UNIVERSITY OF CALIFORNIA,
IRVINE

Learning Hidden Boiling Dynamics using Physics-Informed Neural Networks

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Electrical and Computer Engineering

by

Sebastian Barschkis

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank Prof. Chandramowlishwaran and Prof. Won for supervising this project which was partially supported by the Office of Naval Research (ONR) under Grant No. N00014-22-1-2063.

ABSTRACT OF THE THESIS

Learning Hidden Boiling Dynamics using Physics-Informed Neural Networks

By

Sebastian Barschkis

Master of Science in Electrical and Computer Engineering

University of California, Irvine, 2023

Associate Professor Aparna Chandramowlishwaran, Chair

In heat transfer problems, it is desirable to know when critical heat flux (CHF) has been reached. Exceeding CHF results in diminished heat transfer and can pose a significant risk since at this point not all superfluous energy can be removed. Equipment and boiling rigs are in danger of being damaged if CHF is exceeded for prolonged times. To prevent such events from happening, the temperatures of coolants in boiling rigs need to be known throughout time. Ideally, they should be monitored in real-time. While coolant temperatures can be measured using devices such as thermocouples, this approach is not always feasible in practice due to the small size of experimental boiling setups. When cooling computer chips, for instance, dimensions are on the millimetre scale. In addition, thermocouples only deliver point-wise temperature estimates and particularly temperatures in areas of high interest (e.g. near the surface of a computer chip) would have to be interpolated. This work examines the feasibility of solving part of this problem with physics-informed neural networks (PINNs). The trajectories of bubbles in boiling rigs are observable. As such, one can use these bubble velocities in conjunction with established physical laws for fluid flows and train neural models that can predict velocities in entire boiling rigs. Predicted velocities can then be used to predict temperature fields. The models in this work focus on predicting liquid velocities and show how PINNs can be used to recover hidden physical properties from simulated bubble observations.

Chapter 1

Introduction

Can a neural network learn the physics found in boiling processes? If so, how much and what kind of prior knowledge would it need to have access to? And most importantly, what real-world problems could be solved with the help of such neural networks? These and many more questions on the study of heat transfer processes will be answered in the following work.

1.1 Neural networks solving PDEs

Solving partial differential equations (PDEs) with Deep Neural Networks (DNNs) has become an alternative method to established PDE solvers that find solutions numerically. Among these DNNs, the class of Physics-Informed Neural Networks (PINNs) has proven to be particularly apt at approximating solutions of systems governed by non-linear physical equations. PINNs have gained attention over the past years as they can infer PDE solutions at high precision and much faster than numerical solving schemes ever have. Similar to other neural networks, PINNs learn to approximate solutions by minimizing losses: Given a PDE, a PINN will start with an initial guess of the solution. By trying this solution in the PDE itself, the network obtains residual values that indicate how closely the guess satisfies the PDE. These residuals let the PINN know how closely its weighted neurons describe the unknown function solving the PDE. Guessing more solutions makes it possible for PINNs to determine the configuration of network weights that yields the smallest residual loss. The time and number of iterations it takes to reach a good approximation depends on the complexity and degree of the polynomial of a PDE. For highly non-linear PDEs, such as the Navier-Stokes equations for example, the training process will take a significant amount of time. This is considered a major disadvantage of PINNs compared to numerical solvers. However, a fully-trained PINN whose loss has converged to a desired precision can be used to almost instantly infer PDE solutions given prescribed boundary conditions. Where a numerical method would have to restart from scratch every time it tries to solve a PDE, a trained PINN only needs to wait for inputs to pass through its layers of neurons.

1.2 Motivation for PINNs in heat transfer studies

Both simulations and experiments of heat transfer processes have significantly advanced our understanding of bubble and boiling dynamics. However, with either approach research faces the same questions on the scalability of methods, feasibility when carrying out a study and transferability of results. Advocates of experiments will see the computational demands of simulations hindering and might question the applicability of results in real-world settings. After all, simulations are based on physical assumptions and there is no guarantee for results to match real-world behavior. In contrast, a researcher focusing on simulations might find experimental measuring tools error-prone and experimental rigs limiting in size. To some extent, there is a lack of tools that bridge simulation and experimental domains. Finding a method that combines the numerical accuracy of simulations with real-world observations would take the best of both domains and could advance the theory around boiling processes. PINNs are a promising class of DNNs that could fill this gap. Their input can be obtained from real-world experiments, their output is based on established physical laws. This combination of real-world and equation-based knowledge sets them apart from existing methods.

1.3 Problem statement

Capturing phase-change processes in boiling applications is a hard problem. Bubble growth, departure and merger events are difficult to predict and inferring how liquid and vapour phases influence each other currently requires the use of computationally expensive simulations or experimental setups. This project tests an alternative, non-intrusive approach to understand boiling behaviour better. More concretely, this work explores if hidden quantities in liquid phases, such as velocity and pressure, can be predicted using PINNs. Previous studies [22] have successfully shown that Convolutional Neural Networks (CNNs) can be used to track liquid-vapour interfaces of bubbles found in images of boiling experiments. It raises the question that if bubble interface positions are observable in space and time, would this information be sufficient to infer the behaviour of liquid surrounding bubbles? Could a PINN potentially encode the interface information and, in combination with a set of PDEs, learn to predict hidden physical quantities in liquid flows?

1.4 Governing PDEs in heat transfer problems

The Navier-Stokes (NS) equations describe the relationship between the velocity, pressure, and density of a fluid. In boiling problems where the fluid consists of two phases, liquid and vapour phase, the NS equations find application as well. In this work, only the liquid phase will be considered and the NS equations will only be solved in this one phase. In an incompressible fluid such as the liquid surrounding vapour, the NS equations can be expressed in the form

$$\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p/\rho + \mu \nabla^2 \boldsymbol{u} + \boldsymbol{g} \tag{1.1}$$

where $\boldsymbol{u} = (u, v)$ represents the velocity, ∇p the pressure gradient, μ the dynamic viscosity, ρ the fluid density, and $\boldsymbol{g} = (0, -g)$ the gravity acceleration. The NS equations can be considered the counterpart to Newton's second law as they express the principle of conservation of momentum. The law on the conservation of mass is usually used alongside Newton's second law when solving a fluid system. That is why the continuity equation

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1.2}$$

is solved together with the momentum equation (1.1). In fluids with constant flow density, the gravity term can be merged into the pressure gradient

$$\frac{\nabla p}{\rho} + \boldsymbol{g} = \frac{1}{\rho}(\nabla p + \rho \nabla G(y)) = \frac{1}{\rho}(\nabla(p + \rho G(y)),$$

where $G(y)$ represents the gravity potential. By merging ρG into pressure p, the pressure gradient captures the effects of gravity. This way the conservation of momentum can be expressed as follows

$$\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p/\rho + \mu \nabla^2 \boldsymbol{u}. \tag{1.3}$$

The Navier-Stokes (1.1) and the continuity equation(s) (1.2) are the only formulas used to describe fluid flow in this work. An even more accurate description of boiling fluid flow could include other governing PDEs, such as the non-dimensional energy conservation equation.

1.5 Semi-supervised and unsupervised training of PINNs

Depending on the availability of training data, PINNs learn in an unsupervised or semi-supervised fashion. That is, in the former approach no labelled data is included in training while the latter method would supply some labelled samples. Unsupervised training can be considered the purest form of training a PINN as convergence only depends on the loss resulting from minimizing PDE residuals. While not having to rely on labels is highly desirable, reaching convergence with only PDE residuals requires the problem domain to be well-defined. Initial and boundary conditions should sufficiently constrain the problem. Otherwise a unique PDE solution might not be discovered. In semi-supervised training, a PINN's overall loss is split into a data and a PDE portion. Depending on tuning parameters and the number of training samples, these portions define the overall loss. The main advantage compared to unsupervised training is that a fraction of knowledge about the domain, such as information about BCs, can be missing. That is, as long as a unique PDE solution exists, labelled data can assist the PDE loss to reach convergence. Especially when training data is noisy, comes from real-world sources or not all BCs of the problem domain are known, training PINNs in a semi-supervised manner is often the best and/or only viable strategy.

1.6 Objectives of this work

The goal of this project was to get a better understanding of heat transfer processes in pool-boiling setups through the use of neural networks. To this end, PINN models were implemented, trained, and their results compared against simulation data. While models were trained on simulation data only, data derived from actual boiling setups could be used too. The PINN architecture proposed in this work is dataset-independent and future studies could employ other training data sources.

Chapter 2

Related work

Studies on heat transfer problems have traditionally been driven by experiments (intrusive) and simulations (non-intrusive). While both approaches have been and still are extensively used today, a third approach based on neural networks, finds increasing application too.

2.1 Simulations

The rapid performance improvements of supercomputers fortified the development of high-fidelity boiling simulations [6, 28]. Over the past decades, they enabled researchers to study the forming and motion of bubbles in virtual settings [13, 21]. Studies utilising boiling simulations have led to accurate predictions of critical heat fluxes (CHF) as well as the motion and shape of bubbles [5]. While simulations make it possible to test a wide range of fluids and their behaviour in experimentally unfeasible domain setups, they remain computationally expensive. Boiling simulations typically require small timesteps and adaptive mesh-refinement optimizations to yield physically accurate results. As such, their application requires the use of supercomputers which limits the number of studies in this field.

2.2 Experimental studies

Real-world boiling experiments are another means to study heat transfer problems. In addition to providing validation data for simulations, they are currently the only means to reliably gain an understanding of fluid phase changes in extreme environments, such as under varying gravity. In support of the space missions to the Moon and Mars, for example, multiple efforts were been made to better understand the effects of microgravity on boiling processes (e.g. cooling processes found in “Space Shuttles”). While pool boiling experiments in reduced gravity [1] have proven to be challenging [12], more research has been directed towards the study of flow boiling in space [11, 16]. To this day, however, all state-of-the-art experiments in reduced gravity require elaborate setups, e.g. parabolic flights, drop towers, sounding rockets [8]. In that regard, studies focusing on boiling processes with terrestrial gravity conditions and subject to lab environments are less involved. The challenge in these experiments lies in building rigs that on one hand can reliably generate boiling processes and on the other hand capture bubble phenomena with minimal error. Problem-specific measuring setups equipped with combinations of thermocouples, heating elements, fluid pumps and high-speed cameras are commonly employed to produce and track pool- and flow-boiling regimes [12, 15]. However, external sensing devices such as cameras cannot accurately capture the 3D physical changes that occur during phase changes. To quantify fluid and bubble flows precisely and be able to study how, for instance, bubble merger and departure events affect velocity and temperature fields, optical measuring methods need to be incorporated into experimental setups. Particle image velocimetry (PIV), a technique that observes fluid flow through the use of tracer particles within the working fluid, is one of the optical methods that has been used in research to characterise bubble flows [25]. Other optical measuring techniques include infrared and fluorescence thermometry [24].

2.3 Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) [18] can approximate a wide range of PDEs and, besides having been employed as surrogates of CFD problems [19], found application in heat transfer problems. Based on tomographic background-oriented schlieren (Tomo-BOS) imaging measurements of fluid density and temperature, research has shown that PINNs can estimate the 3D velocity and pressure fields over an espresso cup [4]. A 2D study on micro-bubble dynamics targeted the same velocity and pressure fields and, by encoding Navier-Stokes's and Poisson's equations in a semi-informed PINN, yielded predictions for dynamics found in both single- and multi-bubble flows [29]. Combining PDEs to improve the accuracy of PINNs is also a goal of Nvidia Modulus. The framework makes it possible to build multi-physics surrogates capable of predicting solutions of conjugate heat transfer problems where energy transfers through convection and conduction between fluids and solids [17].

2.4 Convolutional Neural Networks

Neural networks that extract physical features from observations through convolutional kernels have been used to study heat transfer problems as well. For both boiling- and turbulent heat transfer phenomena, studies have shown that Convolutional Neural Networks (CNNs) can predict heat fluxes from grid-based datasets [10, 22]. Similarly to finite-difference methods, these models capture physics by evaluating data from spatially close points on a grid (pixels) in one operation. In contrast to PINNs, however, their solution accuracy can only be as good as the resolution of their inputs. Predictions with high precision must either come from very fine grids or take advantage of an interpolation scheme [26]. PINNs overcome this drawback by learning from points sampled continuously in space instead of having to rely on a discretized space.

2.5 Neural Operators

Most models based on the PINN framework [18] are problem-specific. They can recover physics in a semi-informed or uninformed fashion and make predictions about future timesteps if domain shape and boundary conditions remain constant. Instead of learning specific instances of PDE problems, recent research has looked into learning the underlying operators themselves too. To this end, the U-Net [20] and FNO [14] architectures have been used to build models of heat transfer problems. A multi-physics benchmark on bubble flows [7] employed both neural operators successfully with simulation data [6]. Before the advent of neural operators, research showed that PINNs in combination with domain decomposition methods could also be used to build models that are BC independent [27].

Chapter 3

Methods

The training process of PINNs for heat transfer problems requires the preparation of a nondimensionalized dataset. Points from this dataset can then be used to train and validate models.

3.1 Dimensionless Navier-Stokes

The objective of this work was to build physics-informed neural networks that generalize well to a broad range of boiling setups. That is, models should be usable with various sources, such as different rigs found in real-world laboratories. Models should also not depend on units found in training data, i.e. when training with velocity data in m/s, there should not be a dependence on those units. Otherwise, model training and especially inference becomes cumbersome as units from new datasets would first have to be converted to units of the primary dataset. Using dimensionless training data and formulating PDEs in models to expect non-dimensional data is a robust way to prevent such unit dependencies.

The Navier-Stokes PDEs embedded in heat transfer PINNs can be converted into dimensionless equations by defining reference values for length, velocity and time. To further illustrate the nondimensionalization process, the Navier-Stokes equations will be considered individually in each dimension. The momentum equation in x-direction

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\rho \partial x} + \nu\frac{\partial^2 u}{\partial x^2} + \nu\frac{\partial^2 v}{\partial y^2} \tag{3.1}$$

will be scaled using L as the reference length and U as the reference velocity. The reference time results from L and U and is defined by $T = L/V$. This process yields the dimensionless quantities $u^* = u/U$, $v^* = v/U$, $x^* = x/L$, $y^* = y/L$, $t^* = t/T$ which can be substituted back as follows

$$\frac{\partial u^* U}{\partial t^* L/U} + u^* U\frac{\partial u^* U}{\partial x^* L} + v^* U\frac{\partial u^* U}{\partial y^* L} = -\frac{\partial p}{\rho \partial x^* L} + \nu\frac{\partial^2 u^* U}{\partial x^{*2} L^2} + \nu\frac{\partial^2 v^* U}{\partial y^{*2} L^2} \tag{3.2}$$

Rearranging U, L, and T on both sides of the equation results in

$$\frac{\partial u^*}{\partial t^*} + u^*\frac{\partial u^*}{\partial x^*} + v^*\frac{\partial u^*}{\partial y^*} = -\frac{\partial p}{\rho U^2 \partial x^*} + \frac{\nu}{UL}(\frac{\partial^2 u^*}{\partial x^{*2}} + \frac{\partial^2 v^*}{\partial y^{*2}}) \tag{3.3}$$

Finally, expressing the ratio between viscous and convectional motion of the fluid with Reynolds number $Re = \rho UL/\mu = UL/\nu$ results in the dimensionless Navier-Stokes equation

$$\frac{\partial u^*}{\partial t^*} + u^*\frac{\partial u^*}{\partial x^*} + v^*\frac{\partial u^*}{\partial y^*} = -\frac{\partial p^*}{\partial x^*} + \frac{1}{Re}(\frac{\partial^2 u^*}{\partial x^{*2}} + \frac{\partial^2 v^*}{\partial y^{*2}}) \tag{3.4}$$

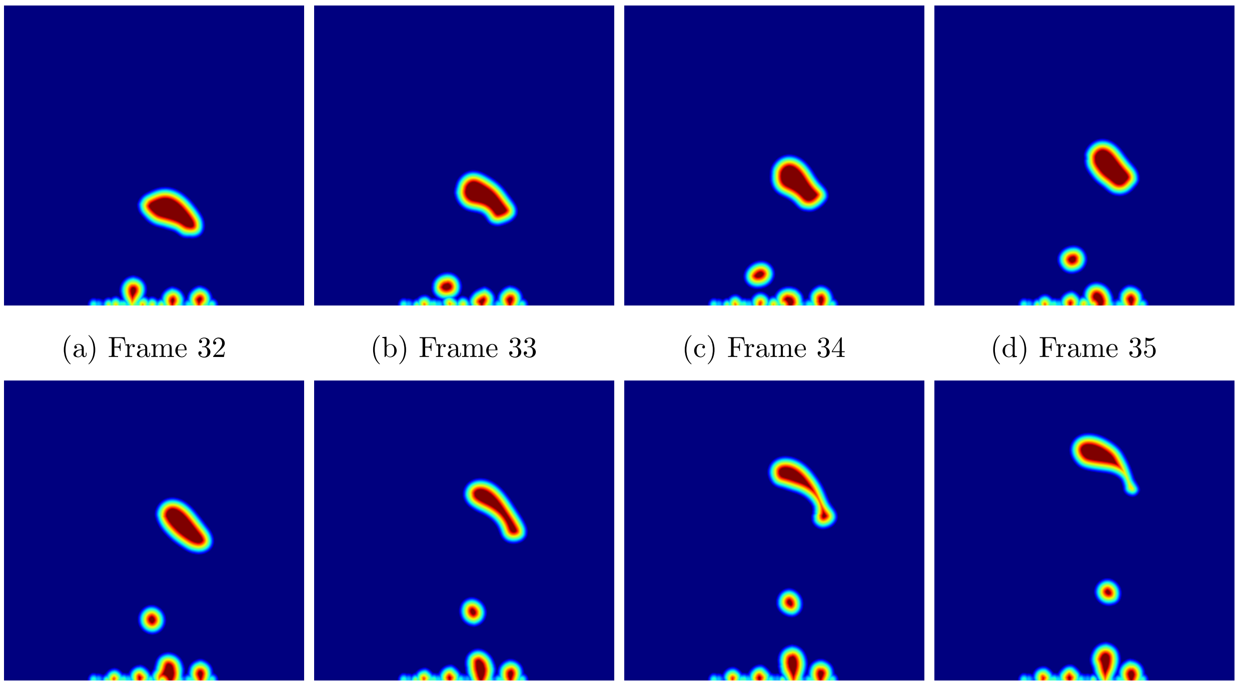
3.2 Nondimensionalization of training data

Before training models, training data needs to be brought to the appropriate scale. This includes nondimensionalization (necessary condition) and normalization (sufficient condition).

The former is required as PDEs in PINNs expect nondimensional data. Normalization is not required but highly desirable as optimizers will be able to minimize errors more efficiently.

3.2.1 Simulation data

Models in this work were trained on simulation data only. Figure 3.1 shows a selection of frames from a pool-boiling simulation carried out with Flash-X [6].



(a) Frame 32 (b) Frame 33 (c) Frame 34 (d) Frame 35

(e) Frame 36 (f) Frame 37 (g) Frame 38 (h) Frame 39

Figure 3.1: Selection of frames from pool-boiling simulation. The liquid- and bubble phases are shown using a levelset.

The main advantage of working with simulation data is that the nondimensionalization can be built into the simulation itself. This way, raw simulation data can directly be used to train PINNs.

Pool-boiling simulations for this study were carried out with FC-72 as the working fluid. Its properties are shown in table 3.1. The nondimensionalization reference values are based

Notation		Description	Value		Unit
Liquid	Vapour		Liquid	Vapour	
ρ_v	ρ_l	Density	1620	13.5	Kg/m^3
μ_v	μ_l	Dynamic Viscosity	4e-4	4e-4	Ns/m^2
C_{p_v}	C_{p_l}	Specific Heat Capacity	1110	925	$J/(KgK)$
k_v	k_l	Thermal Conductivity	5.4e-2	1.35e-2	$W/(mK)$
h_{lv}		Thermal Heat Capacity	83562		J/Kg
σ		Surface Tension	8.4e-3		N/m

Table 3.1: Physical properties of liquid and vapour phases. FC-72 was used as the working fluid in all simulations carried out with Flash-X [6].

on the properties of FC-72 and the capillary length (i.e. the factor that relates fluid surface tension and gravity)

$$L_{cap} = \sqrt{\frac{\sigma}{(\rho_{liquid} - \rho_{vapour}) \cdot g}}, \qquad V_{cap} = \sqrt{g \cdot L}, \qquad t_{cap} = \frac{L}{V} \tag{3.5}$$

Given earth gravity $g = 9.81\ {}^{m}/_{s^2}$ and FC-72's properties from table 3.1, it follows that

$$L = 7 \times 10^{-4}\ meter, \quad V = 8.68 \times 10^{-2}\ meter/second, \quad t = 8 \times 10^{-3}\ second \tag{3.6}$$

The reference values from equation 3.6 serve as the nondimensionalization factors for all position, time, and velocity values in simulation data.

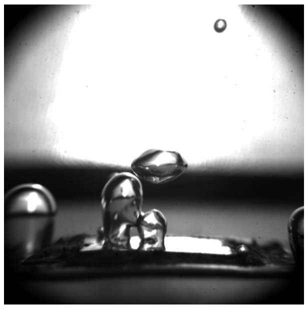
The domain size in simulations was set to range from $[-12.0, 12.0]$ in both (x, y) directions. The discretized domains stored in the dataset measured 384×384 points.

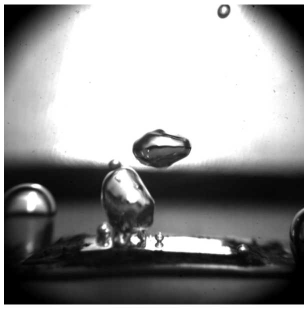
3.2.2 Experimental data

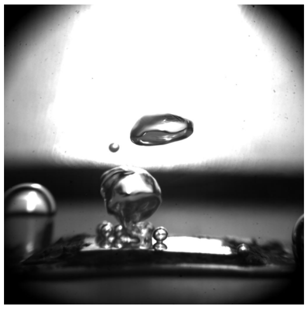
While models from this work were not trained on experimental data, the process of how one would scale real-world data appropriately will be explained by the example of a pool-boiling dataset.

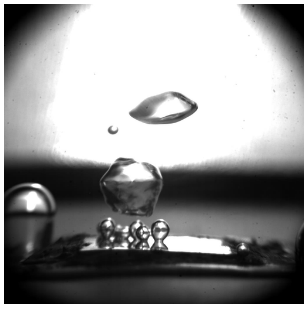
Figure 3.2 shows a selection of images that were obtained from a pool-boiling experiment carried out at 30 Watts (heating element). Besides the images themselves, pixel-wise velocity information is available for every frame (optical flow).

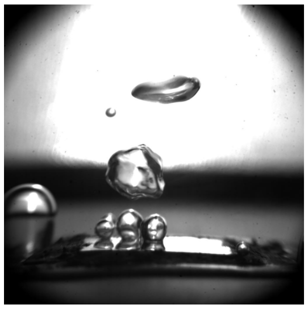
When using this kind of real-world data to train a PINN, it is important to shift all values to the nondimensional space. In the pool-boiling example, data needs to be translated from image space to nondimensional space. In addition, it would be beneficial if the new value range is on a normalized scale.

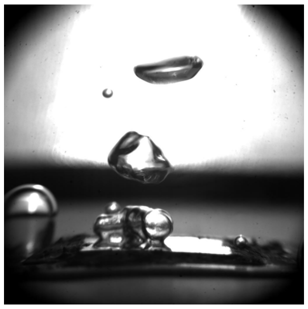

(a) Frame 02

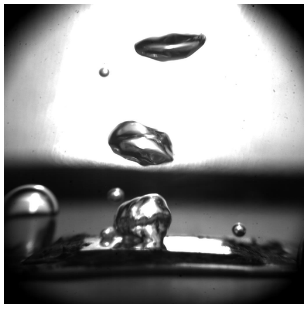

(b) Frame 04


(c) Frame 06


(d) Frame 08


(e) Frame 10


(f) Frame 12


(g) Frame 14

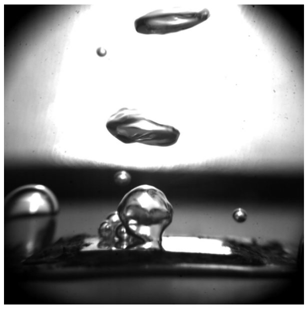
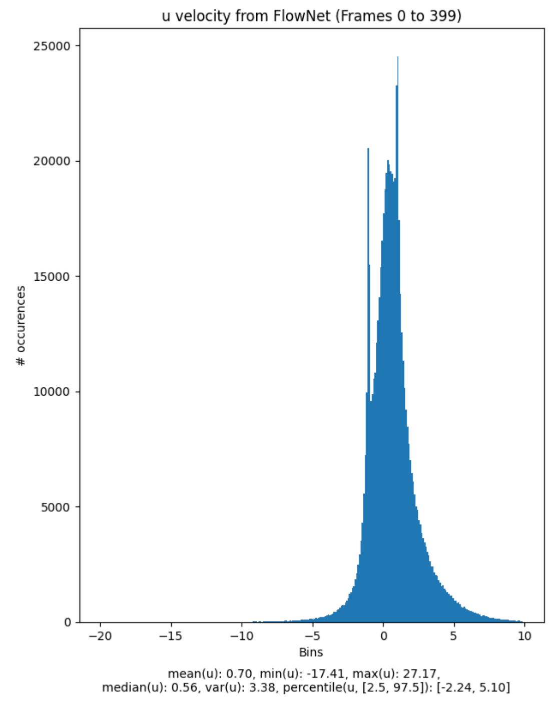
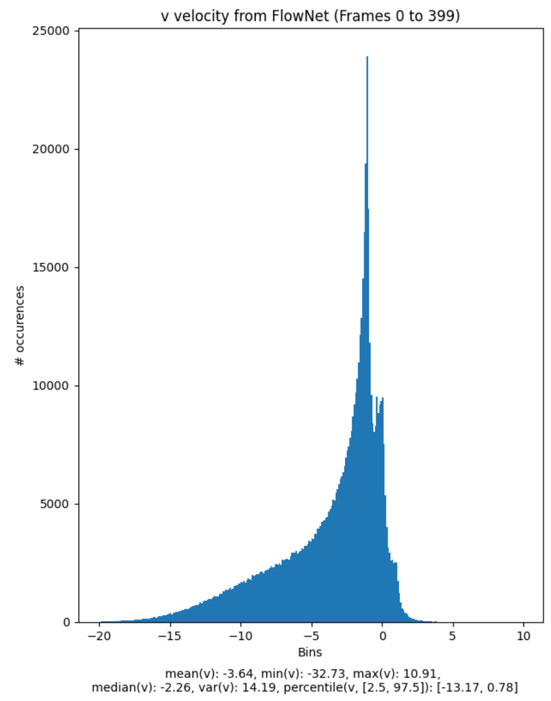

(h) Frame 16

Figure 3.2: Images from a real-world pool-boiling experiment. The flow of bubbles was captured at 400 frames/sec. The heating element was set to 30 Watts.

To this end, the first step is to find the minimum and maximum velocities among the entire dataset. As seen in the histograms in Figure 3.3, the absolute maximum velocities are 27.17 and 32.73 ${}^{pixel}/_{frame}$ for u and v velocities respectively. Based on these values, one can define an expected value range that will be used to nondimensionalize and normalize the data. For the pool-boiling dataset, $[-40, 40]$ ${}^{pixel}/_{frame}$ is a fair estimate. This leaves $\sim 20\%$ of wiggle room in case the model is retrained with more data containing slightly faster bubble flows. Having defined the range of possible velocities and already knowing the maximum



(a) Distribution of u velocity at bubble borders



(b) Distribution of v velocity at bubble borders

Figure 3.3: Histograms of velocity distribution found in pool-boiling experiment. Optical flow was extracted from 400 pool-boiling images using FlowNet [9]

values that position ($x_{max} = 512\ pixel$) and time ($t_{max} = 400\ frame$) can take, one can express maximum values in world space. Given that pool-boiling data was captured at 400 $^{frames}/_{second}$ and the real-world viewing window measures 0.02 $meter$, the maxima for velocity, position and time convert to world space as follows

$$\begin{aligned} Velocity: 40\ \frac{pixel}{frame} \quad &\cdot \frac{400\ frame}{1\ second} \cdot \frac{0.02\ meter}{512\ pixel} = 0.625\ \frac{meter}{second} \\ Position: 512\ pixel \quad &\cdot \frac{0.02\ meter}{512\ pixel} = 0.02\ meter \\ Time: 400\ frame \quad &\cdot \frac{1\ second}{400\ frame} = 1.0\ second \end{aligned} \tag{3.7}$$

The purpose of this conversion is to be able to nondimensionalize and normalize in one step.

If we convert all data from image to world space, the values from equation 3.7 can serve as the nondimensionalization reference values. That is, $V_{ref} = 0.625\ ^{meter}/_{second}$, $L_{ref} = 0.02\ meter$, and $t_{ref} = {}^{L_{ref}}/_{V_{ref}} = 0.032\ second$. Dividing the pool-boiling data in world space by the reference values yields nondimensionalized and normalized velocities and positions in ranges $[-1, 1]$ and $[0, 1]$. Since time depends on positions and velocities, not all frames will fall into a normalized range. For instance, as the dataset contains 400 frames, $t_{max} = 1.0\ second$ maps to $t_{max}/t_{ref} = 1.0\ second/0.032\ second = 31.25$. While training with such large values is not desirable, there are other measures to normalize time. For example, it would be possible to train with smaller time sequences (e.g. the first 40 frames are in the range $[0, 3.125]$)

			Space type	
		Image	World	Dimensionless
Position	Range	$[0, 511]$	$[0, 0.02]$	$[0.0, 1.0]$
	Unit	pixel	meter	n/a
Velocity	Range	$[-40, 40]$	$[-0.625, 0.625]$	$[-1.0, 1.0]$
	Unit	pixel / frame	meter / second	n/a
Time	Range	$[0, 399]$	$[0.0, 1.0]$	$[0.0, 31.25]$
	Unit	frame	second	n/a

Table 3.2: Value ranges and units from pool-boiling data. The corresponding bubble flows are shown in Figure 3.2.

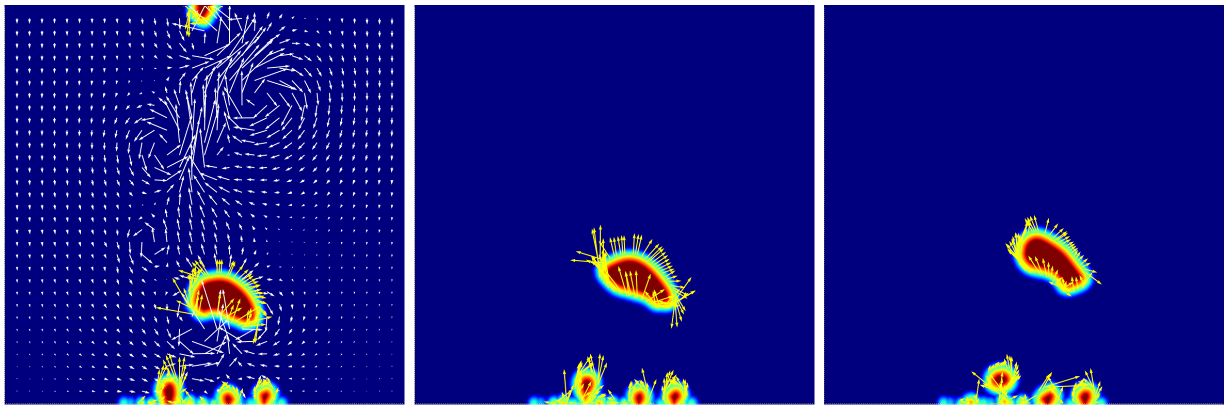
3.2.3 Choice of reference values

The reference values for experimental data (section 3.2.2) were based on the maximum values found in that specific dataset. Simulation data used the capillary length to find the nondimensionalization values. In principle, any reference values can be used to nondimensionalize datasets. As long as these values are constant and used consistently, PINNs trained on the nondimensionalized data will remain generalizable and their predictions will be dimensionless. However as described in section 3.2.2, by choosing reference values L and U in a clever way and considering the expected value range of input coordinates and velocities, the nondimensionalization step can be used for normalization too.

3.3 Ground-truth in training

The PINNs in this work were trained in a semi-informed fashion. Some knowledge was obtained from labels found in datasets, while other information was derived from physical equations.

Figure 3.4 shows a selection of simulation frames and the type of velocities that are used in training. While velocities are known throughout the entire domain during the initial condition (IC), the timesteps following the IC only make use of velocities found at the liquid-vapour interface. Boundary conditions for all four domain sides are known for all timesteps as well.


(a) Frame 31
Initial Condition

(b) Frame 32

(c) Frame 33

Figure 3.4: Velocities from simulation data used in training. At the initial condition, velocities from the entire liquid phase are known (white arrows, every 12th shown). Bubble interface and BC velocities are known at all timesteps (yellow arrows, every 2nd shown).

3.4 Design of Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) operate on batches of points. In spatio-temporal problems, these points consist of coordinates and timestamps. In a 2D setting, for example, a PDE that is a function of position and time could be approximated with a PINN that takes an (x, y, t) triplet as its input. The output is problem-dependent and can contain an arbitrary number of solution nodes, e.g. velocities and pressure (u, v, p). Similarly to network input nodes, solution nodes represent scalar values $\mathcal{NN}(x, y, t)$ for points in space and time.

In Figure 3.6 the PINN input depends on (x, y, t) (coordinates in 2D space and time). The output contains predicted solutions for (u, v, p) (velocities in 2D space and pressure). The network structure between inputs and outputs can be adjusted to the problem. While a trivial neural network would use layers of fully connected (FC) neurons, more problem-specific designs such as cone-shaped architectures that taper down towards output nodes can be employed as well.

3.5 Data points

Data points are one class of points used in PINNs. As their name suggests, these points carry data. Machine learning terminology would classify these points as labels. In PINNs predicting velocities for a heat transfer problem, for example, data points could contain velocity labels, i.e. the velocity at a point in space and time. The motivation for feeding this class of points into a PINN is to inform training about known ground truth values. Even if these true values are known for a subset of the solution space only (semi-informed), this amount of information can be sufficient to train PINNs. In some instances, PINNs can successfully approximate PDEs with no labelled data at all too.

Typically, physics-informed models learn the behaviour at data points by incorporating errors from these points into the total loss. In the simplest case, a PINN would use a mean-squared error (MSE) to minimize prediction errors at data points.

The models from this work that were trained on pool-boiling simulation data used data points found near the liquid-vapour interface, i.e. points around bubbles. While labelled data was available in other areas too, the goal of this work was to train semi-supervised PINNs. By only incorporating labels partially, models remained generalizable and the same architecture could theoretically be used with experimental data where labels are scarce and often only observable around bubbles.

As shown on the right in Figure 3.6, the loss from data points results from the error of true velocities (u_{Data}, v_{Data}) and predictions (u, v). The data loss is scaled with factor α.

3.6 Collocation points

Behaviour in areas with no labelled data can be inferred using collocation points. At these points, losses result from physical laws. Depending on how well predictions at collocation points fulfil the prescribed PDEs, models can make physical assumptions in areas with no labelled data. In pool-boiling datasets, the area between domain boundaries and liquid-vapour interface points represents the sampling space for collocation points. The objective for every point sampled in this area is to minimize the residuals resulting from previously defined PDEs.

The pool-boiling PINNs from this work follow this principle and sample collocation points in the liquid phase only. Based on the assumption that the physics in this phase are governed by the Navier-Stokes (NS) equations, models compute residuals for the continuity 1.2 and momentum equations 1.1. The sum of residuals from all collocation points is added to the total loss. This way, and especially in the absence of labelled points, it is possible to make predictions in areas where only the physical laws are known.

The PDE loss is part of the total loss and weighted with factor β in the overview Figure 3.6.

3.7 Boundary points

To constrain the solution space of PDEs, appropriate boundary conditions (BCs) should be part of model training. In pool-boiling datasets, the BCs constraining the PDE solution space can be found in two locations: (1) near the liquid-vapour interface and (2) along the border of the viewing window. As the former BCs evolve both spatially and temporally, they are considered their own class (data points) and are not considered to be part of the boundary condition (BC) (see section 3.5). For this work, the set of points describing BCs consists only of those points that are constant in space and time. That is, all BC points found along the boundaries of the viewing window, i.e. the left, top, right, and bottom sides.

3.7.1 Soft boundaries

Depending on how BCs are fed into models, one can distinguish between soft and exact BC imposition approaches. In both strategies, models see the same BCs during training. However, depending on the complexity of the problem, one approach can fare better than the other and yield the optimal solution quicker.

The soft BC imposition method follows the same principle used to learn data (3.5) and collocation points (3.6). That is, by sampling additional points near domain boundaries and minimizing the loss at these locations, models learn to reproduce the BC behaviour. The term “soft” BC refers to the idea that models learn BCs through a weighted loss contribution.

BCs are not enforced and how well a model learns BCs depends on the weight factor for the BC loss and the total number of boundary points. The constraint is considered to be a "soft" constraint as it depends on external factors.

Figure 3.6 shows the model with soft boundary loss. Factor γ scales the loss contribution of points found along domain boundaries.

3.7.2 Exact boundaries

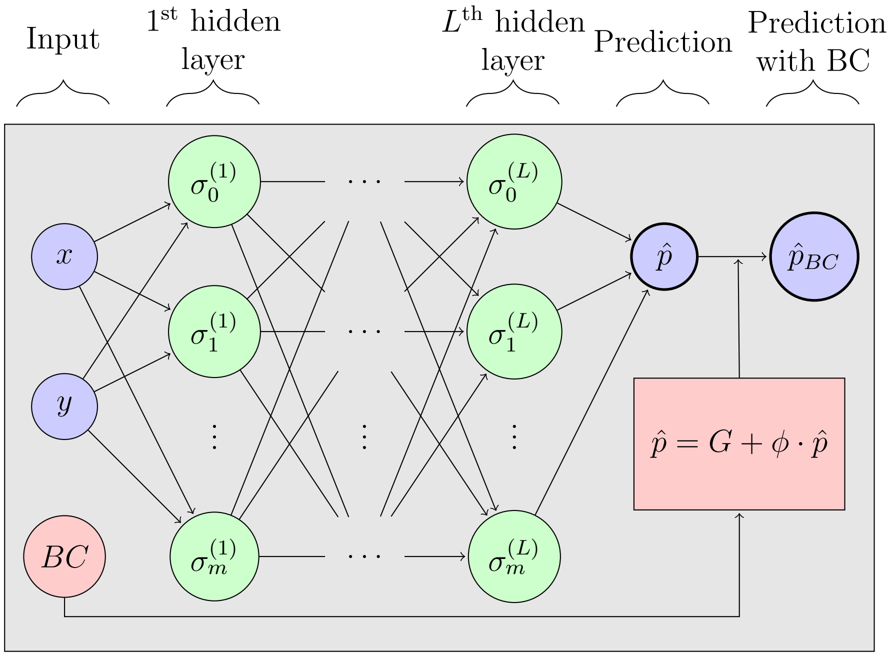


Figure 3.5: Exact BC enforcement in a fully connected architecture. The BC is "injected" into the result tensor.

Previous studies on BCs in PINNS [27] have shown that soft BCs cannot always capture physical motions near domain boundaries accurately. Predictions for flows near solid walls, for example, can appear highly unphysical when streams go into instead of parallel to walls. One approach counteracting this problem is based on an exact BC imposition approach based on distance functions [23]. The exact BC principle finds application in a wide range of PDEs and can also be used in the PINNs predicting bubble motions.

Figure 3.5 gives an overview of the exact BC imposition approach when applied to a FC architecture. In contrast to soft BCs, no additional boundary points have to be sampled in domain space as a boundary loss contribution is unnecessary. BCs are encoded into models instead of being learned with weights. Predictions near BCs will be guaranteed to be correct.

Predictions $\hat{p}$ obtained from an exact BC model know about BCs as all $\hat{p}$'s are compositions of functions G and ϕ which both implicitly encode the BC. Function $G(\boldsymbol{x})$ is an interpolation function that returns interpolated BC values for every position $\boldsymbol{x}$.

$$\begin{aligned} G(\boldsymbol{x}) &= \sum_{i}^{N_{bc}} \frac{w_i z_i}{\sum_i^{N_{bc}} w_i} \\ w_i &= |\boldsymbol{x} - \boldsymbol{x_i}| \end{aligned} \tag{3.8}$$

That is, every collocation point computes the distance w_i to all i BC points. Paired with BC values z_i found at BC locations x_i, G returns a scalar for every position $\boldsymbol{x}$. This scalar is the interpolation of the BC at position $\boldsymbol{x}$.

The second function ϕ in the exact BC method serves the purpose of filtering predictions resulting from feed-forward passes through the network. The idea is to reduce network contributions in those areas where interpolated values $G(\boldsymbol{x})$ already yield predictions with small errors. $\phi(\boldsymbol{x})$ needs to be a smooth function that returns small values in areas near domain BCs and higher values everywhere else. In square 2D domains of size $[0, 1] \times [0, 1]$ where every domain side has a BC, ϕ can be chosen as follows

$$\phi(\boldsymbol{x}) = \phi(x, y) = x \cdot (1 - x) \cdot y \cdot (1 - y) \tag{3.9}$$

The maximum of ϕ is in the centre of the domain, the minima can be found along domain boundaries.

3.8 Initial condition points

When training with temporal data from a dynamical system, the initial state needs to be captured and passed to models during training as well. Otherwise, every pool-boiling model, for example, would assume initial velocities to be zero. To prevent this from happening, the initial condition has to be captured at t_0 and learned alongside all other points. A common strategy is to extract labelled points at t_0 and throughout the entire domain space. These points can be learnt similarly to data points using an error metric of choice (e.g. MSE).

3.9 Loss function

The total loss of the models consists of the sum of losses from each point category.

$$\mathcal{L}_{Total} = \mathcal{L}_{Data} + \mathcal{L}_{PDE} + \mathcal{L}_{BC} + \mathcal{L}_{IC} \tag{3.10}$$

As balancing contributions from individual losses makes training more efficient, it is common to include tuning factors (hyperparameters).

$$\mathcal{L}_{Total} = \alpha \cdot \mathcal{L}_{Data} + \beta \cdot \mathcal{L}_{PDE} + \gamma \cdot \mathcal{L}_{BC} + \delta \cdot \mathcal{L}_{IC} \tag{3.11}$$

Expanding the loss terms $\mathcal{L}_X$ shows that each loss term is the sum of errors from each point class. The hyperparameters expand as well, e.g. α consists of α_1 and α_2 for velocities (u, v), respectively. R_0, R_1, R_2 correspond to the residuals from the continuity and momentum equations.

$$
\begin{aligned}
\mathcal{L}(\theta) &= \frac{1}{N_D}\sum^{N_D}\left(\alpha_1(\hat{u}-u_{Data})^2+\alpha_2(\hat{v}-v_{Data})^2\right)\\
&+\frac{1}{N_C}\sum^{N_C}\left(\beta_1 R_0^2+\beta_2 R_1^2+\beta_3 R_2^2\right)\\
&+\frac{1}{N_{BC}}\sum^{N_{BC}}\left(\gamma_1(\hat{u}-u_{BC})^2+\gamma_2(\hat{v}-v_{BC})^2\right)\\
&+\frac{1}{N_{IC}}\sum^{N_{IC}}\left(\delta_1(\hat{u}-u_{IC})^2+\delta_2(\hat{v}-v_{IC})^2\right)
\end{aligned}
\tag{3.12}
$$

Note that when using an exact boundary condition (section 3.7.2), the error resulting from $\mathcal{L}_{BC}$ will be small since the correct solution values along domain boundaries are encoded into the model. Hence, it is possible to neglect $\mathcal{L}_{BC}$ during the loss calculation (i.e. $\gamma_i = 0$).

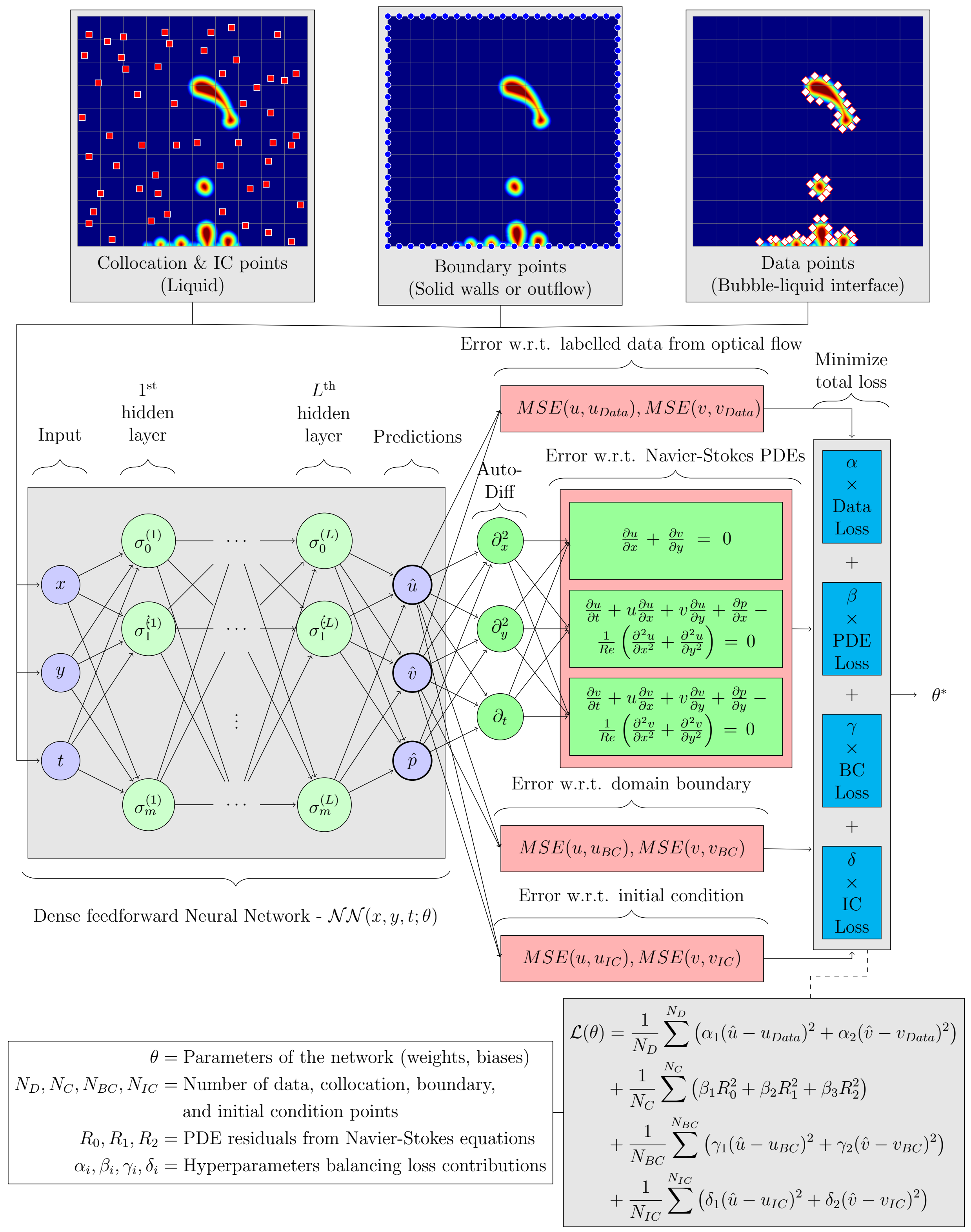

Collocation & IC points (Liquid)
Boundary points (Solid walls or outflow)
Data points (Bubble-liquid interface)
Error w.r.t. labelled data from optical flow
$MSE(u, u_{Data}), MSE(v, v_{Data})$
Minimize total loss
Input
1st hidden layer
Lth hidden layer
Predictions
Auto-Diff
Error w.r.t. Navier-Stokes PDEs
x y t
$\sigma_0^{(1)}$ $\sigma_1^{(1)}$ $\sigma_m^{(1)}$
$\sigma_0^{(L)}$ $\sigma_1^{(L)}$ $\sigma_m^{(L)}$
$\hat{u}$ $\hat{v}$ $\hat{p}$
∂_x^2 ∂_y^2 ∂_t
$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$
$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0$
$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0$
$\alpha \times$ Data Loss + $\beta \times$ PDE Loss + $\gamma \times$ BC Loss + $\delta \times$ IC Loss
θ^*
Error w.r.t. domain boundary
$MSE(u, u_{BC}), MSE(v, v_{BC})$
Error w.r.t. initial condition
$MSE(u, u_{IC}), MSE(v, v_{IC})$
Dense feedforward Neural Network - $\mathcal{NN}(x, y, t; \theta)$
θ = Parameters of the network (weights, biases)
N_D, N_C, N_{BC}, N_{IC} = Number of data, collocation, boundary, and initial condition points
R_0, R_1, R_2 = PDE residuals from Navier-Stokes equations
$\alpha_i, \beta_i, \gamma_i, \delta_i$ = Hyperparameters balancing loss contributions
$\mathcal{L}(\theta) = \frac{1}{N_D}\sum^{N_D}\left(\alpha_1(\hat{u} - u_{Data})^2 + \alpha_2(\hat{v} - v_{Data})^2\right) + \frac{1}{N_C}\sum^{N_C}\left(\beta_1 R_0^2 + \beta_2 R_1^2 + \beta_3 R_2^2\right) + \frac{1}{N_{BC}}\sum^{N_{BC}}\left(\gamma_1(\hat{u} - u_{BC})^2 + \gamma_2(\hat{v} - v_{BC})^2\right) + \frac{1}{N_{IC}}\sum^{N_{IC}}\left(\delta_1(\hat{u} - u_{IC})^2 + \delta_2(\hat{v} - v_{IC})^2\right)$

Figure 3.6: Architecture of physics-informed neural network for boiling problems. Given position (x, y) and time (t) (blue nodes on the left) the model predicts the hidden velocity (u, v) and pressure (p) values (blue nodes in centre) at those coordinates.

Chapter 4

Results

The prediction capabilities of PINNs for heat transfer problems can be illustrated with the help of training data from a pool-boiling simulation.

4.1 Training methodology

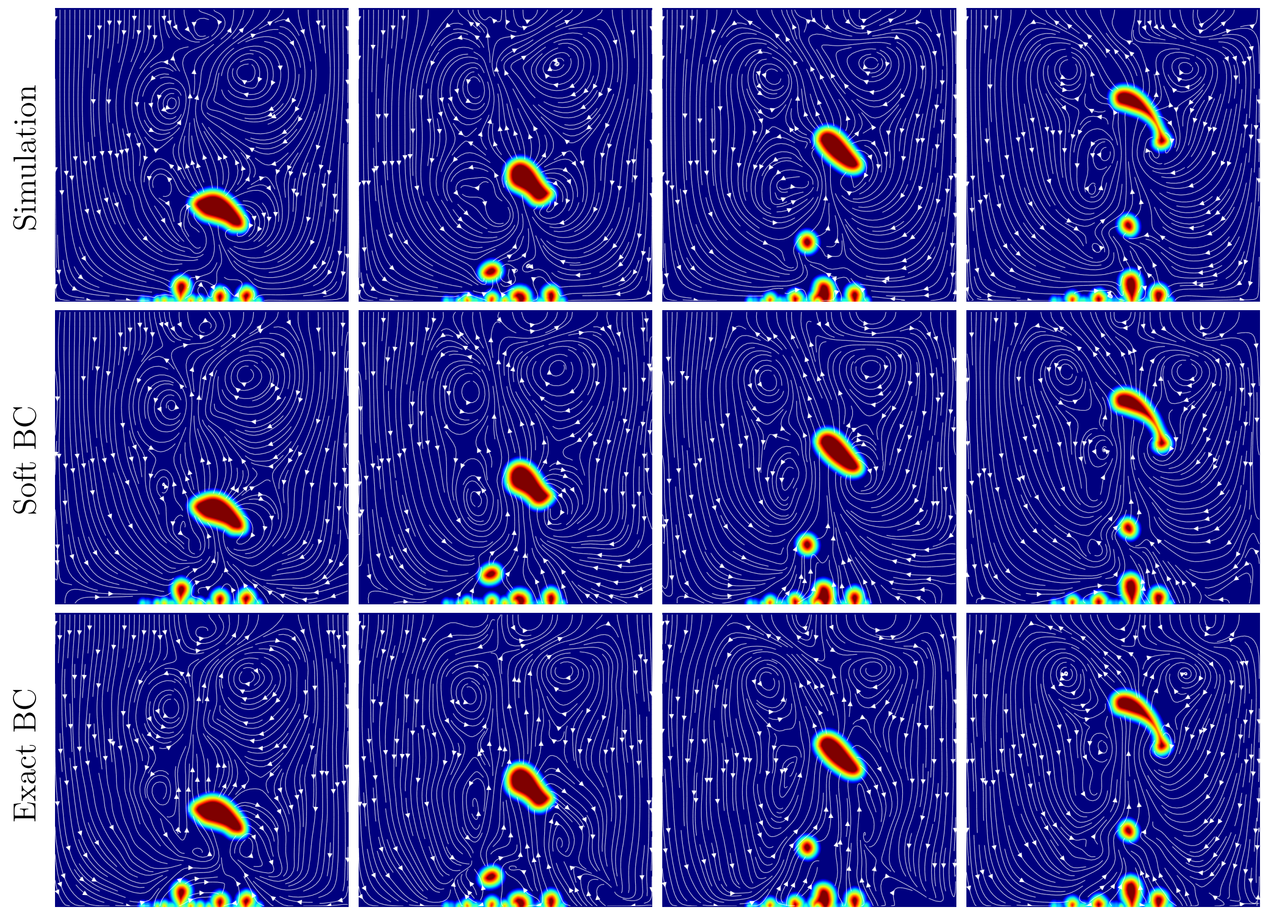
All models in this work were trained using a selection of randomly and uniformly sampled points that were extracted between timestamps $t_0 = 31$ (initial condition) and $t_8 = 39$ from the simulated pool-boiling dataset (section 3.2.1). The exact point ratios used for data, collocation, boundary, and initial condition losses are shown in table 4.1. An Adam optimizer with an initial learning rate of $lr_0 = 0.0005$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ was used to train each model for 100 epochs. The learning rate was automatically reduced by a factor of 0.1 using a

	Data	Collocation	Boundary Condition	Initial Condition
Number of points / frame	1×10^2	5×10^3	4×10^2	1×10^5
Hyperparameter	1.0	1.0	1.0	10.0

Table 4.1: Number of points and hyperparameters for each class of points.

scheduler with patience of 10 epochs and $\delta_{min} = 0.01$. Each mini-batch contained 128 points. All models used fully-connected (FC) architectures with 4 layers and 128 neurons per layer. Except for the last layer that used linear activations, all neurons used $GELU$ activations.

4.2 Velocity predictions



(a) Frame 32 (b) Frame 34 (c) Frame 36 (d) Frame 38

Figure 4.1: Velocity streamline plots. Simulations (top row), soft BC predictions (middle row), and exact BC predictions (bottom row).

The accuracy of velocity predictions can be evaluated by examining flow directions and magnitudes of predicted vector fields. As models were trained on data from discrete points in time, predicted fields can directly be compared against simulated counterparts.

4.2.1 Accuracy of velocity direction

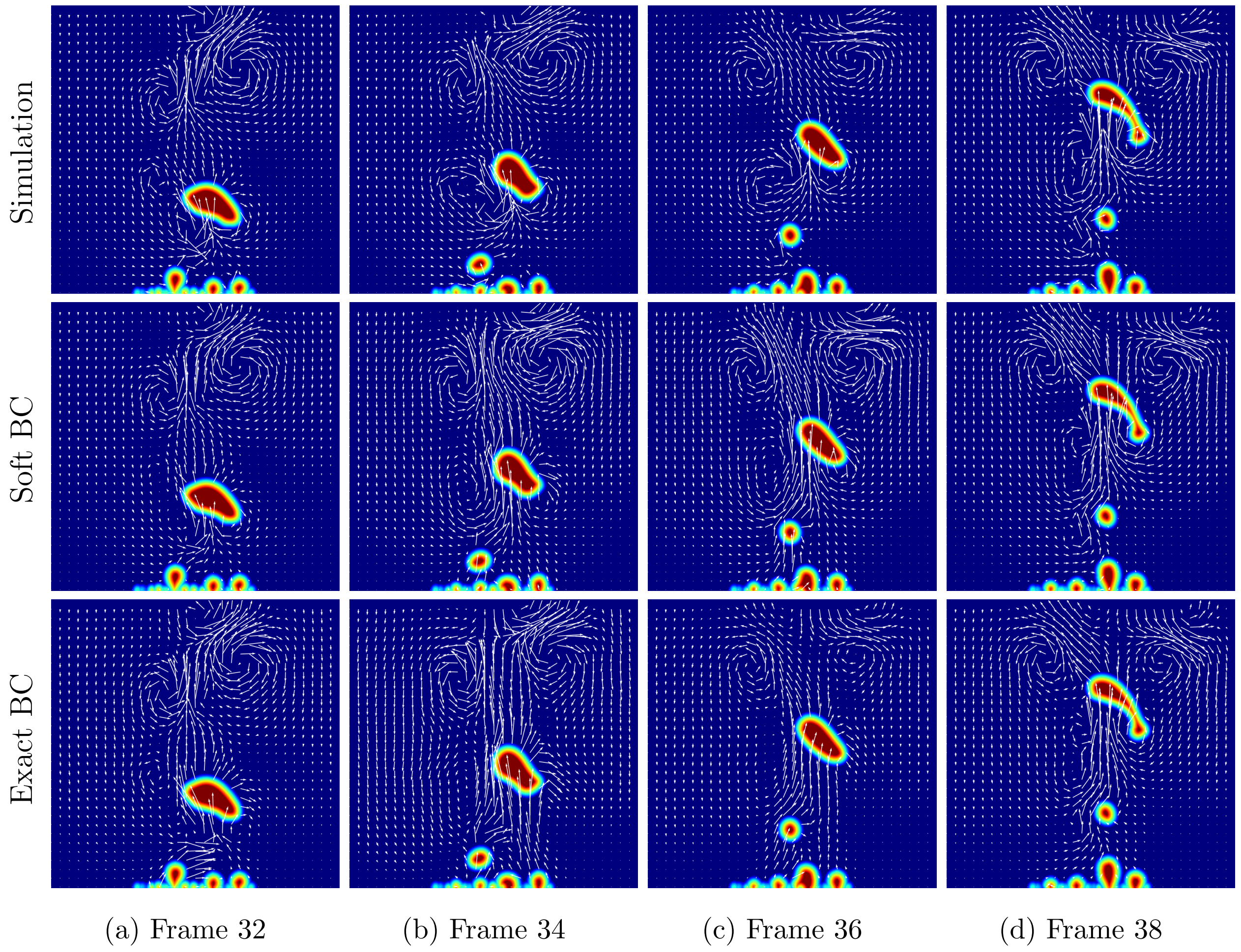


Figure 4.2: Velocity vector plots. Simulations (top row), soft BC predictions (middle row), and exact BC predictions (bottom row).

Near domain boundaries, the exact BC model achieves a significantly higher accuracy than the soft BC model. This can be observed in Figures 4.1 and 4.2 where velocity streamlines and vectors near domain boundaries of the exact BC model fulfil the no-slip condition that was imposed in the simulation much better than the soft BC model. The streamline plots for frames 34 and 36, for example, show flows parallel to walls in exact BC models (Figure 4.1 centre bottom) and unphysical flows out of walls in soft BC models (Figure 4.1 centre). Table 4.3 supports this qualitative observation too. The MSE losses for (u, v) at domain boundaries reach an accuracy in the range of 10^{-17} in exact and 10^{-2} in soft BC models.

4.2.2 Accuracy of velocity magnitude

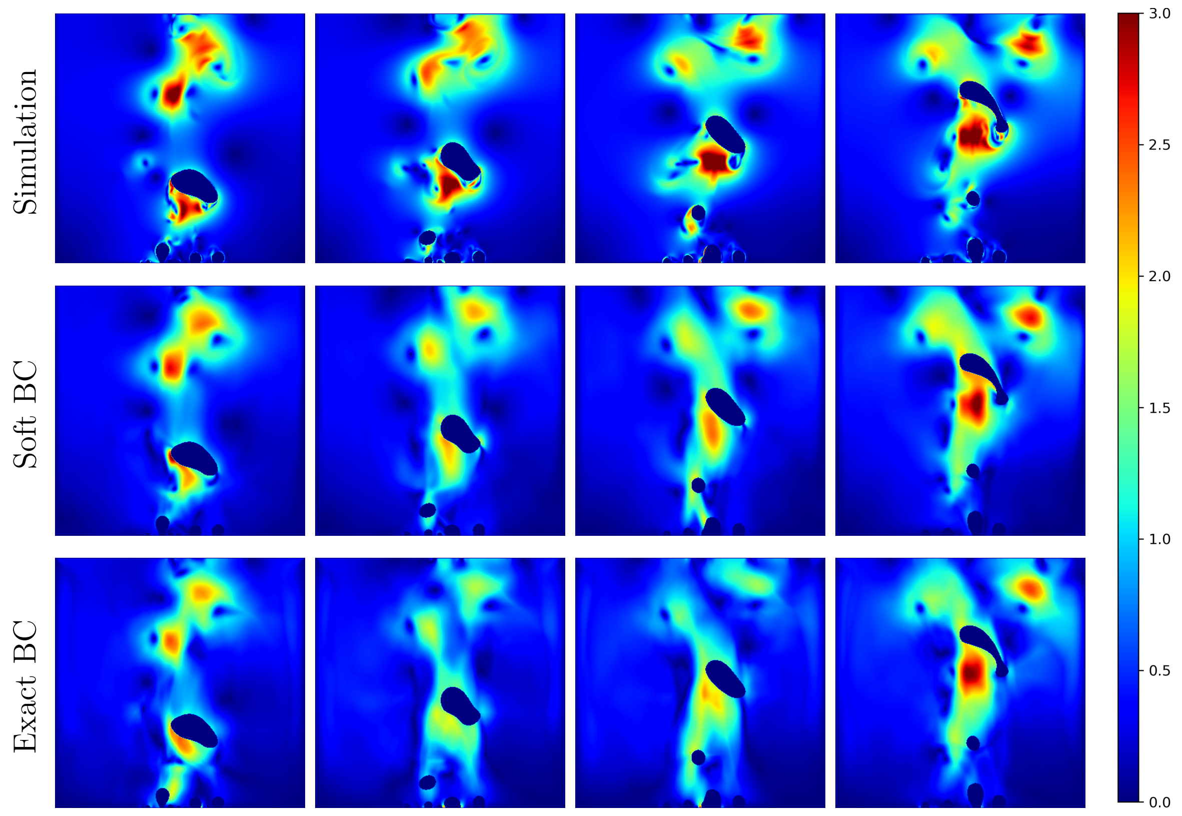
In both soft and exact BC models, the highest accuracy is achieved at the initial condition $t_0 = 31$. This behaviour is expected as at this point in time, the model has access to labels from the entire domain. The initial condition plots shown in Figure 4.4 illustrate qualitatively that both flow direction and magnitude at t_0 match simulated behaviour.

The error metrics from Table 4.2 confirm that models are most accurate at time t_0. Both soft and exact BC models achieve an initial condition accuracy in the range of 10^{-2} for velocity magnitude predictions.

Table 4.2 also shows that the prediction accuracy for timesteps following the initial condition decreases. This finding is expected as fewer labelled data points are available during these frames. The BC imposition method further influences prediction accuracy in future timesteps: While the exact BC model achieved a higher accuracy near domain boundaries, the soft BC model predicts velocities in the domain interior more precisely. This finding can be derived from the training losses and error metrics shown Table 4.3 and 4.2.

Error	Time		Boundary Condition	
			Soft	Exact
MAE	t_0		3.20×10^{-2}	4.25×10^{-2}
	t_1 to t_8	min	3.85×10^{-2}	5.04×10^{-2}
		max	1.45×10^{-1}	2.01×10^{-1}
RMSE	t_0		8.38×10^{-2}	9.91×10^{-2}
	t_1 to t_8	min	9.71×10^{-2}	1.25×10^{-1}
		max	2.39×10^{-1}	3.29×10^{-1}

Table 4.2: Prediction errors of velocity magnitude from soft and exact BC model.



(a) Frame 32 (b) Frame 34 (c) Frame 36 (d) Frame 38

Figure 4.3: Velocity magnitude plots .Simulations (top row), soft BC predictions (middle row), and exact BC predictions (bottom row).

4.3 Initial condition

During training test runs, it became evident that learning the initial condition accurately has a large influence on overall model accuracy. A more accurate IC naturally improved accuracy at timesteps close to the IC. Interestingly though, the timesteps most distant from the IC were improved considerably as well.

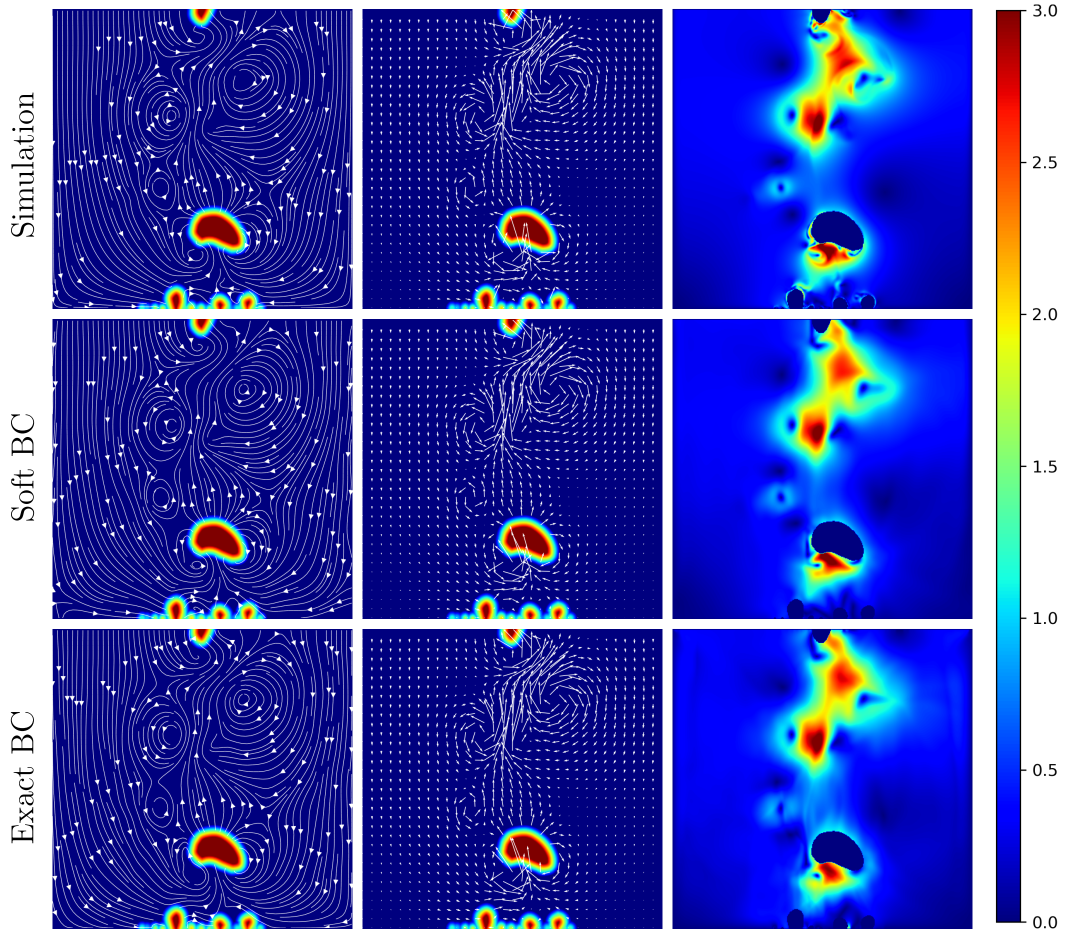


Figure 4.4: Velocity at initial condition. Simulations (top row), soft BC predictions (middle row), and exact BC predictions (bottom row).

Point Type	Error Type	Boundary Condition	
		Soft	Exact
Interface	Velocity U	6.42×10^{-2}	1.01×10^{-1}
	Velocity V	7.46×10^{-2}	1.13×10^{-1}
PDE	Continuity	1.05×10^{-2}	9.97×10^{-2}
	Momentum U	1.16×10^{-2}	1.48×10^{-1}
	Momentum V	1.29×10^{-2}	9.26×10^{-2}
Boundary Condition	Velocity U	2.00×10^{-2}	2.64×10^{-18}
	Velocity V	1.30×10^{-2}	3.36×10^{-17}
Initial Condition	Velocity U	7.98×10^{-3}	1.06×10^{-2}
	Velocity V	7.83×10^{-3}	1.08×10^{-2}

Table 4.3: Training losses for each class of points.

Chapter 5

Discussion

The models trained in this study have shown that, to some extent, hidden liquid physical quantities can be recovered with the help of PINNs. Depending on the BC imposition approach employed in the models, the overall or the accuracy near domain boundaries will be more accurate.

5.1 Challenges with PINNs for heat transfer problems

The flows in liquid phases of boiling fluids are highly turbulent and, as shown in Figure 4.3 exhibit a high variance in velocity magnitudes. While the PINNs from this study initially manage to capture the physics, maintaining prediction accuracy beyond the initial condition (IC) remains a challenge. With the proposed label extraction method, the number of labels to learn from decreases significantly after the IC. As such, the predicted flows in future timesteps are less accurate and dependent on physics learnt during the IC.

5.2 Dataset enhancement: Time-stepping

In preliminary tests using the architecture presented in chapter 4, it became evident that training with datasets with smaller timesteps t improves prediction accuracy. That is, by increasing the frame rate and having more fine-grained temporal information about the motion of bubbles, models can learn to more accurately approximate fluid flows. In pool-boiling data, decreasing nondimensional timesteps from $\Delta t = 1.0$ to $\Delta t = 0.1$ was achieved by capturing $10\times$ more frames during a simulation run. The exact optimal frame rate for a dataset is dependent on the velocity of bubbles though.

5.3 Training strategy: Adaptive loss balancing

Tuning loss contributions manually with hyperparameters is not optimal. More advanced training methods employ adaptive loss balancers that, depending on the evolution of a loss term, automatically adjust hyperparameters. If, for example, the data loss drops more quickly compared to the PDE loss, a loss balancer would increase the hyperparameter scaling the PDE loss contribution.

The models built for this work [2] support multi-objective loss balancing with random look-backs [3]. While no significant improvements were observable in tests with pool-boiling data, more tests with other training datasets should be carried out.

Bibliography

[1] Y. Abe and A. Iwasaki. Pool boiling under microgravity. *Advances in Space Research*, 13(7):165–168, 7 1993.

[2] S. Barschkis. bubble2vel, 2023.

[3] R. Bischof and M. Kraus. Multi-Objective Loss Balancing for Physics-Informed Deep Learning, 10 2021.

[4] S. Cai, Z. Wang, F. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis. Flow over an espresso cup: inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks. *Journal of Fluid Mechanics*, 915, 3 2021.

[5] A. Dhruv, E. Balaras, A. Riaz, and J. Kim. An investigation of the gravity effects on pool boiling heat transfer via high-fidelity simulations. *International Journal of Heat and Mass Transfer*, 180:121826, 2021.

[6] A. Dubey, K. Weide, J. O'Neal, A. Dhruv, S. M. Couch, J. A. Harris, T. Klosterman, R. Jain, J. Rudi, B. Messer, M. Pajkos, J. Carlson, R. Chu, M. Wahib, S. Chawdhary, P. M. Ricker, D. Lee, K. Antypas, K. M. Riley, C. S. Daley, M. K. Ganapathy, F. X. Timmes, D. M. Townsley, M. Vanella, J. Bachan, P. M. Rich, S. Kumar, E. Endeve, W. R. Hix, A. Mezzacappa, and T. Papatheodore. Flash-X: A multiphysics simulation software instrument. *SoftwareX*, 19:101168, 2022.

[7] S. M. S. Hassan, A. Feeney, A. Dhruv, J. Kim, Y. Suh, J. Ryu, Y. Won, and A. Chandramowlishwaran. BubbleML: A Multi-Physics Dataset and Benchmarks for Machine Learning, 2023.

[8] S. Hong, J. X. Wang, Z. Gao, and C. Dang. Review on state-of-the-art research in pool and flow boiling under microgravity. *Experimental Thermal and Fluid Science*, 144:110848, 6 2023.

[9] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. *CoRR*, abs/1612.01925, 2016.

[10] J. Kim and C. Lee. Prediction of turbulent heat transfer using convolutional neural networks. *Journal of Fluid Mechanics*, 882:A18, 2020.

[11] C. Konishi and I. Mudawar. Review of flow boiling and critical heat flux in microgravity. *International Journal of Heat and Mass Transfer*, 80:469–493, 1 2015.

[12] J. Lee, I. Mudawar, M. M. Hasan, H. K. Nahra, and J. R. Mackey. Experimental and computational investigation of flow boiling in microgravity. *International Journal of Heat and Mass Transfer*, 183:122237, 2 2022.

[13] W. Lee, G. Son, and H. Y. Yoon. Direct numerical simulation of flow boiling in a finned microchannel. *International Communications in Heat and Mass Transfer*, 39(9):1460–1466, 11 2012.

[14] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. *CoRR*, abs/2010.08895, 2020.

[15] H. Moon, K. Boyina, N. Miljkovic, and W. P. King. Heat Transfer Enhancement of Single-Phase Internal Flows using Shape Optimization and Additively Manufactured Flow Structures. *International Journal of Heat and Mass Transfer*, 177:121510, 10 2021.

[16] I. Mudawar and R. A. Houpt. Mass and momentum transport in smooth falling liquid films laminarized at relatively high Reynolds numbers. *International Journal of Heat and Mass Transfer*, 36(14):3437–3448, 9 1993.

[17] Nvidia. Nvidia Modulus Documentation, 2023.

[18] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[19] M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[20] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *LNCS*, volume 9351, pages 234–241, 10 2015.

[21] Y. Sato and B. Niceno. Pool boiling simulation using an interface tracking method: From nucleate boiling to film boiling regime through critical heat flux. *International Journal of Heat and Mass Transfer*, 125:876–890, 2018.

[22] Y. Suh, R. Bostanabad, and Y. Won. Deep learning predicts boiling heat transfer. *Scientific Reports*, 11, 2 2021.

[23] N. Sukumar and A. Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

[24] T. Tanaka, K. Miyazaki, and T. Yabuki. Observation of heat transfer mechanisms in saturated pool boiling of water by high-speed infrared thermometry. *International Journal of Heat and Mass Transfer*, 170:121006, 2021.

[25] E. Teodori, A. Moita, and A. L. N. Moreira. Characterization of pool boiling mechanisms over micro-patterned surfaces using PIV. *International Journal of Heat and Mass Transfer*, Volume 66:261, 2 2013.

[26] N. Trask, R. G. Patel, B. J. Gross, and P. J. Atzberger. GMLS-Nets: A framework for learning from unstructured data. *CoRR*, abs/1909.05371, 2019.

[27] H. Wang, R. Planas, A. Chandramowlishwaran, and R. Bostanabad. Mosaic flows: A transferable deep learning framework for solving PDEs on unseen domains. *Computer Methods in Applied Mechanics and Engineering*, 389:114424, 2 2022.

[28] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A Tensorial Approach to Computational Continuum Mechanics Using Object-Oriented Techniques. *Comput. Phys.*, 12(6):620–631, 11 1998.

[29] H. Zhai, Q. Zhou, and G. Hu. Predicting micro-bubble dynamics with semi-physics-informed deep learning. *AIP Advances*, 12(3):35153, 3 2022.