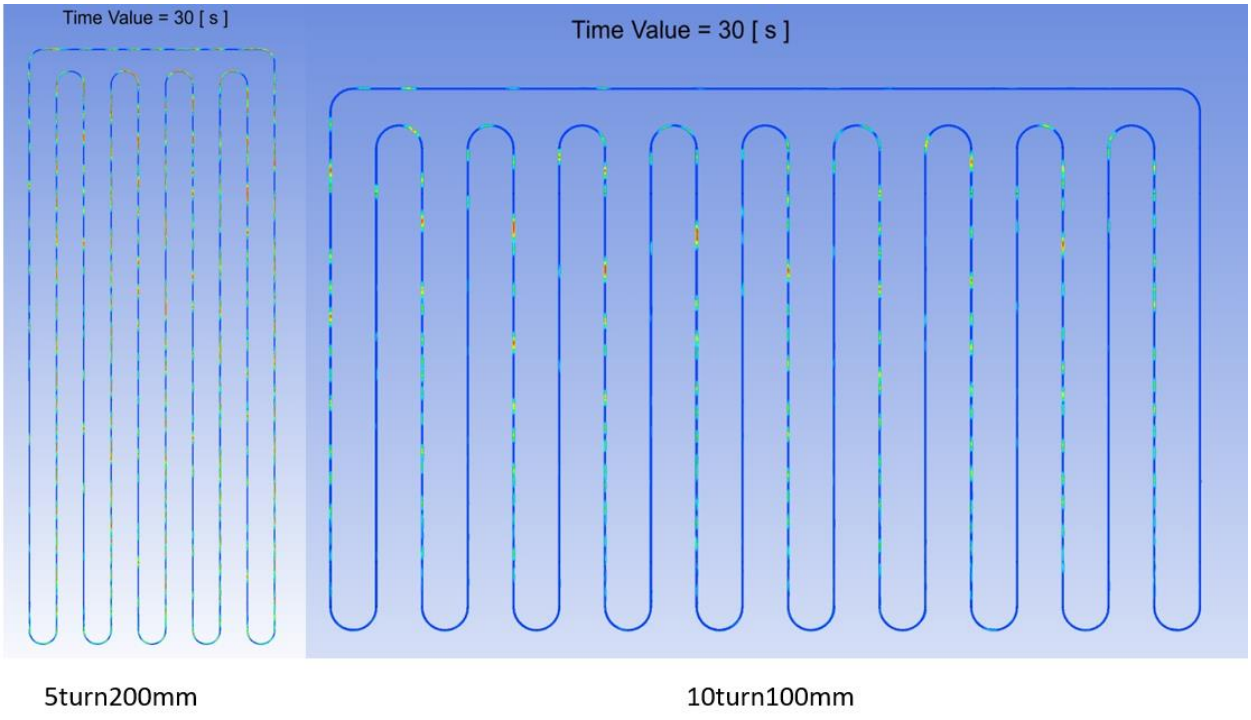

Figure 14-4: Steady state VOF visualization of different geometries with 85 w/m^2 heat flux and 90% fill ratio, part 2

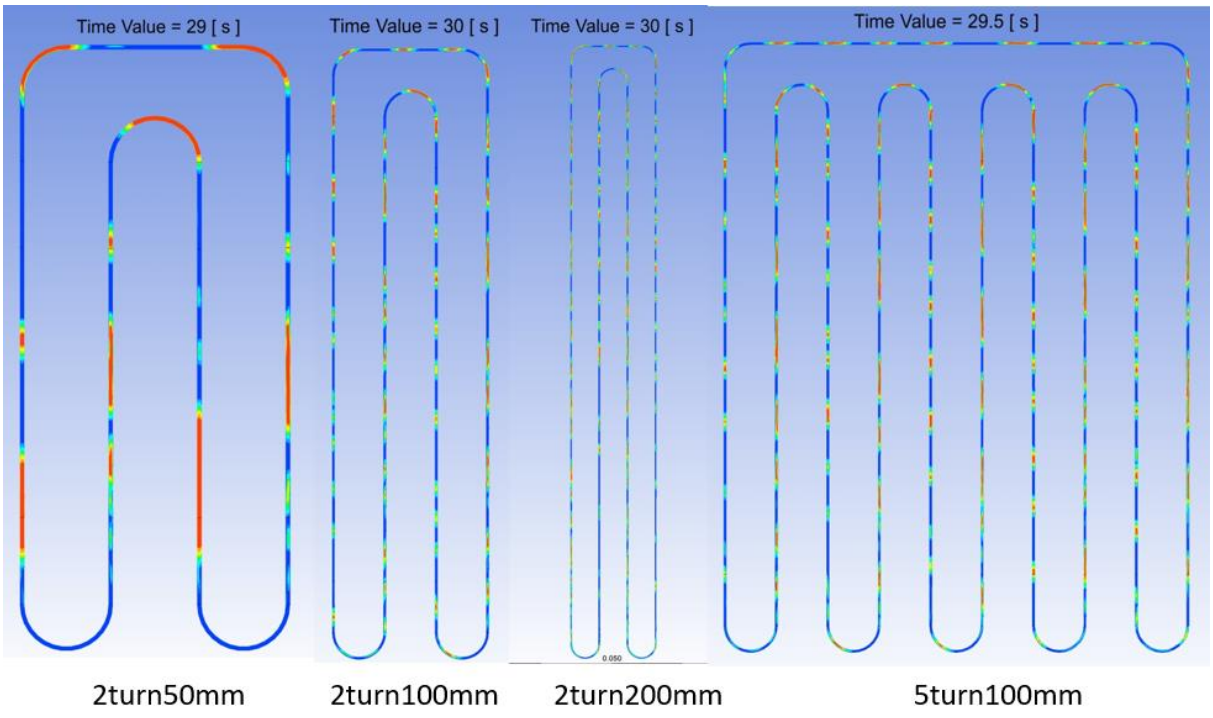

Figure 14-5: Steady state VOF visualization of different geometries with 136 w/m^2 heat flux and 70% fill ratio, part 1

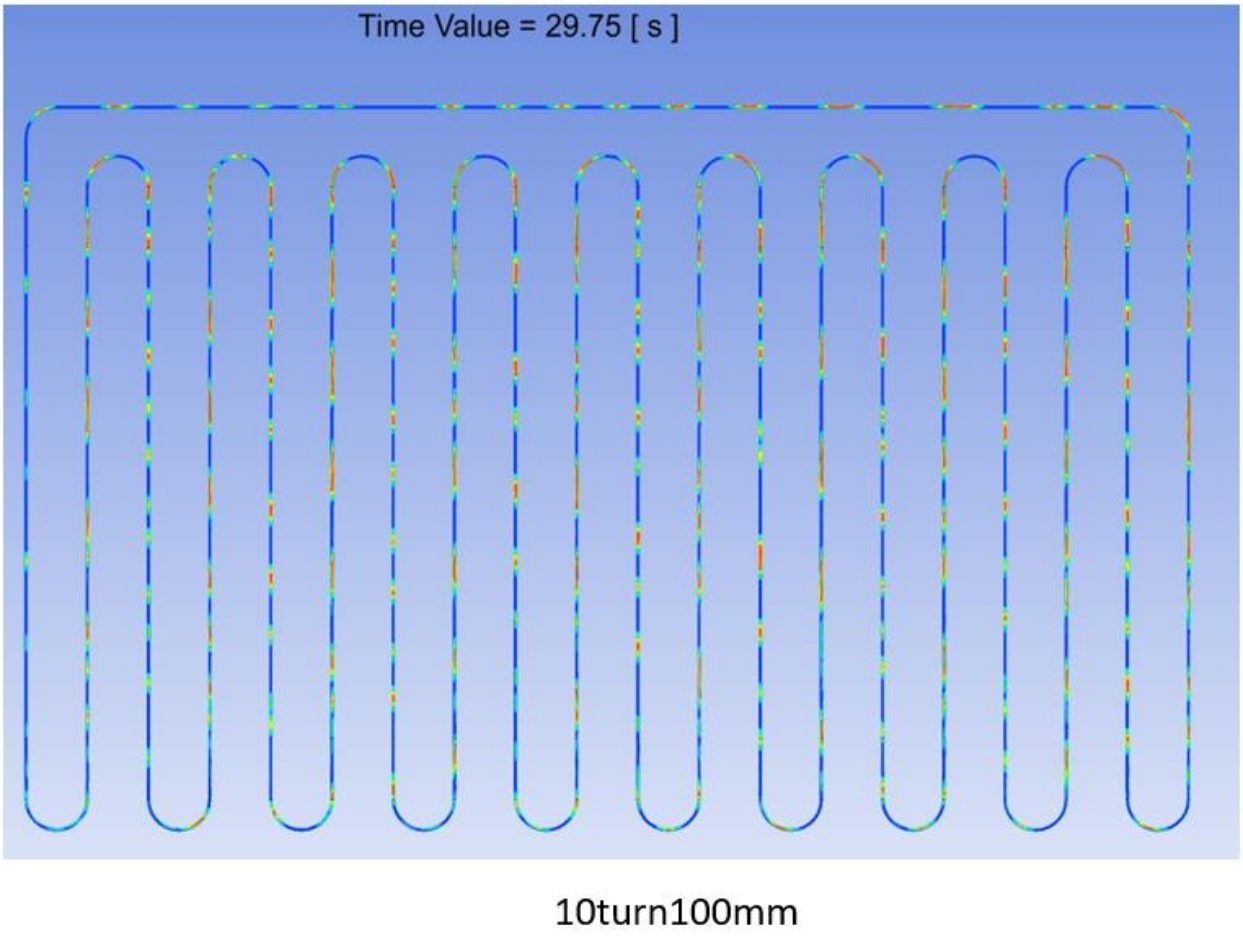

Figure 14-6: Steady state VOF visualization of different geometries with 136 w/m^2 heat flux and 70% fill ratio, part 2

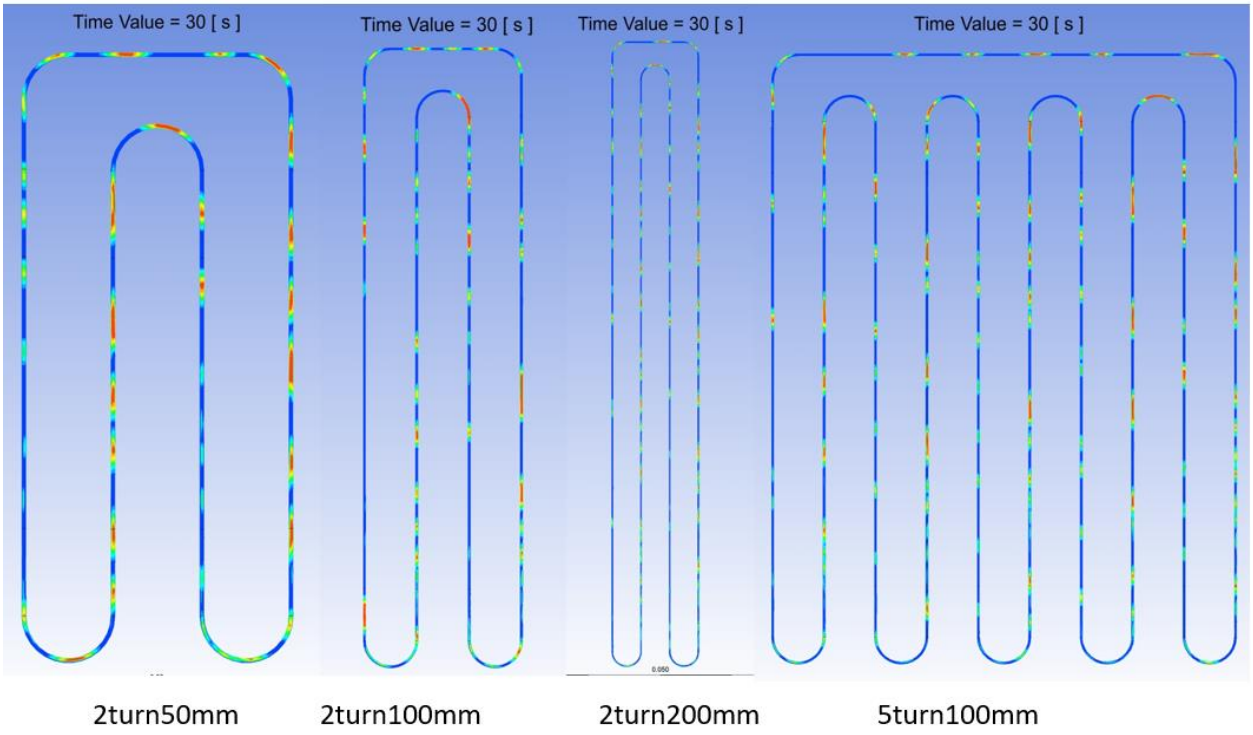

Figure 14-7: Steady state VOF visualization of different geometries with 136 w/m^2 heat flux and 80% fill ratio, part 1

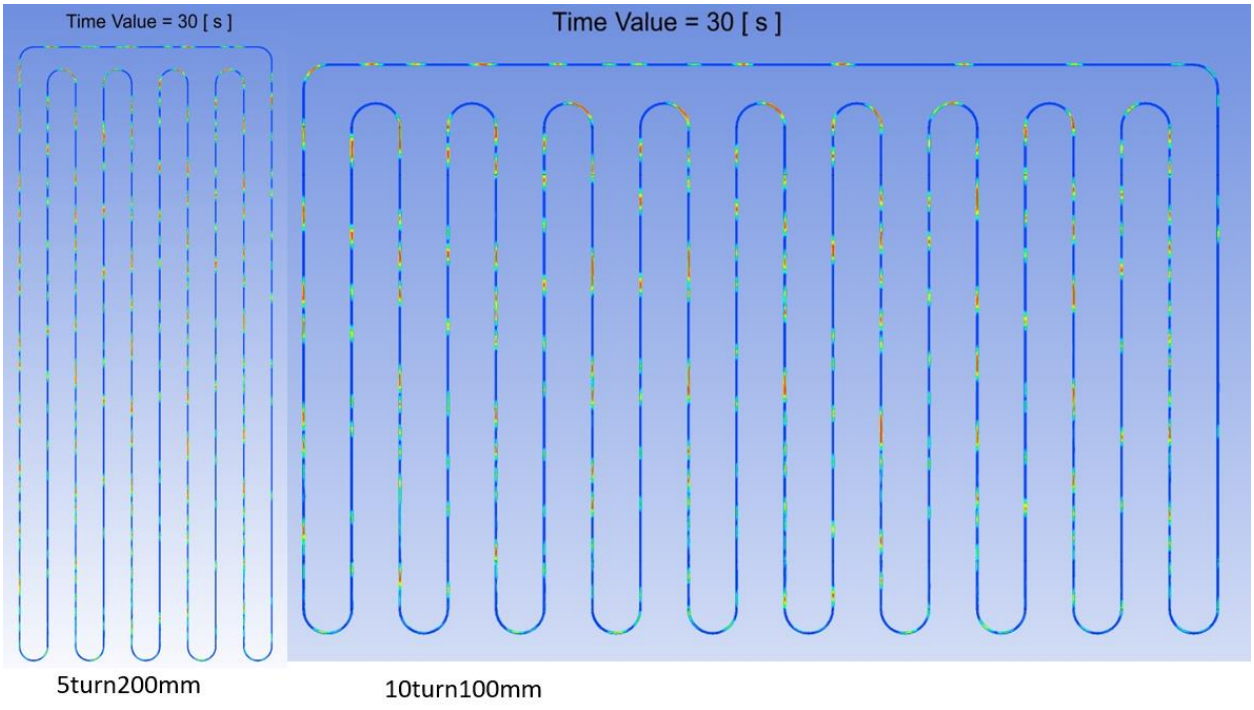

Figure 14-8: Steady state VOF visualization of different geometries with 136 w/m^2 heat flux and 80% fill ratio, part 2

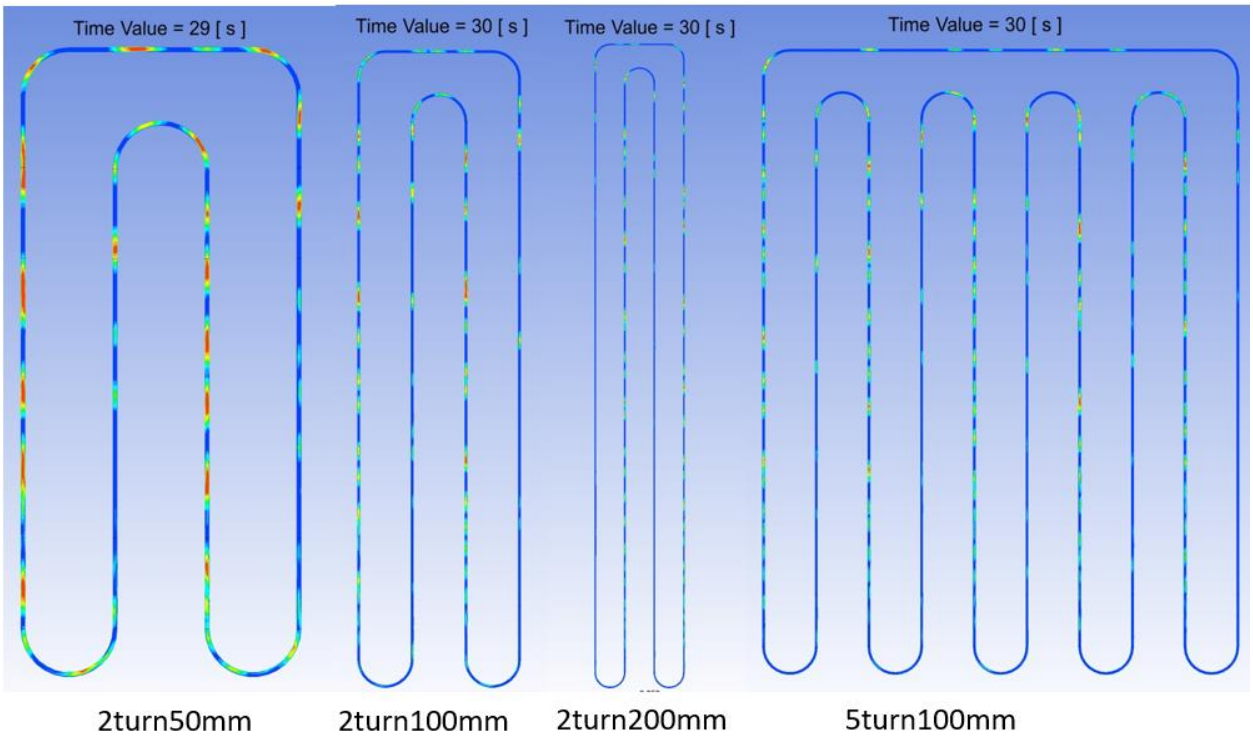

Figure 14-9: Steady state VOF visualization of different geometries with 136 w/m^2 heat flux and 90% fill ratio, part 1

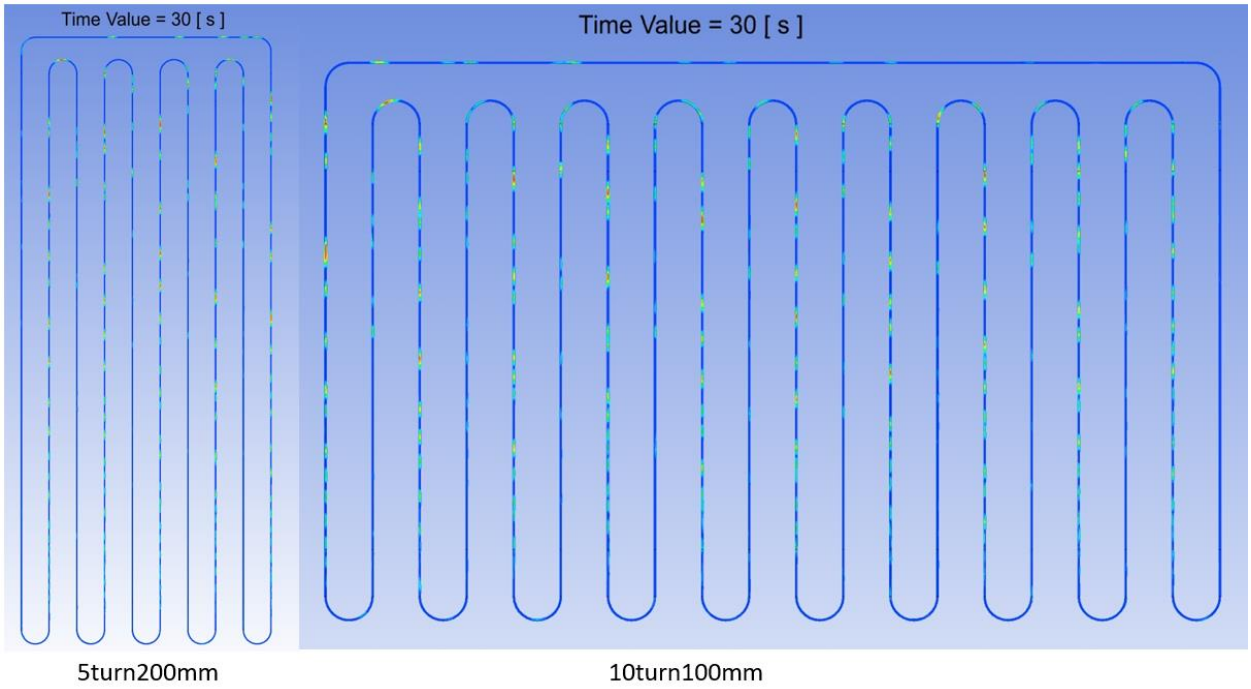

Figure 14-10: Steady state VOF visualization of different geometries with 136 w/m^2 heat flux and 90% fill ratio, part 2

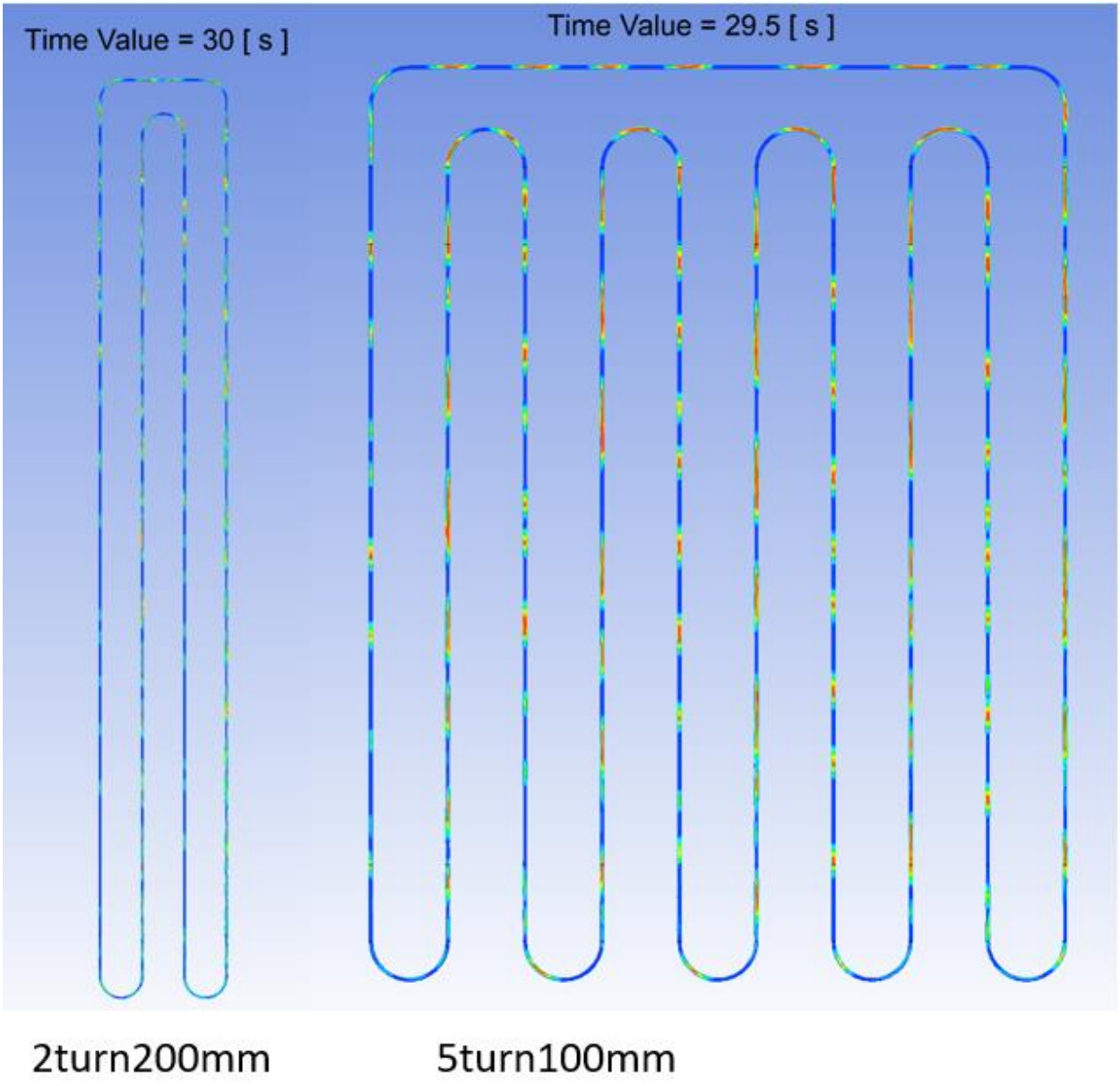

Figure 14-11: Steady state VOF visualization of different geometries with 227 w/m^2 heat flux and 70% fill ratio, part 1

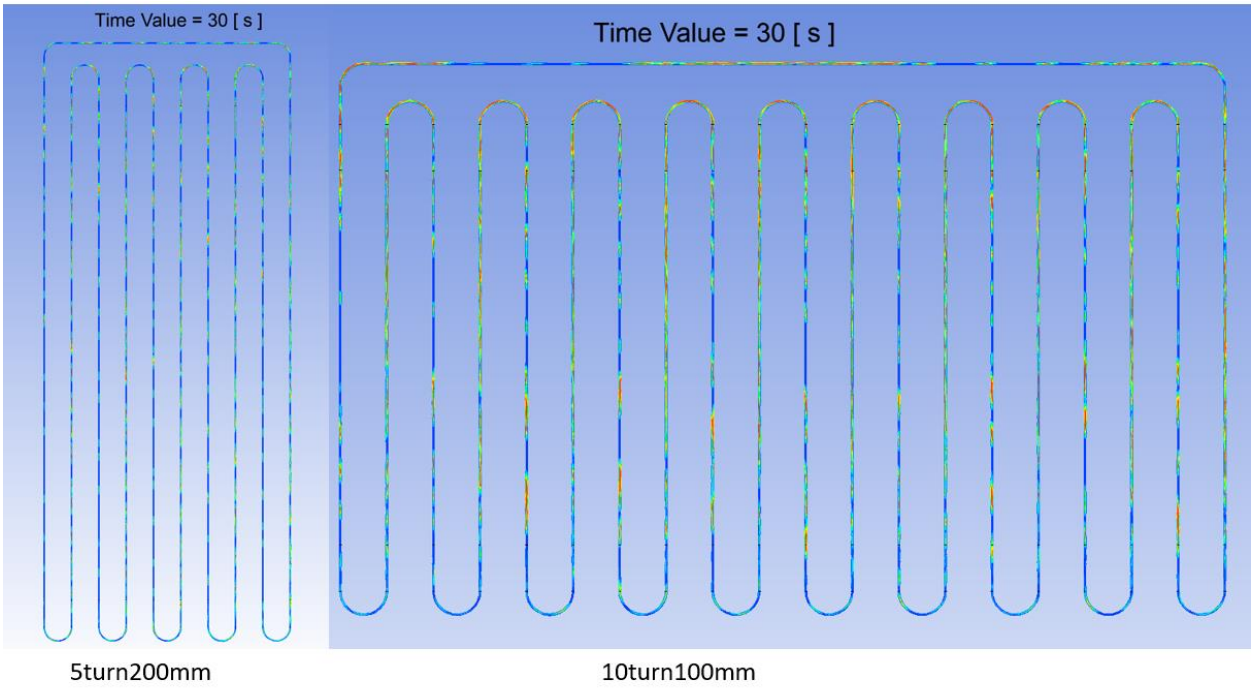

Figure 14-12: Steady state VOF visualization of different geometries with 227 w/m^2 heat flux and 70% fill ratio, part 2

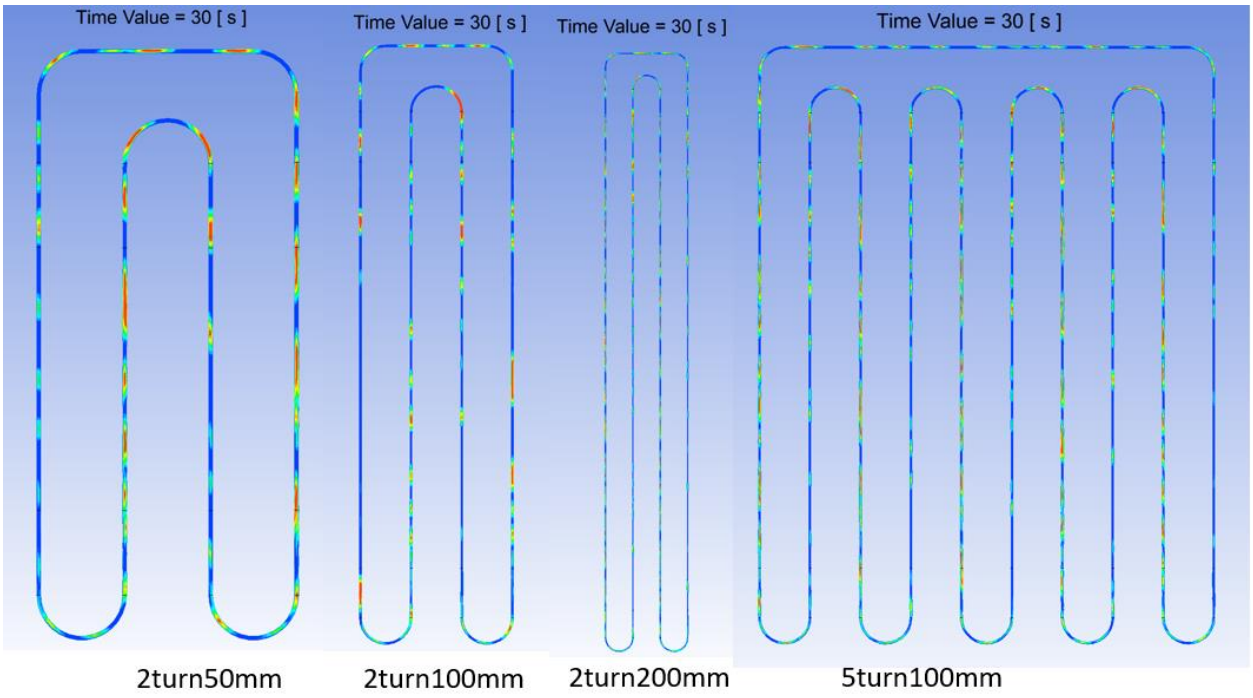

Figure 14-13: Steady state VOF visualization of different geometries with 227 w/m^2 heat flux and 80% fill ratio, part 1

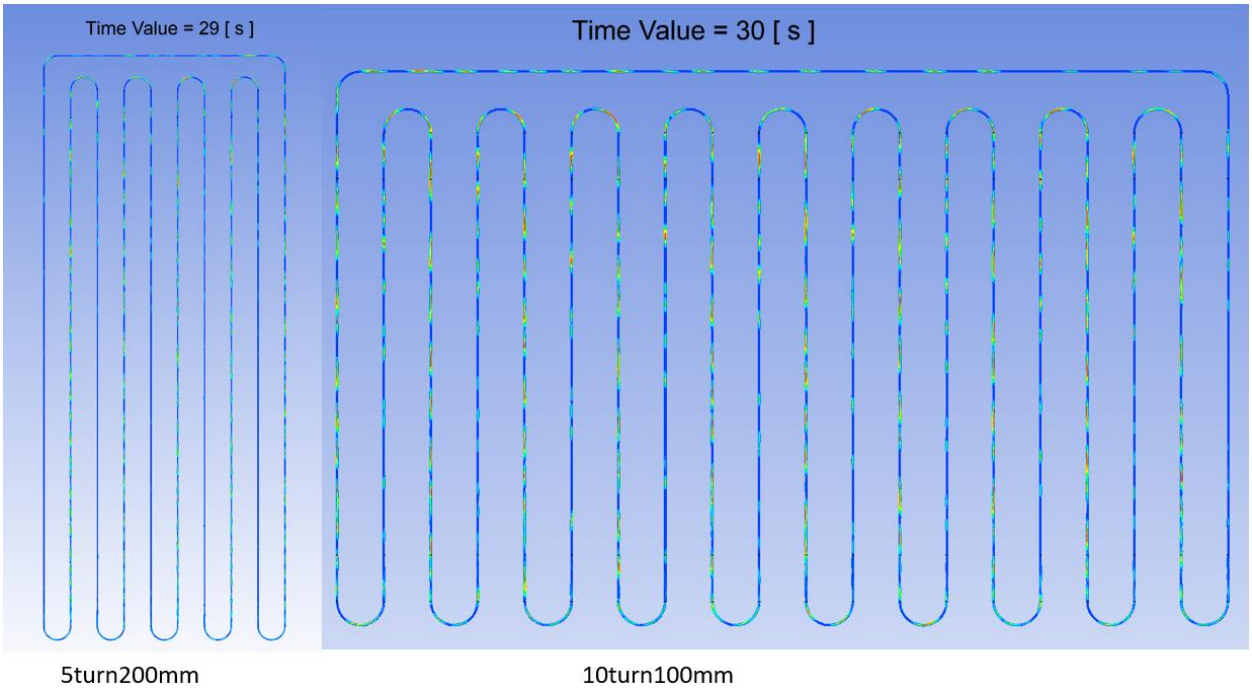

Figure 14-14: Steady state VOF visualization of different geometries with 227 w/m^2 heat flux and 80% fill ratio, part 2

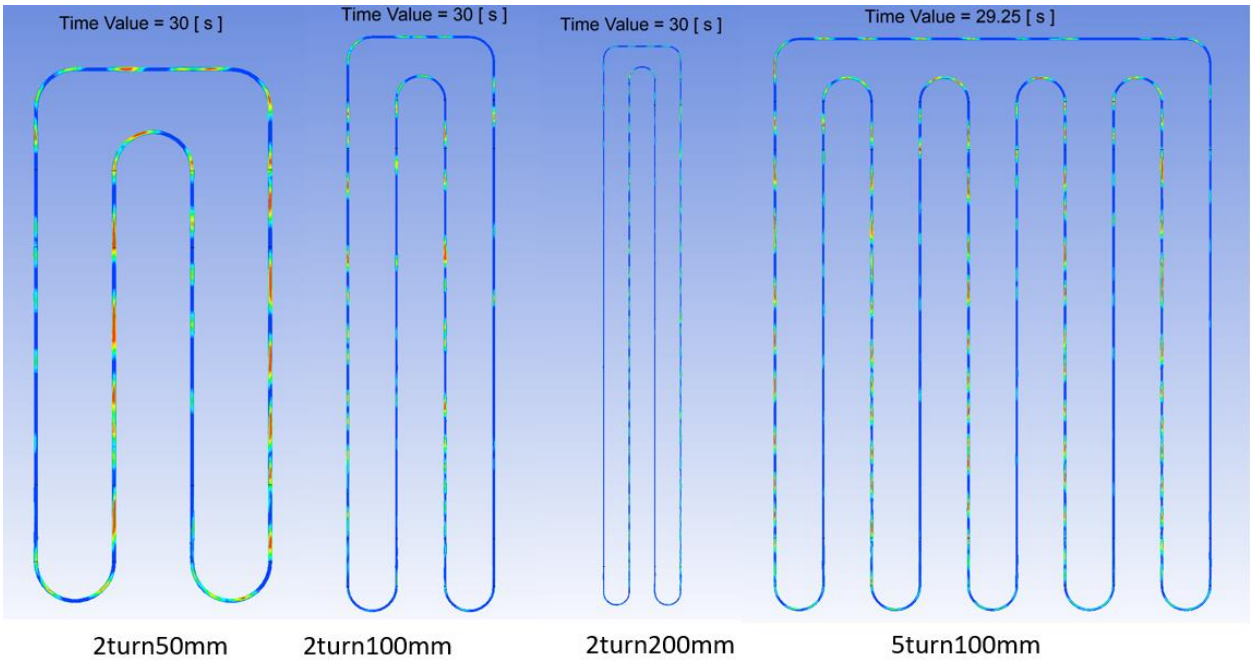

Figure 14-15: Steady state VOF visualization of different geometries with 227 w/m^2 heat flux and 90% fill ratio, part 1

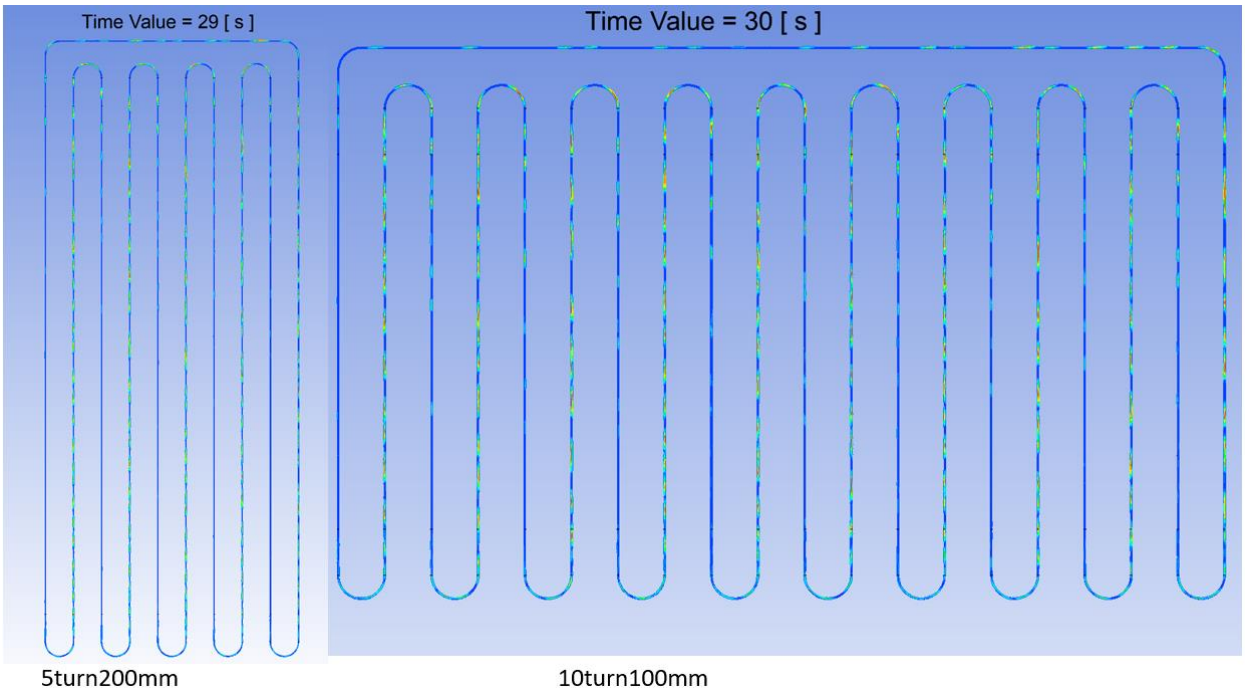

Figure 14-16: Steady state VOF visualization of different geometries with 227 w/m^2 heat flux and 90% fill ratio, part 2

14.8 Summary of velocity versus time plot

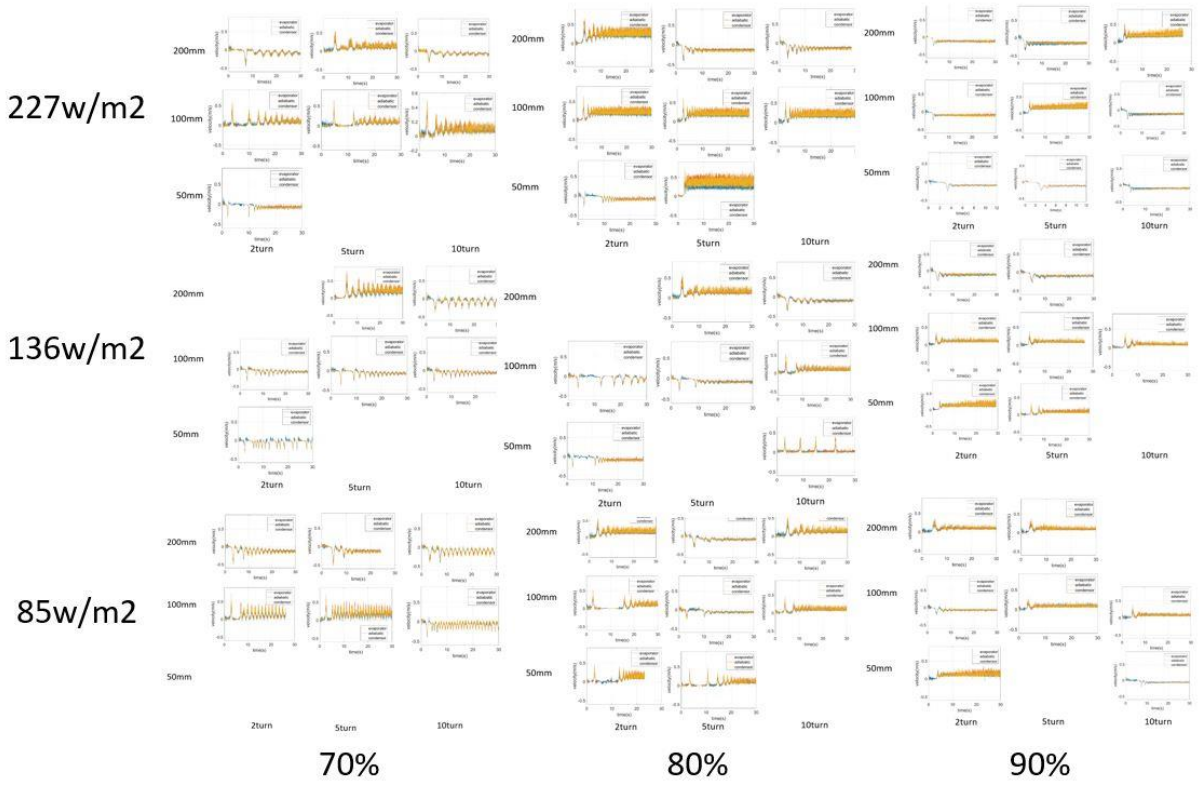

Figure 14-17

14.9 Summary of thermal conductivities

70% fill ratio and 85 w/m^2

	2turn	5turn	10turn
50mm	4.1K	NAN	NAN
100mm	4.0K	4.0K	4.0K
200mm	4.0K	4.0K	4.0K

80% fill ratio and 85 w/m^2

	2turn	5turn	10turn
50mm	3.8K	4.2K	NAN
100mm	4.0K	4.1K	4.0K
200mm	4.2K	3.9K	4.0K

90% fill ratio and 85 w/m^2

	2turn	5turn	10turn
50mm	3.7	3.8	3.9
100mm	3.8	3.8	3.8
200mm	3.8	3.9	3.8

70% fill ratio and 136 w/m^2

	2turn	5turn	10turn
50mm	4.8K	NAN	NAN
100mm	4.7K	4.7K	4.7K
200mm	4.7K	4.5K	4.7K

80% fill ratio and 136 w/m^2

	2turn	5turn	10turn
50mm	4.3K	NAN	NAN
100mm	4.4K	4.4K	4.4K
200mm	4.3K	4.4K	4.4K

90% fill ratio and 136 w/m^2

	2turn	5turn	10turn
50mm	4.2	4.3	4.3
100mm	4.3	4.3	4.3
200mm	4.3	4.3	4.3

70% fill ratio and 227 w/m^2

	2turn	5turn	10turn
50mm	NAN	NAN	NAN
100mm	4.4K	4.7K	4.9K
200mm	4.7K	4.7K	4.7K

80% fill ratio and 227 w/m^2

	2turn	5turn	10turn
50mm	4.6	NAN	NAN
100mm	4.6	4.6	4.6
200mm	4.6	4.5	4.6

90% fill ratio and 227 w/m^2

	2turn	5turn	10turn
50mm	4.5	4.5	4.5
100mm	4.5	4.5	4.5
200mm	4.5	4.5	4.5

Figure 14-18