Chemical Engineering Journal 541 (2026) 177517

Contents lists available at ScienceDirect

Chemical Engineering Journal

journal homepage: www.elsevier.com/locate/cej

Estimating dense-packed zone height in liquid–liquid separation: A physics-informed neural network approach

Mehmet Velioglu [a,b], Song Zhai [e], Alexander Mitsos [c,a,d], Adel Mhamdi [d], Andreas Jupke [e,f], Manuel Dahmen [a,*]

[a] Institute of Climate and Energy Systems, Energy Systems Engineering (ICE-1), Forschungszentrum Jülich GmbH, Jülich 52425, Germany
[b] RWTH Aachen University, Aachen 52062, Germany
[c] JARA-ENERGY, Jülich 52425, Germany
[d] RWTH Aachen University, Process Systems Engineering (AVT.SVT), Aachen 52074, Germany
[e] RWTH Aachen University, Fluid Process Engineering (AVT.FVT), Aachen 52074, Germany
[f] Institute for Bio- and Geosciences (IBG-2), Forschungszentrum Jülich GmbH, Jülich 52425, Germany

ARTICLE INFO

Keywords:
Liquid–liquid separation
Physics-informed neural networks
State estimation
Extended Kalman Filter

ABSTRACT

Separating liquid–liquid dispersions in gravity settlers is critical in chemical, pharmaceutical, and recycling processes. The dense-packed zone height is an important performance and safety indicator but it is often expensive and impractical to measure due to optical limitations. We propose a framework to estimate phase heights by combining a PINN model with readily available volume flow measurements, without requiring phase height measurements during deployment. To this end, a physics-informed neural network (PINN) is first pretrained on synthetic data and physics equations derived from a low-fidelity (approximate) mechanistic model to reduce the need for extensive experimental data. While the mechanistic model is used to generate synthetic training data, only volume balance equations are used in the PINN, as incorporating droplet coalescence and sedimentation submodels would be computationally prohibitive. The pretrained PINN is then fine-tuned with scarce experimental phase height and flow-rate data to capture the actual dynamics of the separator. We then deploy the differentiable PINN as a predictive model in an Extended Kalman Filter inspired state estimation framework, enabling the phase heights to be tracked and updated using flow-rate measurements only. We first test the two-stage trained PINN by forward simulation from a known initial state against the mechanistic model and a non-pretrained PINN. We then evaluate phase height estimation performance with the filter, comparing the two-stage trained PINN with a two-stage trained purely data-driven neural network. All model types are trained and evaluated using ensembles to account for model parameter uncertainty. In all evaluations, the two-stage trained PINN yields the most accurate phase-height estimates.

1. Introduction

The separation of liquid–liquid dispersions in a horizontal gravity separator is an essential step in many processes, such as solvent extraction in the chemical, pharmaceutical, and recycling industries, as well as grease and oil separation in the food industry and water treatment [1]. Liquid–liquid dispersions entering these separators split into their coherent phases under the influence of gravity-driven sedimentation and coalescence. Material systems characterized by a slower coalescence than sedimentation rate form a dispersion layer known as a dense-packed zone (DPZ) [2]. Accumulation of the DPZ height above a critical level floods the separator, disrupting the separation process and leading to complications in subsequent unit operations due to poor separation efficiency [3–5]. Measuring the DPZ height and controlling it below a critical height is of utmost importance to prevent flooding, yet gauge glasses in industrial separators allow only limited capacity to measure the DPZ height [6].

Estimating the DPZ height by a suitable liquid–liquid gravity separator (LLS) model has the potential to detect flooding caused by DPZ accumulation. The steady-state LLS model of Henschke [7] remains the state-of-the-art in describing the DPZ distribution along the LLS but dynamic models are limited and not validated [8]. Backi et al. [9] introduced a lumped zero-dimensional LLS model that has the potential of describing DPZ trajectories (i.e., temporal evolution of DPZ under dynamic operating conditions), yet experimental validation is missing due to lack of suitable data. Both steady-state and lumped models have shown less than 25% accuracy in predicting flow rates

* Corresponding author.
E-mail address: m.dahmen@fz-juelich.de (M. Dahmen).

https://doi.org/10.1016/j.cej.2026.177517
Received 28 January 2026; Received in revised form 27 April 2026; Accepted 18 May 2026
Available online 22 May 2026

at which flooding occurs [3]. These shortcomings highlight the need for alternative approaches, such as data-driven modeling. However, purely data-driven methods typically require extensive amounts of data to generalize reliably [10].

Physics-informed neural networks (PINNs) [11] combine first-principle and data-driven modeling by embedding known physical relationships within the data-driven training process. Thus, PINNs can generalize from scarce training data while adhering to the governing physics. Our previous work [12] presented an *in-silico* proof of concept in which mechanistic model data were used as ground truth to demonstrate that a PINN trained with limited data and incomplete physics can predict DPZ height over short time intervals. However, estimation of DPZ height from experimental settler data for longer time intervals remains unexplored.

One possible strategy to further reduce the demand for experimental data is to first train PINN models on physics equations and/or synthetic data generated from a low-fidelity (i.e., approximate) mechanistic model, and then fine-tune them using high-fidelity data. Chakraborty [13] showed that a two-stage training procedure for PINNs involving a low-fidelity physics model for pretraining and high-fidelity data for fine-tuning can improve prediction accuracy when the known governing physics equations are only approximate and the high-fidelity data is scarce. Wang et al. [14] compared different transfer learning schemes for PINNs, including full fine-tuning and parameter-efficient updates, to adapt PINNs across different tasks. Mustajab et al. [15] have examined high-frequency and multi-scale problems, where pretraining on baseline low-frequency models provides stable initialization that enhances PINN accuracy when predicting high-frequency problems. Prantikos et al. [16] proposed selecting pretraining datasets based on domain similarity for nuclear reactor applications, illustrating how task relevance in PINNs accelerates the training in the fine-tuning step. Related developments outside the PINN literature also point to the value of combining simulation-based pretraining with experimental data, for example in process control of froth flotation systems under partial observability with Gaussian processes [17], or in reinforcement learning where sample efficiency is critical [18].

While the PINN provides a predictive model of the system dynamics, it cannot by itself reconstruct the true system state under realistic operating conditions, where phase heights are not measurable, initial conditions are unknown, and flow-rate measurements are noisy. Therefore, an estimation scheme is required to fuse model predictions with measurements and enable real-time phase height estimation. To this end, Arnold and King [19] showed that PINNs can be effectively integrated within an Extended Kalman Filter (EKF) [20], since Jacobians can be computed through automatic differentiation [21]. Cuomo et al. [22] combined a PINN with the EKF to predict structural displacements in railway motion. In vehicle applications, Tan et al. [23] combined a PINN with the Unscented Kalman Filter (UKF) [24] to estimate vehicle attitude, velocity and position with increased accuracy, while [25] demonstrated the use of a PINN-based predictive model as state-transition model in UKF for dynamic models such as a double pendulum system. Fang and Yu [26] fine-tuned a pretrained dynamic model with physics guidance to represent hybrid dynamics before embedding them in an EKF framework for vehicle state estimation. These examples suggest that PINNs can serve as surrogates for predictive models when governing equations are incomplete, while the filtering step accounts for measurement noise and model uncertainty. To complement this, ensemble learning [27,28] constitutes an additional technique for improving prediction robustness and uncertainty quantification.

Although PINNs have been combined with Kalman-type filters for state estimation, their application for estimating DPZ heights in pilot-scale LLS has not yet been investigated. Demonstrating that DPZ height can be estimated from readily available volume flow-rate measurements with the aid of a PINN would provide significant benefits for LLS operation.

Table 1
Physical properties of the saturated biphasic plant system at 30 °C.

	Density ρ	Dynamic viscosity η	Interfacial tension γ
Water	996 kg/m^3	0.82 mPa s	8.2 mN/m
1-octanol	825 kg/m^3	6.04 mPa s	

This work aims to estimate the phase heights in a pilot-scale LLS. The considered system is a horizontal cylindrical gravity separator (length = 1 m, radius = 0.1 m), where the dispersion flows along the axial direction and separates into a light phase, a dispersion layer (DPZ), and a heavy phase. We conducted step-response experiments in the pilot-scale LLS by varying the volume flow rate. These experiments capture the transient behavior of DPZ heights and provide data for training, validation, and testing (Section 2). The low-fidelity mechanistic LLS model (Section 3) from our previous work [12] combines simple volume balance equations with complex droplet coalescence and sedimentation submodels. The model is only approximate, as it assumes constant phase heights along the separator length (band-shaped DPZ), neglects axial (convective) transport of the DPZ, droplet–droplet coalescence, and axial and vertical variations in Sauter mean diameter. We generate synthetic pretraining data using the mechanistic model, but use only the volume balance equations as physics regularization in the PINN, as the droplet coalescence and sedimentation submodels are computationally challenging to implement in the PINN due to the necessary discretization of the axial length and droplet classes. A PINN model of the LLS dynamics is then developed by pretraining on synthetic data from the low-fidelity mechanistic model and then fine-tuning with experimental phase-height and flow-rate data (Section 4), similar to the training strategy from [13]. The trained PINN is subsequently integrated into an EKF-inspired state estimator to estimate phase heights solely from volume flow-rate measurements during deployment (Section 5). Conclusions and an outlook on future work are presented in Section 6.

This work integrates elements from previous studies across different domains, including our earlier works on the experimental LLS setup [3], the mechanistic LLS model [12], and the PINN model that is largely based on [12]. Moreover, we describe the established EKF method [20]. For self-containment and readability, relevant portions of these works are repeated here with appropriate citation.

2. Experimental setup

2.1. Materials and experimental liquid–liquid separator setup

All experiments are conducted in the pilot-scale continuous LLS at 30 °C shown in Fig. 1. 1-Octanol (99%, Häberle Labortechnik GmbH & Co. KG) is dispersed in deionized water (conductivity 13 µS/cm). Physical properties of the saturated liquids at 30 °C are listed in Table 1. The effective length of the separator is set to 1 m by a weir. The dispersion is generated in the stirred tank R1, inlet volume flows are controlled by frequency-controlled peripheral pumps P1 and P2, temperature is controlled by the heater W12 and W22, and the interfacial position of the separator is controlled by the check valve V208 which manipulates the heavy phase outlet volume flow. Outlet and inlet streams of the separator are recycled by the storage vessels B1 and B2. Ingoing and outgoing flow rates of the aqueous and organic phase are recorded by installed Coriolis flow meters FIRC01, FIRC02, FIR01, FIR02. Heights of the DPZ were measured by external cameras along the separator QIR03, QIR04, QIR05. A complete list of used apparatuses and measurement devices can be found in [3].

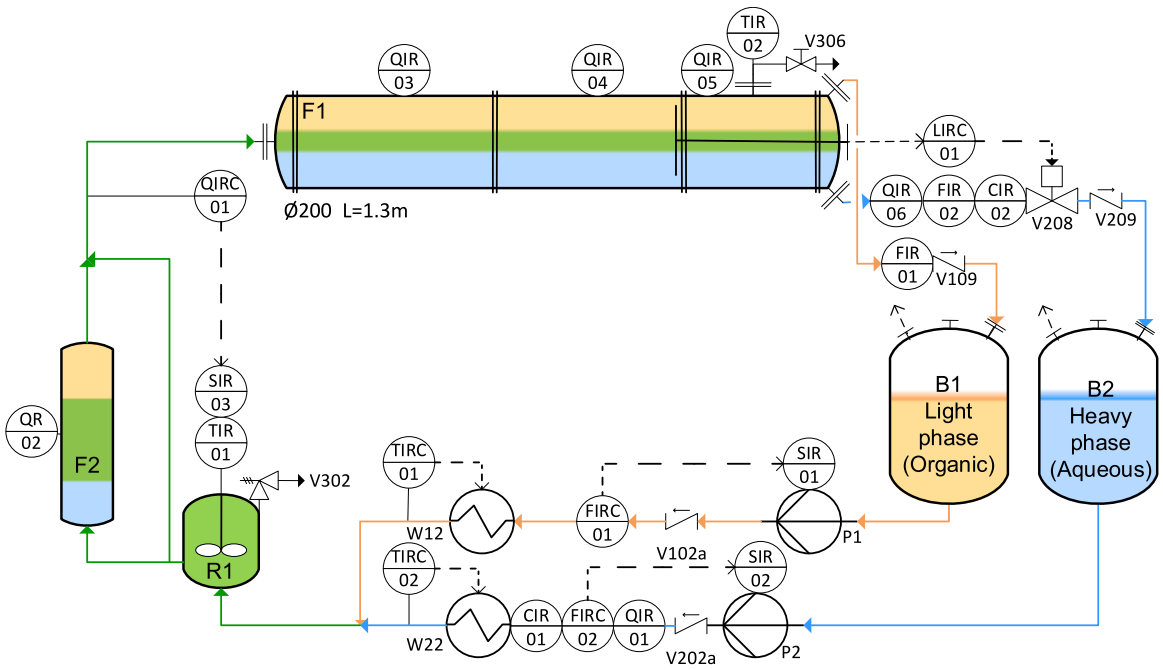


Fig. 1. Piping and instrumentation diagram of the experimental setup with DN 200 separator in the RWTH Aachen Fluid Process Engineering (AVT.FVT) lab. Blue, orange, and green colors mark the aqueous, organic, and dispersion phases (DPZ).
Source: Reproduced from [3].

2.2. Optical measurement of the dense-packed zone height

The phase heights for both the heavy (water) phase and the DPZ are determined from two camera recordings (cf. QIR03, QIR04 in Fig. 1) taken at four distinct axial positions from each camera. The exact detection positions are shown in Fig. 2. Fig. 3 shows the optically measured phase heights for the training trajectory, where the quantities of interest are the average heavy-phase and DPZ heights over the available detection positions; these averaged phase heights are the variables estimated in the present study.

Similar to the detection algorithm for detecting drop size distributions in endoscope images [29,30] or external camera images [31], heights are detected by the convolutional neural network based object detection algorithm YOLOv8 trained on manually labeled lab data [32]. The manual labeling itself contains errors due to the uneven shape of the drops in the DPZ. Moreover, detection errors can occur due to illumination conditions, often appearing as spikes in the recorded trajectories (cf. Fig. 3). Furthermore, occasional missing measurements, resulting from object detection errors, appear as discontinuities in the recorded trajectories (cf. Fig. 3). An exemplary detection is shown in Fig. 2.

2.3. Design of experiments for separator dynamics

The experiments are conducted to investigate the dynamic accumulation of the DPZ phase in the pilot-scale LLS under variations in total volumetric flow rate. Previous studies [3,4,33,34] identified the flow rate as the dominant operating parameter. The flow rate can equivalently be expressed in terms of the theoretical residence time in the separator, defined as $\tau_{res} = V_{sep}/Q_{in}$, which represents the time available for droplet coalescence and sedimentation and thus governs DPZ formation. All other conditions, including temperature, stirrer speed, feed phase fraction, and interfacial position, are kept constant at 30 °C, 600 rpm, 0.5, and 100 mm, respectively. System input is provided as stepwise changes in flow rate, with both the temporal step size (duration of constant operation) and the flow-rate step size (magnitude of change) varied across trajectories. One trajectory extends the flow-rate range beyond that of the others and is used to assess the extrapolation capability of the model in that regime. A practical example in an industrial LLS could be upstream disturbances causing unusually high or low flow rates. Note that the findings cannot be generalized to more extreme variations in flow rates or other types of extrapolation, e.g., changes in phase fraction or changes in temperature.

In total, four distinct trajectories were acquired, with detailed step durations and flow-rate levels summarized in Table 2. Before each experiment, the plant was heated and circulated for at least one hour until homogeneous conditions were established. Each trajectory was measured only once as repeated experiments under identical conditions would have required repeating the time-intensive preconditioning of the pilot-scale setup. This results in a limited and noisy dataset, which is addressed by the proposed framework in Sections 4 and 5. Specifically, data scarcity is mitigated through physics-based regularization and pre-training on synthetic data (cf. Sections 4.2 and 4.4), while ensemble learning (cf. Section 4.6) accounts for model uncertainty, and filtering (cf. Section 5.1) accounts for both model uncertainty and measurement noise. The use of single-run experimental data therefore not only reflects practical constraints, but it is intended to demonstrate that the proposed modeling approach can achieve generalization under data-lean scenarios.

2.4. Post-processing of phase height measurements

Our PINN modeling approach (cf. Section 3) assumes a band-shaped operation, meaning that phase heights would be constant along the separator length. This assumption reduces the separator to a lumped system by eliminating axial height variations, effectively replacing a PDE-based description with an ODE system. Such lumped models are conceptually and computationally easier to handle. Although the band-shaped assumption is violated at higher volume flow rates, as evidenced by the wedge-shape DPZ formed at QIR04 in Fig. 2, we use the mean of the available measurements to represent the average DPZ height to be modeled and train a separate neural network to estimate the DPZ

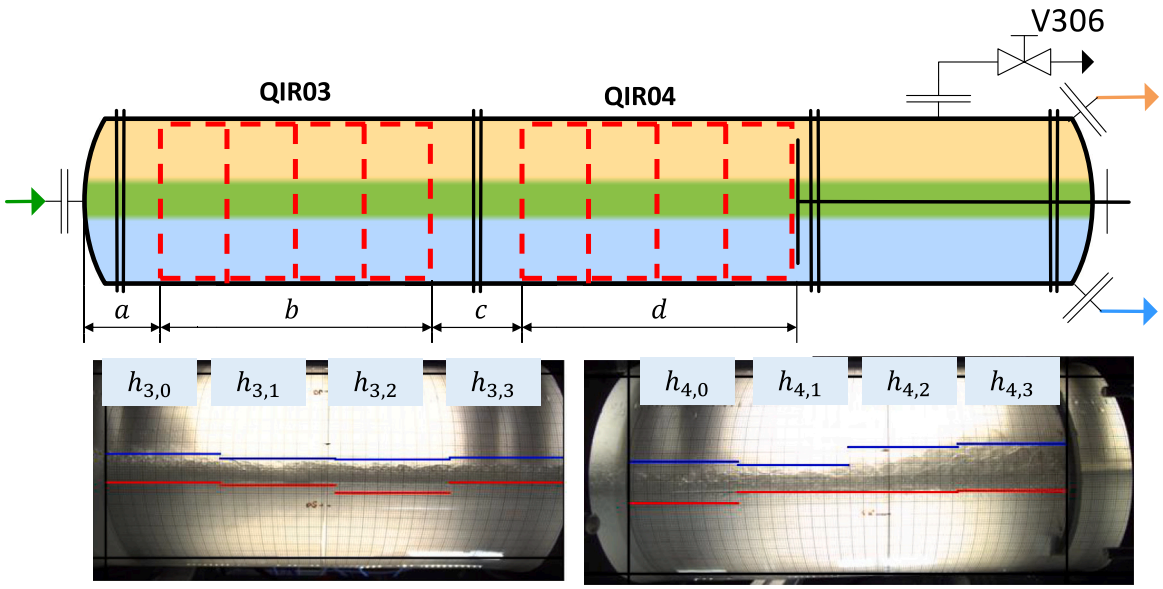


Fig. 2. Detection of DPZ heights along the separator with effective length of 1 m. Images refer to detections in QIR03 (left) and QIR04 (right). In each image, four heights are detected for same widths. The distances between inlet and first height detection, width of QIR03, width between QIR03 and QIR04, and width of QIR04 is $a = 21$ cm, $b = 36$ cm, $c = 10$ cm, $d = 33$ cm, respectively.

Table 2
Summary of setpoints for the experimental trajectories.

Trajectory	Time between step change	Volume flow rate Q_{in}/ L/s									
		1	2	3	4	5	6	7	8	9	10
1 – training	2.00 min	0.278	0.417	0.556	0.417	0.278					
2 – validation	1.50 min	0.278	0.417	0.556	0.417	0.278					
3 – test (interpolation)	2.00 min	0.278	0.347	0.417	0.486	0.556	0.486	0.417	0.347	0.278	
4 – test (extrapolation)	2.00 min	0.208	0.347	0.486	0.625	0.486	0.347	0.208			

height at the separator outlet, which is most critical for flooding, based on the average DPZ height (cf. Section 5.2).

Object detection based phase height readings arrive at irregular intervals, typically every 2–3 s; we therefore linearly interpolate the phase height measurements to 1-second resolution to match the resolution of the flow-rate measurements.

We acquired four trajectories with varying time duration from the experimental gravity settler setup, as demonstrated in Table 2. From those, we use trajectory #1 (Fig. 4(a)) for training, trajectory #2 (Fig. 4(b)) for validation, and trajectories #3 and #4 for testing. Specifically, trajectory #3 (Fig. 4(c)) is used for the interpolation test, as it contains inlet volume flow rates within the range $Q_{in} \in [0.278, 0.556]$ L/s, but has values at intermediate levels not encountered during training (e.g., 0.347 L/s in Table 2). Trajectory #4 (Fig. 4(d)) is used for the extrapolation test, as it includes inlet volume flow rates Q_{in} outside the range observed in the training trajectory (e.g., 0.625 L/s in Table 2). The validation trajectory is used in hyperparameter tuning and tuning for the EKF application (cf. Section 5.1). The respective bounds for all variables are given in Table 3. The resulting experimental dataset is denoted by D_{exp}.

3. Low-fidelity mechanistic LLS model

The low-fidelity mechanistic LLS model was introduced in [12] and is visualized in Fig. 5. It divides the settler content into three zones: the light phase, the DPZ, and the heavy phase. The geometry of the entire fluid is defined by an effective length of $L = 1$ m and a radius of $r = 0.1$ m. The total inlet flow rate, Q_{in}, enters from the left with a Sauter mean diameter $d_{32,in}$ and organic phase fraction of $\epsilon_{in} = 0.5$,

Table 3
Operating ranges for the phase heights and volume flow rates. The lower bound is denoted by lb, and the upper bound is denoted by ub.

Variable	Unit	Interpolation bounds		Extrapolation bounds	
		lb	ub	lb	ub
h_{HP}	m	0.071	0.091	0.067	0.100
h_{DP}	m	0.023	0.059	0.019	0.069
Q_{in}	L/s	0.245	0.563	0.175	0.644
Q_{top}	L/s	0.105	0.286	0.069	0.321
Q_{bot}	L/s	0.117	0.295	0.083	0.356

and is assumed to disperse completely into the heavy phase. The inlet drop size distribution is modeled by the self-similarity assumption with a normalized standard deviation of the volume-based log-normal drop size distribution $\sigma_{selfsimilar} = \sigma/d_{32,in} = 0.32$ [35,36].

The DPZ is assumed to be band-shaped, meaning constant phase heights, h_{HP} (heavy phase) and h_{DP} (DPZ), along the separator length. Within the DPZ, droplet–droplet coalescence is assumed to be negligible, while droplet–interface coalescence results in a volume flow $Q_c(t)$ into the light phase. The dispersed droplets in the heavy phase rise as a droplet swarm with swarm exponent $n_{swarm} = 2$ [4,37,38], resulting in a volume flow $Q_s(t)$ into the DPZ. Both the coalescence $Q_c(t)$ and sedimentation $Q_s(t)$ flow rates are derived from complex submodels that discretize the axial length of the separator into 200 spatial discretization elements and the droplet size distribution into 50 droplet size classes. Additionally, the volume flow of water from the heavy phase into the DPZ is expressed by the coalescence rate $Q_c(t)$ and the sedimentation rate $Q_s(t)$, by assuming a constant DPZ hold-up of $\bar{\epsilon}_{DP} = 0.9$ in accordance with the separator model proposed

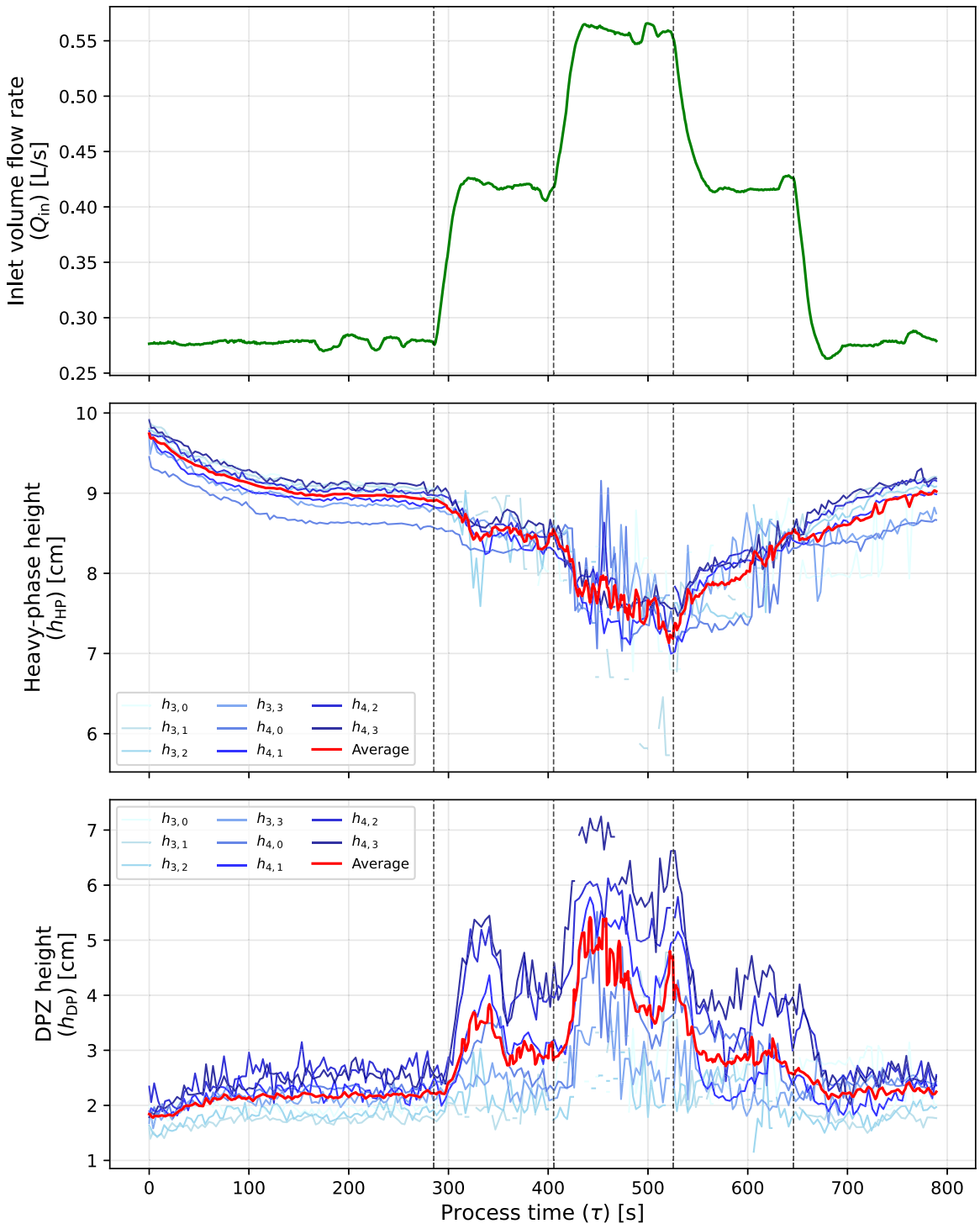


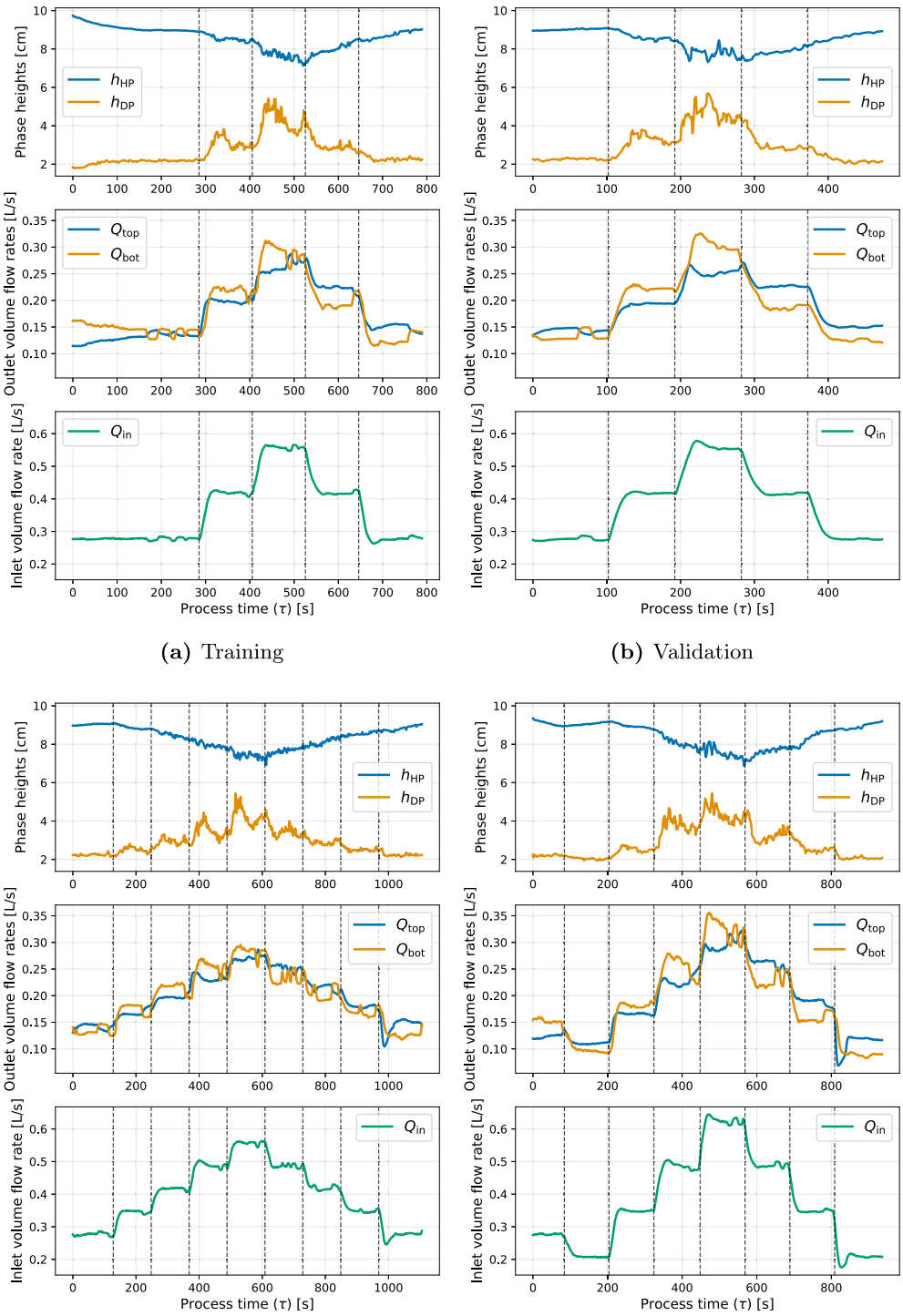
Fig. 3. Pre-processed inlet volume flow rate and phase height measurements for the **training trajectory**, measured at eight different positions at **QIR03** and **QIR04** (see Fig. 2). Vertical dashed lines denote the times of inlet volume flow setpoint changes.

by Henschke [7]. The convective transport of the DPZ is also neglected, as well as axial variation of the Sauter mean diameter and its effect on the DPZ.

The separated light and heavy phases exit with outlet volume flow rates Q_{top} and Q_{bot}, respectively, both assumed to be free of entrained droplets. The total volume of the three phases is kept constant and equal to the separator volume, yielding the volume balance given by Eq. (1a). The mechanistic LLS model takes as inputs the initial phase heights $h_{HP}(t = 0)$ and $h_{DP}(t = 0)$, the inlet volume flow rate Q_{in}, the bottom volume flow rate Q_{bot}, the Sauter mean diameter at inlet $d_{32,in}$, and provides as outputs the phase heights $h_{HP}(t)$, $h_{DP}(t)$, the top outlet volume flow rate $Q_{top}(t)$, the coalescence rate $Q_c(t)$ and sedimentation rate $Q_s(t)$. Further details of the mechanistic LLS model can be found in [12].

The mechanistic LLS model [12] is based on the following volume balance equations obtained after transforming the volume of a cylindrical segment to the height of each segment, similar to [9]:

$$0 = g_{sep}(Q_{in}(t), Q_{bot}(t), Q_{top}(t)) = Q_{in}(t) - Q_{bot}(t) - Q_{top}(t), \quad (1a)$$



(a) Training **(b)** Validation

(c) Interpolation test **(d)** Extrapolation test

Fig. 4. Post-processed experimental trajectories obtained from the gravity settler. The quantities h_{HP} and h_{DP} denote the average heavy-phase height and DPZ height, respectively, computed over the eight detection positions. Q_{in} denotes the inlet volume flow rate, and Q_{bot} and Q_{top} denote the bottom and top outlet volume flow rates, respectively. Vertical dashed lines denote the times of inlet volume flow setpoint changes.

$$\dot{h}_{DP}(t) = f_{DP}(h_{HP}(t), h_{DP}(t), Q_{in}(t), Q_{bot}(t), Q_c(t), Q_s(t)) \tag{1b}$$
$$= \frac{Q_{in}(t) - Q_{bot}(t) - Q_c(t)}{2L\sqrt{\left(h_{DP}(t) + h_{HP}(t)\right)\left(2r - h_{DP}(t) - h_{HP}(t)\right)}}$$
$$- \frac{Q_{in}(t) - Q_{bot}(t) - Q_s(t)\frac{1}{\bar{\epsilon}_{DP}} + Q_c(t)\frac{1-\bar{\epsilon}_{DP}}{\bar{\epsilon}_{DP}}}{2L\sqrt{h_{HP}(t)(2r - h_{HP}(t))}},$$

$$\dot{h}_{HP}(t) = f_{HP}(h_{HP}(t), h_{DP}(t), Q_{in}(t), Q_{bot}(t), Q_c(t), Q_s(t)) \tag{1c}$$
$$= \frac{Q_{in}(t) - Q_{bot}(t) - Q_s(t)\frac{1}{\bar{\epsilon}_{DP}} + Q_c(t)\frac{1-\bar{\epsilon}_{DP}}{\bar{\epsilon}_{DP}}}{2L\sqrt{h_{HP}(t)(2r - h_{HP}(t))}},$$

Note that the submodels governing the coalescence and sedimentation rates, $Q_c(t)$ and $Q_s(t)$, are not included here due to their complexity; the complete formulations are provided in the Supplementary Materials

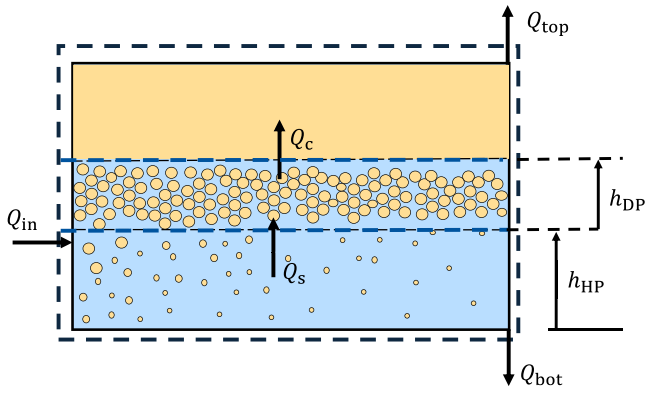


Fig. 5. Mechanistic model schematic of the pilot-scale liquid–liquid separator. *Source:* Adapted from [12].

(SM4) of Velioglu et al. [12]. Their dependencies on the states and parameters are expressed as:

$$Q_s(t) = g_s(h_{HP}(t), d_{32,in}, \sigma_{self\,similar}, n_{swarm}, \epsilon_{in}, \Delta\rho, \eta_{HP}, r, L), \tag{2a}$$

$$Q_c(t) = g_c(h_{HP}(t), h_{DP}(t), d_{32,in}, \sigma_{self\,similar}, n_{swarm}, \epsilon_{in}, \Delta\rho, \eta_{HP}, \gamma, r, L) \tag{2b}$$

where $\Delta\rho$ is the density difference between the two liquids, η_{HP} is the dynamic viscosity of the heavy phase and γ is the interfacial tension (cf. Table 1).

Using the mechanistic LLS model described above, we generate 1000 segments with a duration of 1 s where we sample the model inputs with Latin Hypercube Sampling (LHS) [39] from the bounds provided in Table 3. We use all of the segments as pretraining data. The simulation segments have a constant step size of $\Delta t = 0.1$ s for the PINN (cf. Section 4). The pretraining dataset is referred to as $\mathcal{D}_{sim}$. We choose the bounds for initial states and control inputs that correspond to minima and maxima of state values in the experimental trajectories (cf. Table 3), including the extrapolation trajectory. The predictive capability of the low-fidelity mechanistic model is later evaluated by forward simulation against experimental trajectories, alongside the proposed PINN approach (see Section 4.6).

4. Physics-informed neural network model of the settler dynamics

In this section, we present the dynamic modeling of the gravity settler through the PINN approach. In Section 4.1, we present the PINN architecture. The physics equations and the associated modeling assumptions are presented in Section 4.2 and data handling for the PINN training is presented in Section 4.3. Section 4.4 discusses the PINN training strategy, Section 4.5 presents the purely data-driven neural network for comparison, and Section 4.6 presents prediction results and comparison of different models.

The PINN model used in this work is largely based on our LLS PINN model introduced in [12]. We employ the same network architecture (a feedforward neural network), activation functions, two-optimizer training scheme and the dynamic weighting scheme. For the sake of self-containment, we restate these components in this section. The notable changes introduced in this work are as follows: (i) In contrast to the previous study, which used exclusively synthetic simulation data, we now also use experimental data for both training and testing. (ii) The outlet volume flow rates, Q_{bot} and Q_{top}, are treated as PINN outputs instead of PINN inputs, whereas the inlet volume flow rate, Q_{in}, is treated as a PINN input (control action) instead of a PINN output. (iii) A shorter PINN time horizon is used to match the resolution of the experimental data. (iv) The Sauter mean diameter at the inlet, $d_{32,in}$, and the coalescence parameter, r_v, are no longer used as inputs; the former did not contribute substantial predictive capability, and the latter depends primarily on temperature, which is held constant throughout this work. (v) Phase heights are no longer modeled from the bottom of the separator. (vi) Loss terms are conceptually the same, although exact terms are different due to changes mentioned in points (ii) and (v). (vii) A new two-stage training scheme is introduced (pretraining and fine-tuning).

4.1. Model architecture

We define a PINN with learnable parameters θ that takes PINN time $t \in [0, T]$, initial heights $\mathbf{x}(0) = [h_{DP}(0), h_{HP}(0)]^\top$ and inlet volume flow (control) $\mathbf{u} = [Q_{in}]$ as input and outputs the measurable settler heights $\mathbf{x}(t) = [h_{DP}(t), h_{HP}(t)]^\top$, measurable outlet volume flows $\mathbf{y}(t) = [Q_{bot}(t), Q_{top}(t)]^\top$ and immeasurable internal flows $\mathbf{z}(t) = [Q_c(t), Q_s(t)]^\top$:

$$[\hat{h}_{HP}(t), \hat{h}_{DP}(t), \hat{Q}_{bot}(t), \hat{Q}_{top}(t), \hat{Q}_c(t), \hat{Q}_s(t)]^\top = \mathrm{PINN}_\theta(t, h_{HP}(0), h_{DP}(0), Q_{in}) \tag{3}$$

The hat ($\hat{}$) notation denotes a PINN prediction. Note that the PINN time $t \in [0, T]$ differs from the process time τ. Specifically, we partition the process time domain of a *trajectory* into shorter *segments* of duration $T = 1$ s, which corresponds to the length of PINN time $t \in [0, T]$ (cf. Fig. 6) [12]. This segmentation keeps the control actions constant within each segment.

The given input configuration allows the PINN model to be trained to extensively cover the state and control action spaces. The PINN can be used for multi-step predictions by iteratively feeding the predicted final state of one segment as the initial condition for the next segment (forward simulation, cf. Fig. 6).

To determine the architecture hyperparameters of the PINN, we utilize a grid search varying the following hyperparameters: number of the hidden layers $\in \{1, 2, 3, 4\}$ and width of hidden layers (number of nodes) $\in \{16, 32, 64, 128\}$. We find that a network with 2 hidden layers and 32 nodes performs the best on the validation trajectory. The parameters of the PINN models (θ) are initialized with the Xavier normal distribution [40]. All input variables are normalized to lie within the range $[-1, 1]$ prior to training. After each layer except for the output layer, we use the tanh activation function. We use the sigmoid activation function for the output layer to prevent the argument of the square root in the denominator of Eqs. (1b) and (1c) from attaining negative values during PINN training. The architecture of the PINN-based dynamic model is shown in Fig. 7.

4.2. Physics equations and modeling assumptions

The physics-informed component of the proposed PINN is based on the volume-balance equations governing the phase height dynamics in the gravity settler, as introduced in Section 3. In the present work, only these macroscopic balance equations are incorporated into the PINN as soft constraints, while the detailed droplet coalescence and sedimentation submodels are not embedded.

Based on the volume balance equations in Eq. (1), the physics constraints are formulated as residuals that penalize deviations from the underlying dynamics. Specifically, we define an algebraic residual (e_1) describing the overall flow balance of the separator, and differential residuals (e_2 and e_3) describing the phase height dynamics:

$$e_1(t) = g_{sep}\big(Q_{in}(t), \hat{Q}_{bot}(t), \hat{Q}_{top}(t)\big), \tag{4a}$$

$$e_2(t) = \dot{\hat{h}}_{DP}(t) - f_{DP}\big(\hat{h}_{DP}(t), \hat{h}_{HP}(t), Q_{in}(t), \hat{Q}_{bot}(t), \hat{Q}_c(t), \hat{Q}_s(t)\big), \tag{4b}$$

$$e_3(t) = \dot{\hat{h}}_{HP}(t) - f_{HP}\big(\hat{h}_{HP}(t), Q_{in}(t), \hat{Q}_{bot}(t), \hat{Q}_c(t), \hat{Q}_s(t)\big) \tag{4c}$$

Here, f_{DP} and f_{HP} denote the right-hand sides of the volume-balance equations defined in Section 3. The time derivatives of the predicted phase heights, $\dot{\hat{h}}$, are obtained via automatic differentiation of

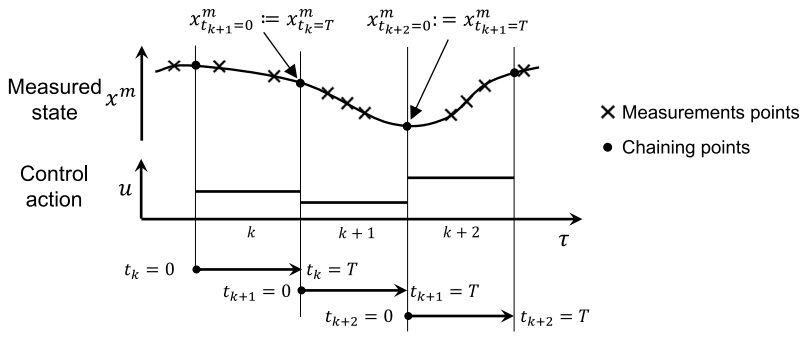


Fig. 6. Relationship between PINN time t and process time τ.
Source: Reproduced from [12].

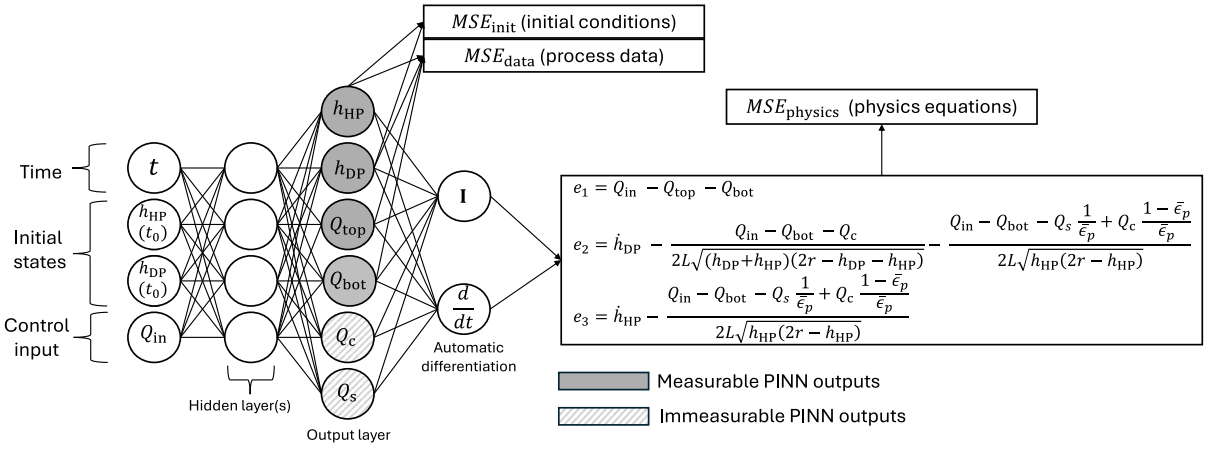


Fig. 7. Network schematic of the PINN model for liquid–liquid separator. Note that the figure does not show the actual depth and width of the hidden layers. Time dependence is omitted for better readability.

the neural network outputs with respect to the input time. These residuals define the physics-informed component of the training objective and penalize deviations from the governing equations.

Restricting the PINN to the volume-balance equations is mainly motivated by computational considerations. The semi-empirical submodels that define $Q_c(t)$ and $Q_s(t)$ as functions of the system states (cf. Eq. (2)) are lumped formulations obtained by resolving discretized internal compartments and subsequently aggregating their contributions. In particular, the separator is discretized along the axial direction into 200 segments and across 50 droplet diameter classes, resulting in approximately $200 \times 50 = 10{,}000$ internal compartments. Embedding these calculations into the PINN would either require introducing a tremendous number of additional outputs to represent the states of the discretized compartments or evaluating the submodels sequentially using PINN-predicted states during training. In both cases, this would lead to a prohibitive computational cost, as the full set of discretized compartments would need to be evaluated repeatedly at each training iteration. Moreover, the internal states associated with these submodels are not measurable in the experimental setup.

Instead of embedding the submodels for $Q_c(t)$ and $Q_s(t)$ into the PINN, the influence of these submodels is incorporated indirectly through the synthetic data used during pretraining, which is generated using the full mechanistic model. In this way, the PINN may learn an implicit representation of the underlying physics while maintaining a tractable model structure.

This modeling choice introduces a trade-off between physical completeness and computational efficiency. While the reduced physics representation may limit extrapolation in regimes dominated by droplet-scale phenomena, the combination of physics-informed training and experimental fine-tuning enables accurate phase height estimation in the operating conditions considered, as will be shown in Section 5.3.

4.3. Data handling for PINN training

Based on the preprocessed experimental dataset $\mathcal{D}_{exp}$ described in Section 2.4, phase height and flow-rate measurements are available at a uniform sampling interval of 1 s. This results in 789 segments (i.e., training samples), each corresponding to a single-step transition over 1 s. To align with the PINN formulation, which is defined over a temporal domain of $t \in [0, 1]$ s, the trajectories are partitioned into consecutive segments of duration 1 s. Each segment represents a single-step state transition from process time τ_k to $\tau_{k+1} = \tau_k + 1$ s and contains two measurement points for the phase heights, corresponding to the segment boundaries at $t = 0$ and $t = 1$.

For each segment, the PINN takes as input the initial states, the applied control input, and a specified value of the continuous time variable $t \in [0, 1]$, and returns the corresponding system states and outputs at that time. The explicit inclusion of time as an input enables the use of automatic differentiation to evaluate time derivatives of the network outputs. At the same time, this formulation provides a continuous-time representation of the system dynamics within each segment, allowing the PINN to generate predictions at arbitrary intermediate times.

An analogous procedure is applied to the simulation dataset $\mathcal{D}_{sim}$ generated from the mechanistic model described in Section 3. Specifically, we generate 1000 segments of duration 1 s, with inputs sampled within the bounds of the experimental data (cf. Table 3). Within each segment, the mechanistic model is evaluated at a finer temporal resolution of $\Delta t = 0.1$ s, resulting in additional training points that enhance the representation of the system dynamics over the interval.

In addition to simulation and experimental datasets, $\mathcal{D}_{\text{sim}}$ and $\mathcal{D}_{\text{exp}}$, respectively, we create two more datasets, $\mathcal{D}_{\text{physics}}$ and $\mathcal{D}_{\text{init}}$ for physics-based training and initial condition training, respectively. For the physics-based training, we sample $|\mathcal{D}_{\text{physics}}| = 10000$ collocation points corresponding to the PINN inputs: (i) time t, (ii) control Q_{in}, (iii) initial states $h_{\text{DP}}(0)$ and $h_{\text{HP}}(0)$. We use the extrapolation minimum and maximum bounds for controls and initial states given in Table 3, and sample using LHS [39]. Similarly, we choose $|\mathcal{D}_{\text{init}}| = 1000$ collocation points as initial conditions within the same bounds at $t = 0$. We use $\mathcal{D}_{\text{physics}}$ and the initial condition dataset $\mathcal{D}_{\text{init}}$ in both training stages (pretraining and fine-tuning). Note that, for stable training, we scale all the heights and the flow rates by division by the separator height $h_{\text{sep}} = 0.2\,\text{m}$ and by a prescribed flow rate $Q_{\text{scale}} = 0.001\ \text{m}^3/s$ respectively.

It is important to note that, while the PINN yields continuous-time predictions, the experimental data provide measurements only at the discrete sampling instants corresponding to the segment boundaries. Intermediate predictions within each segment are therefore not directly supervised by experimental data, but are supported during training by both the pretraining dataset $\mathcal{D}_{\text{sim}}$, which provides more densely sampled data, and the collocation points in $\mathcal{D}_{\text{physics}}$. The validation and test trajectories are not included in $\mathcal{D}_{\text{exp}}$ and are used exclusively for model evaluation.

4.4. Training strategy

Due to the considerable noise present in both the training and testing experimental trajectories, as well as the limited amount of data available, we use a two-stage training process, similar to [13]. First, the PINN is pretrained with the synthetic dataset $\mathcal{D}_{\text{sim}}$ generated by the mechanistic model in Section 3. The pretraining stage establishes baseline network weights and aligns the PINN with the dynamics of the mechanistic model before scarce and noisy experimental data is introduced.

In the second stage, we fine-tune the pretrained models using data from the experimental setup, $\mathcal{D}_{\text{exp}}$. The experimental dataset used for fine-tuning includes measurements of both phase heights and flow rates. To prevent the model from forgetting what it has learned during pretraining, we reduce the learning rate and the number of training epochs. The overall training strategy is depicted in Fig. 8.

Note that the immeasurable internal flows $\mathbf{z}(t) = [Q_{\text{c}}(t), Q_{\text{s}}(t)]^\top$ are outputs of the mechanistic model. We therefore use these quantities as training data during the pretraining stage. During fine-tuning, however, their corresponding loss terms are disabled, as these internal flows are not measurable and thus no experimental data are available for them.

The PINN parameters $\boldsymbol{\theta}$ can be learned by minimizing the mean squared error (MSE) loss, similar to [11] and [12]:

$$MSE_{total} = \lambda_{\text{data}} MSE_{\text{data}} + \lambda_{\text{latent}} MSE_{\text{latent}} + \lambda_{\text{init}} MSE_{\text{init}} \tag{5a}$$

$$+ \lambda_{\text{diff}} MSE_{\text{diff}} + \lambda_{\text{alg}} MSE_{\text{alg}}, \tag{5b}$$

$$MSE_{\text{data}} = \frac{1}{4|\mathcal{D}_{\text{data}}|} \sum_{j=1}^{|\mathcal{D}_{\text{data}}|} \Big[\left(\hat{h}_{\text{DP}}(t_j) - h_{\text{DP}}(t_j)\right)^2 + \left(\hat{h}_{\text{HP}}(t_j) - h_{\text{HP}}(t_j)\right)^2$$

$$+ \left(\hat{Q}_{\text{bot}}(t_j) - Q_{\text{bot}}(t_j)\right)^2 + \left(\hat{Q}_{\text{top}}(t_j) - Q_{\text{top}}(t_j)\right)^2 \Big] \tag{5c}$$

$$MSE_{\text{latent}} = \frac{1}{2|\mathcal{D}_{\text{data}}|} \sum_{j=1}^{|\mathcal{D}_{\text{latent}}|} \Big[\left(\hat{Q}_{\text{c}}(t_j) - Q_{\text{c}}(t_j)\right)^2 + \left(\hat{Q}_{\text{s}}(t_j) - Q_{\text{s}}(t_j)\right)^2 \Big] \tag{5d}$$

$$MSE_{\text{init}} = \frac{1}{2|\mathcal{D}_{\text{init}}|} \sum_{j=1}^{|\mathcal{D}_{\text{init}}|} \Big[\left(\hat{h}_{\text{DP}_j}(t=0) - h_{\text{DP}_j}(0)\right)^2 - \left(\hat{h}_{\text{HP}_j}(t=0) - h_{\text{HP}_j}(0)\right)^2 \Big] \tag{5e}$$

$$MSE_{\text{diff}} = \frac{1}{2|\mathcal{D}_{\text{physics}}|} \sum_{j=1}^{|\mathcal{D}_{\text{physics}}|} \Big[\left(e_2(t_j)\right)^2 + \left(e_3(t_j)\right)^2 \Big] \tag{5f}$$

$$MSE_{\text{algebraic}} = \frac{1}{|\mathcal{D}_{\text{physics}}|} \sum_{j=1}^{|\mathcal{D}_{\text{physics}}|} \left(e_1(t_j)\right)^2 \tag{5g}$$

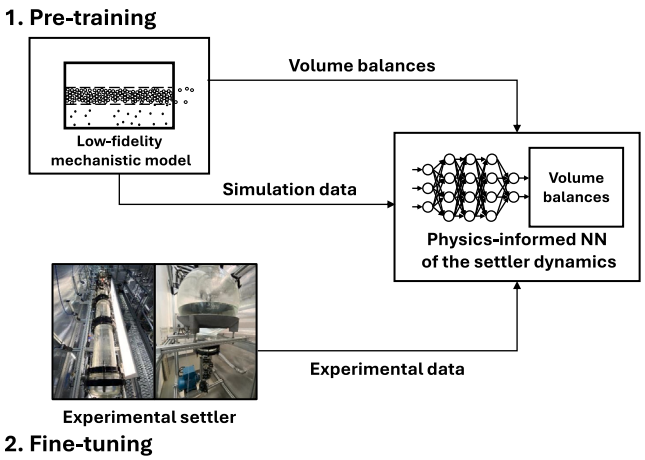
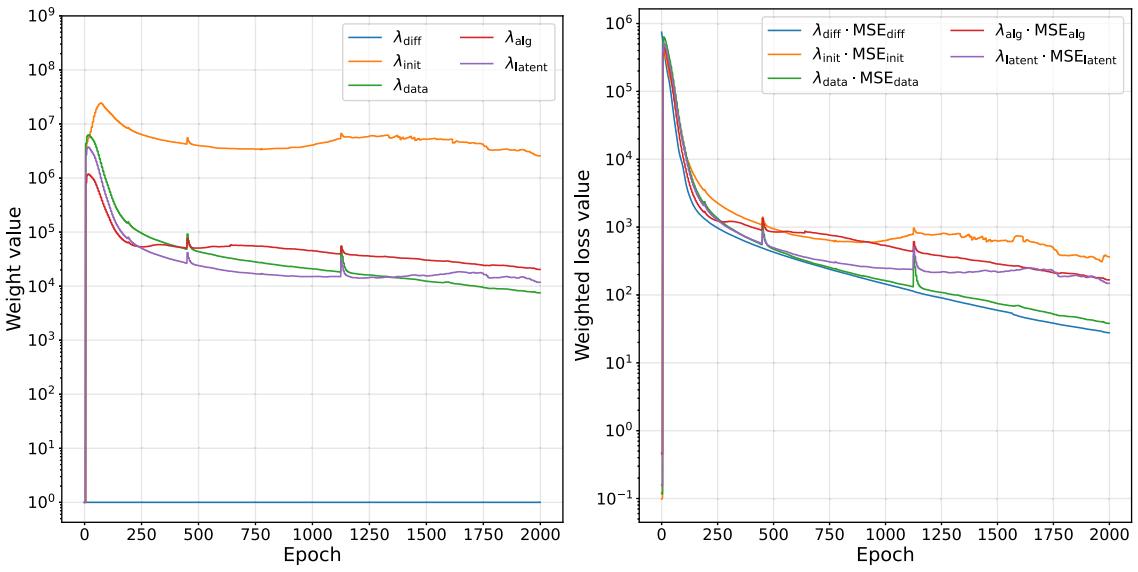


Fig. 8. Two-stage training strategy of PINN-based liquid–liquid separator model.

Here, MSE_{data} denotes the loss term associated with measurement data for phase heights and outlet flow rates, while MSE_{latent} corresponds to the loss for the latent internal flow rates. MSE_{init} quantifies the mismatch between the neural network predictions at $t = 0$ and the prescribed initial values $h_{\text{HP}_j}(0)$ and $h_{\text{DP}_j}(0)$. The terms MSE_{diff} and MSE_{alg} represent the physics-based loss contributions derived from the differential and algebraic residuals defined in Eq. (4). The subscript j indexes a finite set of samples taken at times t_j, each associated with initial values $h_{\text{HP}_j}(0)$ and $h_{\text{DP}_j}(0)$ and control action $Q_{in,j}$; for simplicity, initial values and control action are omitted from the notation. λ_{data}, λ_{latent}, λ_{diff}, λ_{alg}, and λ_{init} denote the weights assigned to the data, latent, differential physics, algebraic physics, and initial condition loss terms, respectively. $\mathcal{D}_{\text{data}}$ refers to the dataset consisting of measurements; specifically, we use $\mathcal{D}_{\text{sim}}$ in the pretraining stage and $\mathcal{D}_{\text{exp}}$ in the fine-tuning stage. The simulation dataset $\mathcal{D}_{\text{sim}}$ contains the internal flows $Q_{\text{c}}(t)$ and $Q_{\text{s}}(t)$. These quantities are absent from the experimental dataset $\mathcal{D}_{\text{exp}}$; consequently, fine-tuning is performed with the corresponding loss weight for the latent internal flows set to zero ($\lambda_{\text{latent}} = 0$) . Note that the samples used to compute MSE_{data} and MSE_{latent} are drawn from the same dataset $\mathcal{D}_{\text{data}}$, and similarly the samples used for MSE_{diff} and MSE_{alg} are both drawn from $\mathcal{D}_{\text{physics}}$.

We employ a two-optimizer training scheme for the PINN in each training stage. We start the training with a stochastic gradient descent (SGD) based optimizer (ADAM) [41], then switch to Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) [42] for refinement. L-BFGS has been shown to be a very successful optimizer for training PINNs [43], however, since it is a local optimizer, it has a tendency to get stuck in local minima [43]. Du et al. [44] have shown that SGD, due to its stochastic directions taken in optimization landscape can lead to (near) global training. Thus, we first use SGD to mitigate getting stuck in a local minimum, then we use L-BFGS to refine the network weights. In the pretraining stage, we use a fixed number of training epochs for each optimizer, 2000 for ADAM with a fixed learning rate of 0.001 and 300 for L-BFGS with the strong-Wolfe condition check activated. Moreover, we apply inverse Dirichlet weighting (IDW) [45] every five epochs during the ADAM optimization stage to dynamically update the individual loss term weights, λ_{data}, λ_{diff}, λ_{init}, λ_{alg} and λ_{latent} in Eq. . We note that IDW is adopted here as a practical training strategy based on preliminary empirical observations of improved stability and performance compared to a static weighting, consistent with its use in our previous work [12]; a systematic quantitative evaluation of alternative weighting schemes is beyond the scope of this study. We report the evolution of the adaptive loss weights (IDW) and the corresponding weighted loss terms during pre-training



(a) IDW weights (b) Weighted loss terms

Fig. 9. Evolution of adaptive loss weights (IDW) and corresponding weighted loss terms during pre-training, averaged over the ensemble. During fine-tuning, the weights are fixed to the final values from the pre-training.

in Fig. 9. Although the weights span several orders of magnitude, the weighted loss contributions are of comparable magnitude. In the fine-tuning stage, we reduce the learning rate by an order of magnitude to 0.0001 and lower the number of training epochs to 1000 for the ADAM optimizer and 200 for the L-BFGS optimizer. Moreover, we disable the IDW as the weightings of the different loss components are kept identical to the final values established in pretraining.

4.5. Benchmark vanilla neural network model

As a comparison for the PINN, we implement a benchmark *vanilla neural network* (VNN) model. The VNN has the same input configuration as the PINN: it takes time t, the current heights $\mathbf{x}(t = 0) = [h_{DP}(0), h_{HP}(0)]^\top$ and control inputs $\mathbf{u} = [Q_{in}]$ as inputs and predicts the corresponding measurable heights $h_{DP}(t)$, $h_{HP}(t)$ and outlet flows $Q_{bot}(t)$, $Q_{top}(t)$. Note that the VNN cannot predict the immeasurable internal flows $Q_c(t)$ and $Q_s(t)$, since no experimental data can be gathered for those. The VNN is trained in a data-driven fashion, i.e., without the physics-based loss terms in Eqs. (5f) and (5g). For comparison with the PINN model, we adopt the same network architecture as used for the PINN in Section 4.4, i.e, the same number of layers and neurons and the same activation functions. Moreover, we use the same datasets, except for, $\mathcal{D}_{physics}$, which is used exclusively in physics-based training.

4.6. Prediction results

We analyze how the models predict the experimental trajectories of phase heights when initialized from a *known state* at $\tau = 0$. We compare the accuracies of the PINN, VNN, and mechanistic model predictions, and examine the effects of pretraining. For both PINN and VNN, we employ an ensemble of models. Using an ensemble enables averaging over predictions to mitigate the impact of outlier models and provides a means to quantify prediction uncertainty [28]. The ensemble size for both PINN and VNN is fixed at $n_{ensemble} = 40$. To construct the ensemble, we train $n_{ensemble} = 40$ models independently with identical architectures and training data, but different random initialization of the network parameters. Due to the nonconvex loss landscape and stochasticity of the training algorithm, this may result in different trained parameter configurations and, consequently, a distribution of predictions. The ensemble spread is interpreted as an approximation of epistemic uncertainty arising from limited data and training variability.

We report prediction results in Figs. 10 and 11. To quantitatively assess model performance, we report the root mean square error (RMSE) of the ensemble-mean predictions over each trajectory, defined as

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2},$$

where $\hat{y}_i$ and y_i denote the predicted and reference values, respectively, and N is the number of samples. In addition, to quantify prediction uncertainty, we report the time-averaged interquartile range (IQR) [46] of the ensemble predictions over each trajectory, defined as

$$\mathrm{IQR} = q_{0.75} - q_{0.25},$$

where $q_{0.25}$ and $q_{0.75}$ denote the 25th and 75th percentiles of the ensemble predictions at a given time instant, respectively. The IQR is used instead of the standard deviation to reduce sensitivity to a small number of outlier models [46].

Since the PINN and VNN models take time $t \in [0, 1]$s, initial states, and control inputs as input and predict the heights and outlet flows at time t, trajectories over longer horizons must be constructed by chaining, i.e., the predicted heights is passed forward as the initial condition for the subsequent segment. Control inputs are known and provided for each segment of the trajectory.

In Figs. 10(a) and 10(b), we present the results for the interpolation trajectory for the heavy-phase height and the DPZ height, respectively. While the two-stage trained PINN ensemble tracks the averaged heights with good accuracy, the VNN ensemble mean overshoots with regard to the DPZ height. For both model types, the non-pretrained versions yield noticeably poorer predictions, with the effect being especially pronounced for the VNN. Moreover, we see that the mechanistic model predictions diverge, predicting flooding of the separator with the dispersed-phase at around $\tau = 380$ s.

In Figs. 11(a) and 11(b), we present the extrapolation trajectory results. The overall trends are similar to those observed for the interpolation trajectories: The two-stage trained PINN ensemble provides a fairly accurate tracking (albeit a slight overshoot towards the end), the VNN and the non-pretrained models perform significantly worse and the mechanistic model predicts flooding at around $\tau = 420$ s.

These results demonstrate the effectiveness of the two-stage training procedure and physics-based regularization. Moreover, while neither the mechanistic model nor the PINNs solely trained on experimental data can produce high-quality predictions, the combination of the two (through pretraining and fine-tuning) enables the PINN ensemble to track the trajectories quite accurately.

5. Phase height estimation

We now apply the models to estimate the phase heights in the gravity settler. It is important to distinguish between training and deployment: While phase height measurements are used during training, they will not be available during deployment. In Section 5.1 we present the state estimation framework that is inspired by the EKF to track and estimate the settler heights. Next, we describe the simple neural network to predict the DPZ height at separator outlet in Section 5.2. Finally, in Section 5.3, we discuss the estimation results, comparing PINN and VNN.

5.1. State estimation framework

The EKF is the nonlinear version of the Kalman filter [47], with the following state transition and measurement models in the discrete form [20]:

$$\mathbf{x}_k = \boldsymbol{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}, \tag{6}$$

$$\mathbf{y}_k = \boldsymbol{h}(\mathbf{x}_k) + \mathbf{v}_{k-1} \tag{7}$$

Here, subscript k denotes the time step, $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are the state-transition and measurement models, respectively, and $\mathbf{w}$ and $\mathbf{v}$ are process and measurement noise, respectively, that are assumed to be random Gaussian processes with zero mean and covariance $\mathbf{W}_k$ and $\mathbf{R}_k$ respectively [20]. Note that, since $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are nonlinear, they cannot be directly applied to the filter but have to be linearized around the current estimate, with Jacobians $\mathbf{F} = \left.\frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{k-1}}$ and $\mathbf{H} = \left.\frac{\partial \boldsymbol{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}_k}$. The Jacobians of the neural networks (PINN and VNN) can be readily obtained by automatic differentiation [21] of network outputs w.r.t. the network inputs [19].

We implement an EKF-inspired filter using the PINN as follows. We take subsets of the PINN outputs that are estimated and measured, i.e.,

$$\mathbf{x} = [h_{\mathrm{HP}}, h_{\mathrm{DP}}]^\top,\ \mathbf{y} = [Q_{\mathrm{bot}}, Q_{\mathrm{top}}]^\top,$$

respectively, where h_{HP} and h_{DP} are the states to be estimated by the filter, and Q_{bot} and Q_{top} are the measurements. Note that the bottom outlet flow rate Q_{bot} is controlled via the valve position during the experimental campaign and therefore constitutes an input to the physical system, directly influencing the internal phase heights. Despite this causal direction, an inverse mapping from the phase heights to the outflows is learned by the PINN, which will constitute the measurement model of the filter. This does not imply that the outlet flows are treated as states of the physical system; rather, they are used as signals whose dependence on the internal states is captured implicitly by the PINN.

The PINN is trained on a time domain $t \in [0, 1]$ s, matching the process data sampling interval of length $\Delta\tau = 1$ s. Let the PINN be defined as

$$\mathrm{PINN}_\theta(\cdot):\ \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times [0,1] \to \mathbb{R}^{n_x+n_y+n_z},$$

which maps the current estimated states $\mathbf{x}_{k-1}$, control input $\mathbf{u}$, and time $t \in [0, 1]$ to a joint prediction vector of (next) estimated states $\mathbf{x}_k$, measurements $\mathbf{y}$ and immeasurable internal flows $\mathbf{z}$. We then partition (by notation) the PINN into two functions, i.e.,

$$\mathrm{PINN}_\theta(\hat{\mathbf{x}}, \mathbf{u}, t) = \begin{bmatrix} \boldsymbol{f}_\theta(\hat{\mathbf{x}}, \mathbf{u}, t) \\ \boldsymbol{h}_\theta(\hat{\mathbf{x}}, \mathbf{u}, t) \end{bmatrix},$$

where $\boldsymbol{f}_\theta(\cdot):\ \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times [0,1] \to \mathbb{R}^{n_x}$ and $\boldsymbol{h}_\theta(\cdot):\ \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times [0,1] \to \mathbb{R}^{n_y}$ map the current estimated states, control input, and time $t \in [0, 1]$ into predicted estimated states and predicted measurements, respectively. The outputs associated with the immeasurable internal flows $\mathbf{z} = [Q_c(t), Q_s(t)]^\top$ are omitted from the filter, since our PINN architecture does not allow computing Jacobians w.r.t. to these flows. We then define the *state transition model* and *the measurement model* by evaluating the PINN at $t = 1$ and $t = 0$ respectively:

$$\text{State transition model:}\quad \hat{\mathbf{x}}_{k|k-1} = \boldsymbol{f}_\theta(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, t = 1) \tag{8}$$

$$\text{Measurement model:}\quad \hat{\mathbf{y}}_k = \boldsymbol{h}_\theta(\hat{\mathbf{x}}_{k|k-1}, \mathbf{u}_{k-1}, t = 0) \tag{9}$$

where the notation $\hat{\mathbf{x}}_{k|k-1}$ denotes the predicted state at time step k based on information up to time step $k-1$.

The prediction step [20] is then used to obtain the predicted state and covariance estimates:

$$\hat{\mathbf{x}}_{k|k-1} = \boldsymbol{f}_\theta(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, t = 1), \tag{10}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{W}_k, \tag{11}$$

where $\mathbf{F}_k = \left.\frac{\partial \boldsymbol{f}_\theta}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1|k-1}}$ is the Jacobian of the state transition model $\boldsymbol{f}_\theta(\cdot)$ with respect to the estimated states, $\mathbf{P}_k$ is the state covariance matrix, and $\mathbf{W}_k$ is the process noise covariance at time step k. Here, we initialize $\mathbf{P}_0$ as a diagonal matrix with values 0.0001 and 0.0001, corresponding to $\mathbf{x} = [h_{\mathrm{HP}}, h_{\mathrm{DP}}]^\top$.

The process noise covariance $\mathbf{W}_k$ is computed adaptively at each time step. Such adaptive sampling of the covariance matrix $\mathbf{W}_k$, sometimes referred to as Adaptive Kalman Filtering [48], is a well-established technique that has been shown to improve both the stability and performance of the Kalman Filter [48,49]. We leverage the model ensemble of 40 independently trained PINNs to calculate the process noise covariance $\mathbf{W}_k$ in the following way: (i) First, we take the ensemble mean of the last estimated state $\hat{\mathbf{x}}_{k-1|k-1}$ and feed it as the initial state to the transition model $\boldsymbol{f}_\theta(\cdot)$ of each ensemble member individually. (ii) We then obtain the next state estimate $\hat{\mathbf{x}}_{k|k-1}$ from each ensemble member. (iii) We calculate the sample covariance of these 40 predictions to obtain the process noise covariance $\mathbf{W}_k$ at each time step k. Here, the ensemble spread is interpreted as a sample-based approximation of the prediction uncertainty of the PINN model. The sample covariance of the ensemble predictions is used as a proxy for the process noise covariance $\mathbf{W}_k$, thereby incorporating model (epistemic) uncertainty into the filtering procedure. This allows the filter to account for increased uncertainty in regions where the model predictions diverge.

Next, we use the measurement model $\boldsymbol{h}_\theta(\cdot)$ that takes the last estimated states as initial state and outputs corresponding (predicted) measurements:

$$\hat{\mathbf{y}}_k = \boldsymbol{h}_\theta(\hat{\mathbf{x}}_{k|k-1}, \mathbf{u}_{k-1}, t = 0) \tag{12}$$

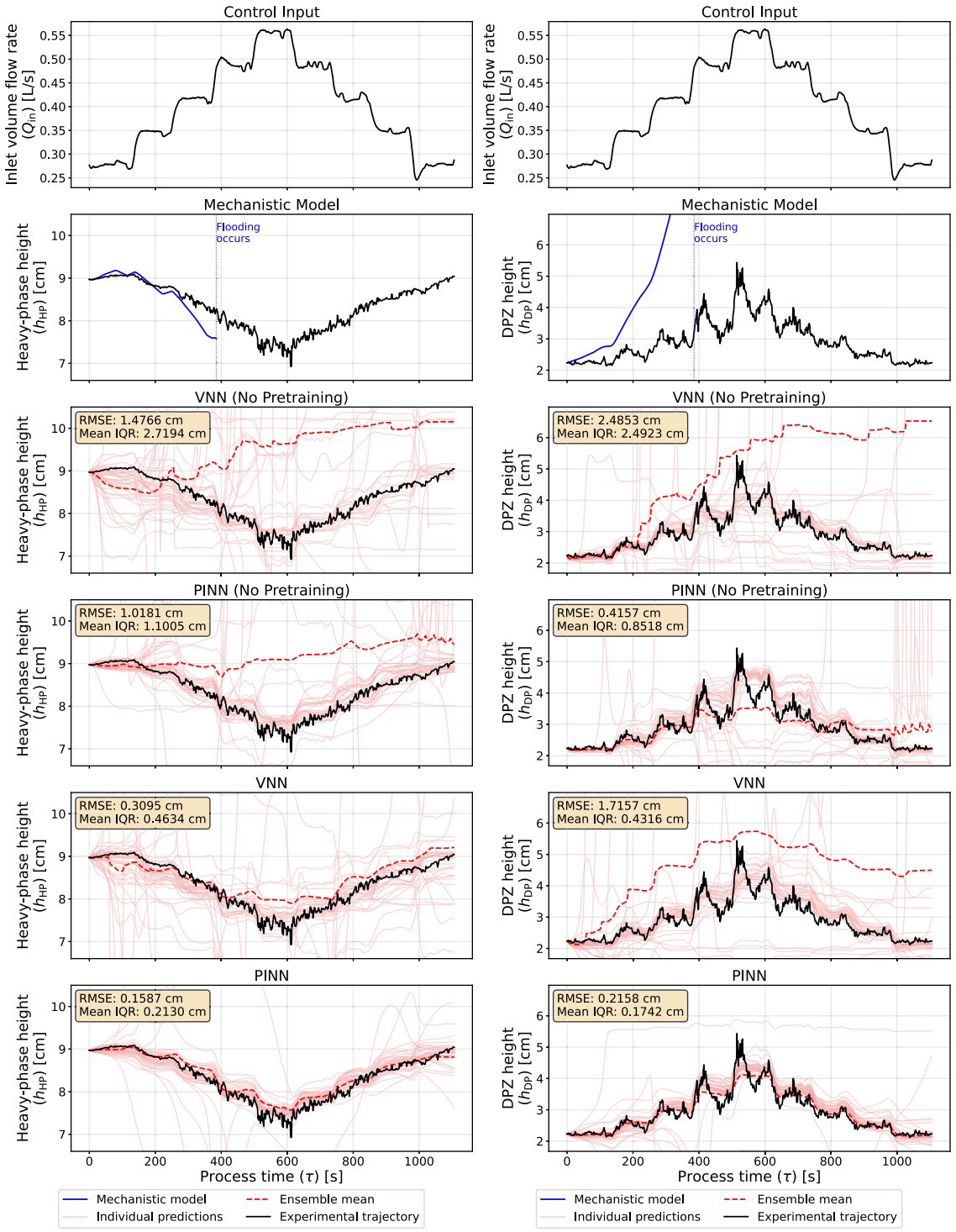
The update equations are then used to update the state and covariance estimates using the residual of true and predicted measurements and the Kalman gain $\mathbf{K}_k$ [20]:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \left(\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1}, \tag{13}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left(\mathbf{y}_k - \hat{\mathbf{y}}_k\right), \tag{14}$$

$$\mathbf{P}_{k|k} = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_{k|k-1} \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T, \tag{15}$$

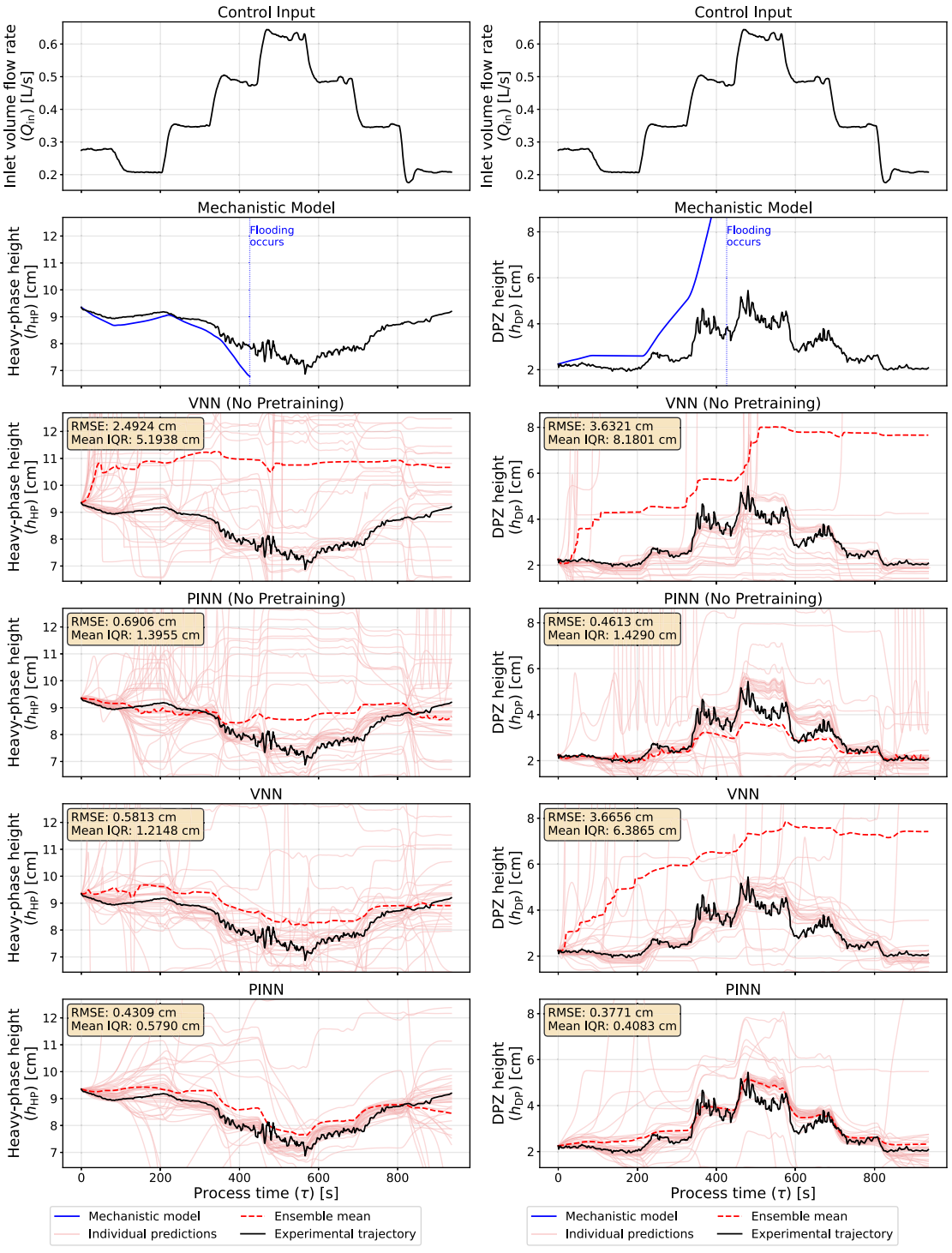
where $\mathbf{H}_k = \left.\frac{\partial \boldsymbol{h}_\theta}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}}$ is the Jacobian of the measurement model $\boldsymbol{h}_\theta(\cdot)$ w.r.t to the estimated states. The notation $\hat{\mathbf{x}}_{k|k}$ represents the updated state at time step k after incorporating the measurement at that step. The measurement noise covariance matrix $\mathbf{R}$ is diagonal and constant, with entries derived from the flow sensor specifications. Since both flow measurements come from the same sensor, a single $\mathbf{R}_k$ value is used for both states. The sensor specifies a maximum error of $e_{\max} = \pm 0.15\%$. Assuming a Gaussian error distribution, the standard deviation is approximated using the empirical 3σ heuristic as $\sigma = e_{\max}/3 = 5 \times 10^{-4}$ [50]. The resulting covariance is $\mathbf{R}_{\mathrm{sensor}} = \sigma^2 = 2.5 \times 10^{-7}$. Note that we use the Joseph form [51] for the covariance update in



(a) Heavy-phase (water) height **(b) Dense-packed zone** height

Fig. 10. Simulation results for the **interpolation test trajectory**. Predictions from an individual model are shown in transparent red, whereas the ensemble mean estimation is shown with a red dashed line. The reported root mean square error (RMSE) values quantify the deviation between the ensemble mean prediction of each model and the corresponding experimental trajectory over the entire trajectory. Time-averaged IQR values are reported to quantify the ensemble spread.

Eq. (15), for improved stability. Preliminary analysis of the validation trajectory showed that the average ratio of $\frac{\mathbf{W}_k}{\mathbf{R}_k}$ was high, causing the filter to overtrust measurement updates and yield noisy state estimates. To address this, $\mathbf{W}_k$ was decreased by a factor of 100. After each update, the PINN initial state is reset to $\hat{\mathbf{x}}_{k|k}$ to ensure that the state transition model starts from the most recent state estimate. We apply the filter for each model in the ensemble separately, and we only use the ensemble mean of state estimates for the process noise covariance calculations.

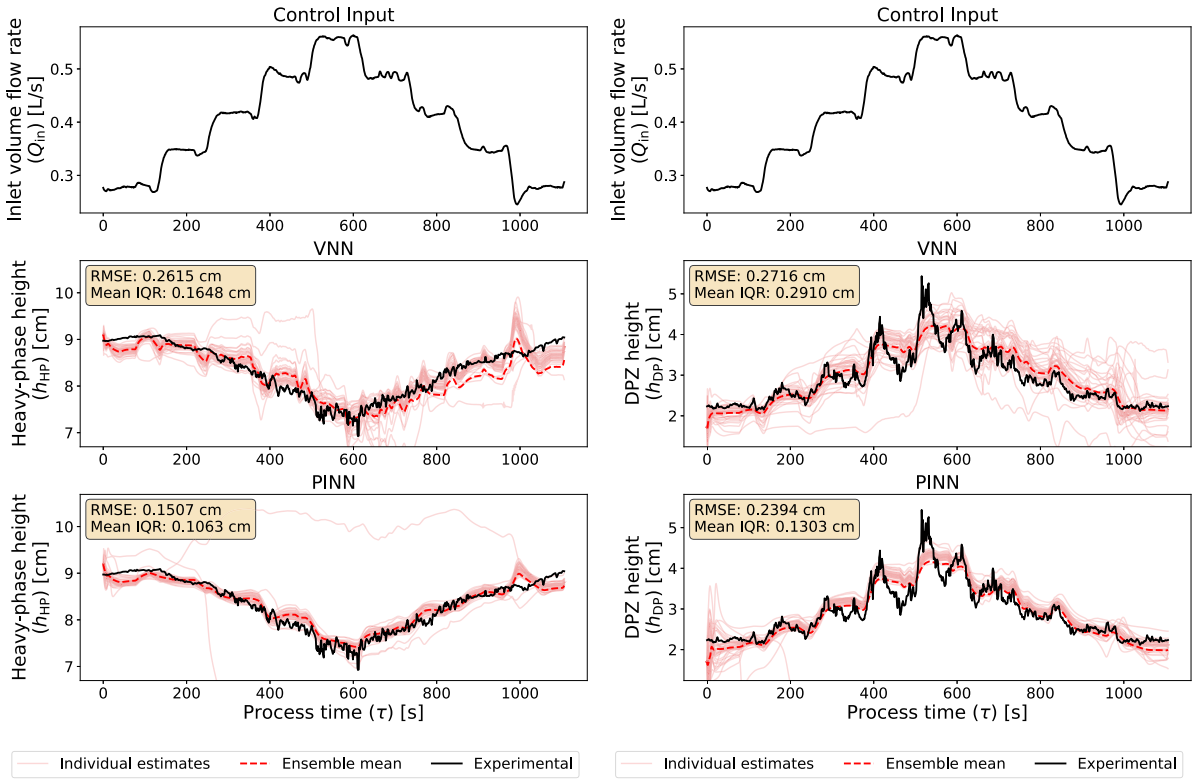


(a) Heavy-phase (water) height **(b) Dense-packed zone** height

Fig. 11. Simulation results for the **extrapolation test trajectory**. Predictions from an individual model are shown in transparent red, whereas the ensemble mean estimation is shown with a red dashed line. The reported RMSE values quantify the deviation between the ensemble mean prediction of each model and the corresponding experimental trajectory over the entire trajectory. Time-averaged IQR values are reported to quantify the ensemble spread.

5.2. Prediction of DPZ height at the separator outlet

In the wedge-shaped DPZ regime for higher flow rates, the maximum DPZ height consistently occurs at the end of the separator (cf. $h_{4,3}$ in Fig. 3). In contrast, our models, assuming a band-shaped DPZ, predict the average DPZ height along the separator. Aiming at flooding detection, which requires monitoring the DPZ height close to the separator outlet, we construct a simple feedforward neural network (NN) that maps the average DPZ height to $h_{4,3}$. The network has two hidden layers with 16 and 8 nodes, and is trained in a purely data-driven fashion using data from the training trajectory. Training is performed with the Adam optimizer for up to 1000 epochs, with early stopping if the validation error (from the validation trajectory) does not improve



(a) Heavy-phase (water) height **(b) Dense-packed zone** height

Fig. 12. State estimation results for the **interpolation test trajectory**. Estimations from an individual model are shown in transparent red, whereas the ensemble mean estimation is shown with a red dashed line. The reported RMSE values quantify the deviation between the ensemble mean estimation of each model and the corresponding experimental trajectory over the entire trajectory. Time-averaged IQR values are reported to quantify the ensemble spread.

after 30 epochs. After estimating the average phase heights with the PINN–EKF and VNN–EKF, we use the simple feedforward NN to predict $h_{4,3}$.

5.3. Results

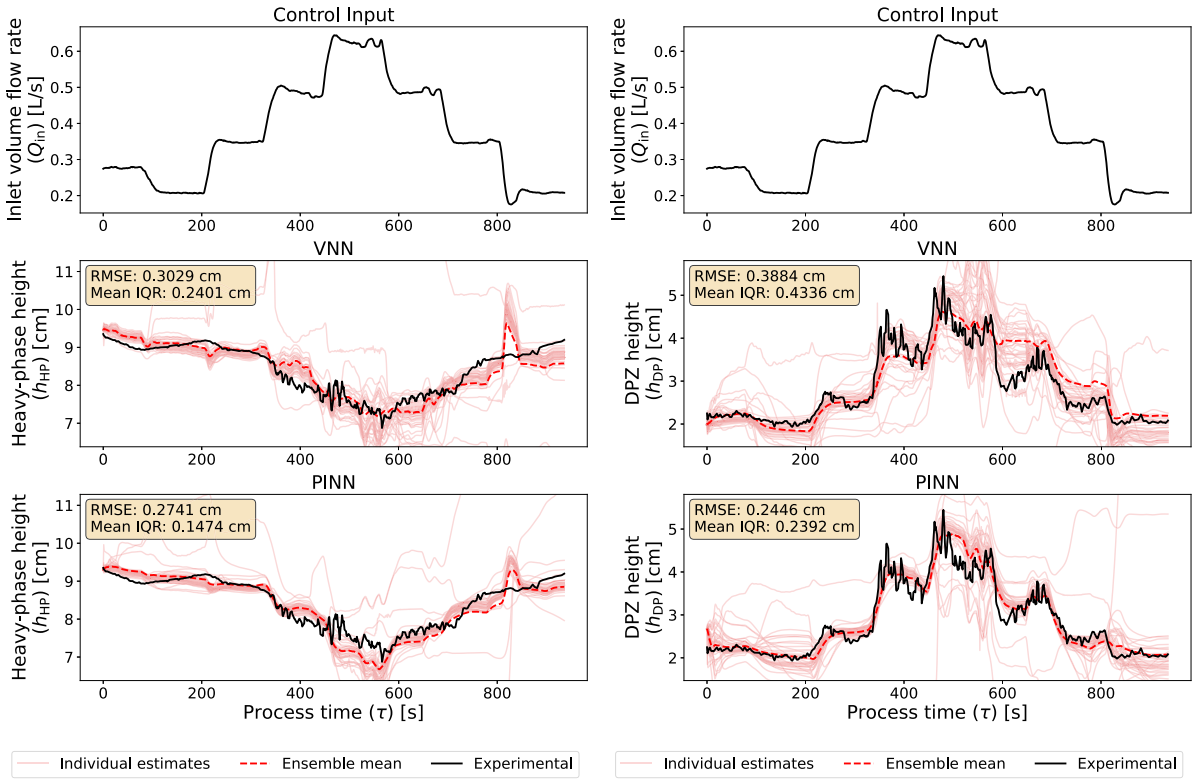
This section presents the results of the state estimation analysis. First, an initial guess for the phase heights is obtained by uniformly sampling 100 height vectors within their bounds and using those as initial state inputs to PINN (and VNN) to predict the outlet volume flows at $\tau = 0$. These predicted outflows are compared against the experimentally measured outlet flows at $\tau = 0$, and the candidate yielding the best agreement is selected as the initial state estimate. We then feed these initial states into the filter and proceed to track and estimate the phase heights for each PINN (and VNN) model individually. The individual trajectories, along with the ensemble means, are shown in Figs. 12 and 13. As in Section 5, model performance and uncertainty are quantified using RMSE and time-averaged IQR, respectively.

In Fig. 12, we report the results for the interpolation test trajectory. For both PINN and VNN, the estimated initial DPZ height $h_{DP}(\tau_0)$ deviates from the experimental trajectory, indicating weak reconstruction of the internal states from outlet flow measurements alone. Nevertheless, in both cases the ensemble mean converges to the experimental trajectory at around $\tau = 100\,\text{s}$. For the heavy phase height, the PINN consistently outperforms the VNN. Despite the significant overshoot observed in the forward simulation results shown in Fig. 10(b), the VNN–filter combination achieves comparable accuracy to the PINN ensemble in predicting DPZ height for the interpolation trajectory. This indicates that the filter update step effectively compensates for the VNN's error accumulation by incorporating outflow measurements. However, compared to the PINN, the individual estimates of the VNN ensemble exhibit lesser consistency, as shown by the lower IQR value of the PINN ensemble.

For the extrapolation test trajectory in Fig. 13, the PINN ensemble performs better than the VNN ensemble in case of the DPZ height. In particular, after the maximum control action (an operating value unseen in the training trajectory) is applied at around process time $\tau = 500\,\text{s}$, the VNN mean estimate overshoots the experimental DPZ height, whereas the PINN continues to track the DPZ height with good accuracy. Again, the individual estimates from the PINN ensemble are also more consistent than the ones from the VNN, as evidenced by the lower IQR value. For the heavy phase height, the VNN and PINN ensembles demonstrate similar accuracy, with the PINN underestimating the heavy phase height around process time $\tau = 500\,\text{s}$, while the VNN shows increasing deviation from the experimental trajectory thereafter.

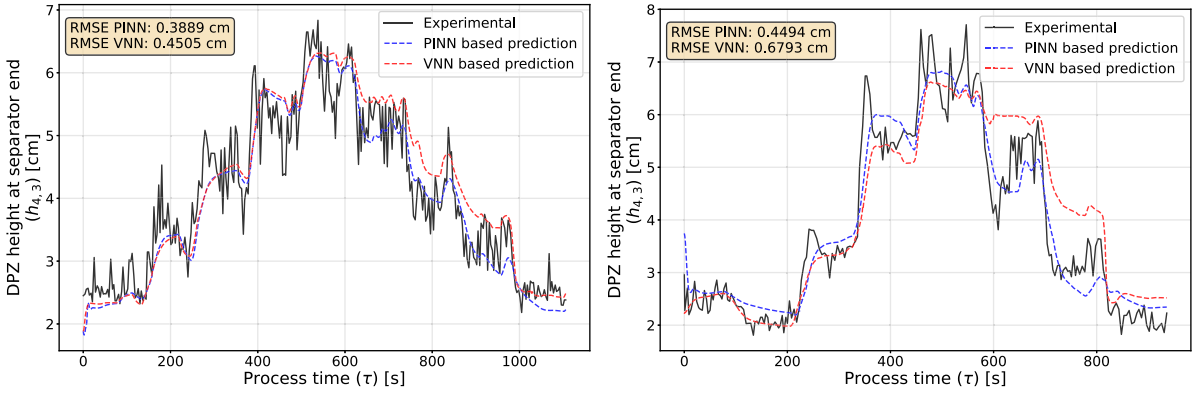
After estimating the average DPZ height along the separator using the filter, we apply the simple NN to predict the DPZ height at the separator end (Fig. 14). Both the PINN and VNN ensembles demonstrate good accuracy. Deviations occur primarily when the average DPZ height itself deviates, indicating that the NN performs well, but errors propagate where the underlying ensemble predictions are inaccurate (cf. Figs. 12 and 13). This effect is emphasized in the extrapolation trajectory (cf. Fig. 13), where larger deviations in the VNN ensemble lead to correspondingly larger errors in the predicted DPZ height at the separator end.

We therefore conclude that the two-stage trained PINN, when combined with the filter, enables effective real-time estimation of the internal separator states based on outlet flow measurements. As the outlet flow-rate measurements are inexpensive to obtain, our PINN constitutes a practical soft sensor for the difficult-to-measure phase heights. Moreover, we demonstrate that predicting the DPZ height at



(a) Heavy-phase (water) height **(b) Dense-packed zone** height

Fig. 13. State estimation results for the **extrapolation test trajectory**. Estimations from an individual model are shown in transparent red, whereas the ensemble mean estimation is shown with a red dashed line. The reported RMSE values quantify the deviation between the ensemble mean estimation of each model and the corresponding experimental trajectory over the entire trajectory. Time-averaged IQR values are reported to quantify the ensemble spread.



(a) Interpolation test trajectory **(b) Extrapolation** test trajectory

Fig. 14. Prediction of the DPZ height at the separator end. Experimental results are obtained from camera **QIR04** at detection window $h_{4,3}$. Estimates of the average DPZ from both PINN and VNN are fed into the NN that predicts the DPZ height at separator end. The reported RMSE values quantify the deviation between the prediction of each model and the corresponding experimental trajectory over the entire trajectory.

the separator end (the maximum value) from the average DPZ height is possible using a simple NN.

6. Conclusion and outlook

We investigated the estimation of phase heights in a pilot-scale liquid–liquid separator operating with 1-octanol and deionized water at 30 °C under varying flow conditions. Direct measurement of phase heights is challenging, as the interior of the separator is not readily accessible for reliable observation of the phase interface. Although we employ a YOLOv8-based phase-boundary detection algorithm in our experimental setup, the resulting measurements still contain noise due to the difficulty of distinguishing the interface and the sensitivity of the algorithm to illumination conditions. Moreover, the available mechanistic model relies on multiple simplifying assumptions, resulting in limited predictive accuracy. To address these challenges, we developed a PINN-based dynamic model and proposed a two-stage training strategy: pretraining on approximate physics, i.e., synthetic data from the low-fidelity model, followed by fine-tuning with experimental data. We further used ensemble learning to mitigate the influence of outlier

models. Moreover, the ensemble not only improves prediction robustness but also provides a practical measure of model uncertainty, which is incorporated into the state estimation framework through adaptive covariance estimation. Finally, we combined the trained PINN with an EKF-inspired filter to enable estimation of the aqueous phase and DPZ heights. The filter prevents error accumulation from chaining of PINN predictions by fusing model predictions with real-time flow measurements to update the estimated phase-height trajectories.

The two-stage PINN ensemble outperformed both the low-fidelity mechanistic model and a single-stage PINN ensemble trained only on experimental data, showing that integrating mechanistic model knowledge with experimental measurements yields better results than either approach alone. For further comparison, we also pretrained and fine-tuned an ensemble of purely data-driven vanilla neural networks (VNNs). Both the PINN and VNN ensembles were embedded in a state estimation framework to enable phase height estimation during deployment from flow-rate measurements only. The PINN ensemble achieved higher accuracy in tracking phase heights. Furthermore, a simple neural network can predict the outlet DPZ height (typically the maximum value and most critical for flooding) from average DPZ height estimates. The results demonstrate that the PINN-based state estimation can serve as a reliable soft sensor, enabling the real-time monitoring of this separation process from readily available flow-rate measurements.

A few limitations of the present study should be noted. First, the model assumes a band-shaped DPZ, which becomes inaccurate at higher flow rates where the DPZ develops a distinct wedge-shaped profile. Although a mapping between the average and maximum DPZ height was learned by a simple neural network, the PINN model does not fully represent the actual separator dynamics. Second, the experimental data contain substantial gaps caused by missing phase-boundary detections, particularly for QIR03, which limits the training data quality. Third, the experimental dataset includes no trajectories with conditions close to flooding. While such conditions are difficult to obtain experimentally for safety reasons, their absence restricts the ability of the model to detect or classify flooding behavior. Fourth, we acknowledge that filtering the immeasurable internal flows in the proposed state estimation framework could improve phase height estimation accuracy, however this is not possible with the current PINN architecture. Lastly, the experimental dataset comprises only four trajectories, with only one being used for training.

These limitations outline clear directions for future work. Extending the framework to handle non-uniform DPZ geometries would improve predictive capability under a wider range of operating conditions. Experimental data quality for phase heights may be enhanced through improved illumination, refinements to the phase-boundary detection framework, and the use of larger training datasets. Entrained droplets in the outlets relevant to quantifying incomplete separation can also be investigated. Finally, near-flooding and flooding data would allow more rigorous evaluation of the estimation performance under safety-relevant operating conditions.

Nomenclature

Abbreviations

DPZ	dense-packed zone
EKF	extended Kalman Filter
IDW	inverse Dirichlet weighting
IQR	interquartile range
L-BFGS	limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm
LHS	Latin hypercube sampling
LLS	liquid–liquid separator
MSE	mean squared error
NN	neural network
ODE	ordinary differential equation
PDE	partial differential equation
PINN	physics-informed neural network
RMSE	root mean square error
SGD	stochastic gradient descent
UKF	unscented Kalman Filter
VNN	vanilla neural network

Greek Symbols

γ	interfacial tension
ϵ	holdup
$\boldsymbol{\theta}$	learnable parameters
λ	PINN loss term weight
η	dynamic viscosity
ρ	density
$\Delta\rho$	density difference between organic and aqueous phase
σ	standard deviation
τ	process time

Latin Symbols

d	diameter
D	dataset
e	error
$\boldsymbol{f}$	differential equation right-hand side
$\mathbf{F}$	Jacobian of state transition model
$\boldsymbol{g}$	algebraic equation right-hand side
h	height
$\boldsymbol{h}$	measurement model
$\mathbf{H}$	Jacobian of measurement model
$\mathbf{I}$	identity matrix
$\mathbf{K}$	Kalman gain matrix
L	separator length
n	number
$\mathbf{P}$	state covariance matrix
PINN_θ	PINN functional representation
Q	volume flow rate
$\mathbf{W}$	process noise covariance matrix
r	separator radius
$\mathbf{R}$	measurement noise covariance matrix
t	PINN time
T	time duration
$\mathbf{u}$	control variables
$\mathbf{v}$	measurement noise
$\mathbf{w}$	process noise
$\mathbf{x}$	state variables
$\mathbf{y}$	system outputs
$\mathbf{z}$	immeasurable internal flows

Subscripts

0	initial state
32	Sauter mean
bot	bottom outflow
c	coalescence
data	measurement data
DP	dense-packed zone
ensemble	ensemble
exp	experimental
g	algebraic equation
HP	heavy phase
in	inlet
k	time step
max	maximum
out	output
physics	physics
init	initial condition
s	sedimentation

scale	scaling
selfsimilar	self-similarity
sensor	sensor
sep	separator
sim	simulation
swarm	swarm
θ	learnable parameters
top	top outflow
z	immeasurable internal flows

CRediT authorship contribution statement

Mehmet Velioglu: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Song Zhai:** Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Alexander Mitsos:** Conceptualization, Writing – review & editing, Supervision. **Adel Mhamdi:** Methodology, Writing – review & editing. **Andreas Jupke:** Conceptualization, Writing – review & editing, Supervision, Funding acquisition. **Manuel Dahmen:** Conceptualization, Methodology, Writing – review & editing, Supervision, Funding acquisition.

Declaration of Generative AI and AI-Assisted Technologies

During the preparation of this work, generative AI tools (ChatGPT, Claude) were used to assist with grammar, spelling, and style improvements in the writing. Generative AI coding tools (Claude) were used for post-processing tasks, specifically for plotting and visualization of the results. In all cases, the authors reviewed and edited the outputs, and they take full responsibility for the integrity and accuracy of the publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 466656378 – within the Priority Programme “SPP 2331:Machine Learning in Chemical Engineering”. This work was performed as part of the Helmholtz School for Data Science in Life, Earth and Energy (HDS-LEE). We acknowledge financial support by the Helmholtz Association of German Research Centers through program-oriented funding.

Data availability

Data will be made available on request.

References

[1] T. Frising, C. Noïk, C. Dalmazzone, The liquid/liquid sedimentation process: From droplet coalescence to technologically enhanced water/oil emulsion gravity separators: A review, J. Dispers. Sci. Technol. 27 (7) (2006) 1035–1057, http://dx.doi.org/10.1080/01932690600767098.

[2] M. Henschke, Determination of a coalescence parameter from batch-settling experiments, Chem. Eng. J. 85 (2–3) (2002) 369–378, http://dx.doi.org/10.1016/S1385-8947(01)00251-0.

[3] S. Zhai, N. Bartkowiak, S. Sibirtsev, A. Jupke, Experimental determination and model-based prediction of flooding points in a pilot-scale continuous liquid-liquid gravity separator, Sep. Purif. Technol. 377 (2025) 134177, http://dx.doi.org/10.1016/j.seppur.2025.134177.

[4] S. Ye, L. Hohl, M. Kraume, Impact of feeding conditions on continuous liquid-liquid gravity separation, part I: Inlet and outlet drop size, dense-packed zone and separation efficiency, Chem. Eng. Sci. 282 (2023) 119237, http://dx.doi.org/10.1016/j.ces.2023.119237.

[5] S. Ye, L. Hohl, M. Kraume, Impact of feeding conditions on continuous liquid-liquid gravity separation, part II: Inlet/outlet drop size distribution and fractional separation efficiency, Chem. Eng. Sci. 285 (2024) 119611, http://dx.doi.org/10.1016/j.ces.2023.119611.

[6] R. Cusack, Rethink your liquid–liquid separations: A fresh look investigates general principles in designing process coalescers, 2009, Hydrocarbon Processing, Special Report: Process & Plant Optimization, June 2009. URL https://www.hydrocarbonprocessing.com/magazine/2009/june-2009/special-report-processplant-optimization/rethink-your-liquid-liquid-separations/. (Accessed 23 September 2025).

[7] M. Henschke, Dimensionierung liegender Flüssig-flüssig-Abscheider anhand diskontinuierlicher Absetzversuche (Ph.D. thesis), Aachen Technische Hochschule, 1995, URL https://publications.rwth-aachen.de/record/57406.

[8] J. Kamp, J. Villwock, M. Kraume, Drop coalescence in technical liquid/liquid applications: a review on experimental techniques and modeling approaches, Rev. Chem. Eng. 33 (1) (2017) 1–47, http://dx.doi.org/10.1515/revce-2015-0071.

[9] C.J. Backi, B.A. Grimes, S. Skogestad, A control- and estimation-oriented gravity separator model for oil and gas applications based upon first-principles, Ind. Eng. Chem. Res. 57 (21) (2018) 7201–7217, http://dx.doi.org/10.1021/acs.iecr.7b04297.

[10] N. Shlezinger, J. Whang, Y.C. Eldar, A.G. Dimakis, Model-based deep learning, 2022, arXiv preprint arXiv:2012.08405.

[11] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045.

[12] M. Velioglu, S. Zhai, S. Rupprecht, A. Mitsos, A. Jupke, M. Dahmen, Physics-informed neural networks for dynamic process operations with limited physical knowledge and data, Comput. Chem. Eng. 192 (2025) 108899, http://dx.doi.org/10.1016/j.compchemeng.2024.108899.

[13] S. Chakraborty, Transfer learning based multi-fidelity physics informed deep neural network, J. Comput. Phys. 426 (2021) http://dx.doi.org/10.1016/j.jcp.2020.109942.

[14] Y. Wang, J. Bai, M.S. Eshaghi, C. Anitescu, X. Zhuang, T. Rabczuk, Y. Liu, Transfer learning in physics-informed neural networks: Full fine-tuning, lightweight fine-tuning, and low-rank adaptation, 2025, arXiv preprint arXiv:2502.00782.

[15] A.H. Mustajab, H. Lyu, Z. Rizvi, F. Wuttke, Physics-informed neural networks for high-frequency and multi-scale problems using transfer learning, 2024, arXiv preprint arXiv:2401.02810.

[16] K. Prantikos, S. Chatzidakis, L.H. Tsoukalas, A. Heifetz, Physics-informed neural network with transfer learning (TL-PINN) based on domain similarity measure for prediction of nuclear reactor transients, Sci. Rep. 13 (2023) http://dx.doi.org/10.1038/s41598-023-43325-1.

[17] Y. Wang, E.A. del Río Chanona, P. Quintanilla, Gaussian process nonlinear model predictive control for online partially observable systems: An application to froth flotation, Ind. Eng. Chem. Res. 64 (2025) 13307–13322, http://dx.doi.org/10.1021/acs.iecr.5c00660.

[18] W. Zhao, J.P. Queralta, T. Westerlund, Sim-to-real transfer in deep reinforcement learning for robotics: a survey, in: 2020 IEEE Symposium Series on Computational Intelligence, SSCI, 2020, pp. 737–744, http://dx.doi.org/10.1109/SSCI47803.2020.9308468.

[19] F. Arnold, R. King, State–space modeling for control based on physics-informed neural networks, Eng. Appl. Artif. Intell. 101 (2021) http://dx.doi.org/10.1016/j.engappai.2021.104195.

[20] A. Gelb, J.F. Kasper, R.A. Nash, C.F. Price, A.A. Sutherland (Eds.), Applied Optimal Estimation, MIT Press, Cambridge, MA, 1974.

[21] U. Naumann, The Art of Differentiating Computer Programs, Society for Industrial and Applied Mathematics, 2011, http://dx.doi.org/10.1137/1.9781611972078.

[22] S. Cuomo, M.D. Rosa, F. Piccialli, L. Pompameo, Railway safety through predictive vertical displacement analysis using the PINN-EKF synergy, Math. Comput. Simulation 223 (2024) 368–379, http://dx.doi.org/10.1016/j.matcom.2024.04.026.

[23] C. Tan, Y. Cai, H. Wang, X. Sun, L. Chen, Vehicle state estimation combining physics-informed neural network and unscented Kalman filtering on manifolds, Sensors 23 (2023) http://dx.doi.org/10.3390/s23156665.

[24] S. Julier, J. Uhlmann, Unscented filtering and nonlinear estimation, Proc. IEEE 92 (3) (2004) 401–422, http://dx.doi.org/10.1109/JPROC.2003.823141.

[25] J. de Curtò, I. de Zarzà, Hybrid state estimation: Integrating physics-informed neural networks with adaptive UKf for dynamic systems, Electronics 13 (2024) http://dx.doi.org/10.3390/electronics13112208.

[26] S. Fang, K. Yu, Fine-tuning hybrid dynamics with physics-informed neural networks for vehicle dynamics estimation, Int. J. Intell. Robot. Appl. (2025) http://dx.doi.org/10.1007/s41315-025-00452-4.

[27] L. Breiman, Stacked regressions, Mach. Learn. 24 (1) (1996) 49–64, http://dx.doi.org/10.1007/BF00117832.
[28] T.G. Dietterich, Ensemble methods in machine learning, in: International Workshop on Multiple Classifier Systems, Springer, 2000, pp. 1–15, http://dx.doi.org/10.1007/3-540-45014-9_1.
[29] S. Sibirtsev, S. Zhai, M. Neufang, J. Seiler, A. Jupke, Mask R-CNN based droplet detection in liquid–liquid systems, part 2: Methodology for determining training and image processing parameter values improving droplet detection accuracy, Chem. Eng. J. 473 (2023) 144826, http://dx.doi.org/10.1016/j.cej.2023.144826.
[30] S. Sibirtsev, L. Thiel, S. Zhai, Y.T. Cai, L. Recke, A. Jupke, Experimental and model–based investigation of the droplet size distribution during the mixing process in a batch–settling cell, Can. J. Chem. Eng. (2024) http://dx.doi.org/10.1002/cjce.25563.
[31] A. Palmtag, L. Lehmann, L.R. Hanz, U. Kiseleva, A. Jupke, Towards the digital extraction column: Online-monitoring and analysis of fluid dynamics in liquid-liquid extraction columns, Chem. Eng. J. Adv. 22 (2025) 100727, http://dx.doi.org/10.1016/j.ceja.2025.100727.
[32] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: 29th IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Piscataway, NJ, 2016, pp. 779–788, http://dx.doi.org/10.1109/CVPR.2016.91.
[33] R. Padilla, M. Ruiz, W. Trujillo, Separation of liquid-liquid dispersions in a deep-layer gravity settler: Part I. Experimental study of the separation process, Hydrometallurgy 42 (2) (1996) 267–279, http://dx.doi.org/10.1016/0304-386X(95)00095-X.
[34] S.A.K. Jeelani, S. Hartland, Dynamic response of gravity settlers to changes in dispersion throughput, AIChE J. 34 (2) (1988) 335–340, http://dx.doi.org/10.1002/aic.690340220.
[35] M. Kraume, A. Gäbler, K. Schulze, Influence of physical properties on drop size distribution of stirred liquid-liquid dispersions, Chem. Eng. Technol. 27 (3) (2004) 330–334, http://dx.doi.org/10.1002/ceat.200402006.
[36] S. Ye, L. Hohl, E. Charlafti, Z. Jin, M. Kraume, Effect of temperature on mixing and separation of stirred liquid/liquid dispersions over a wide range of dispersed phase fractions, Chem. Eng. Sci. 274 (2023) 118676, http://dx.doi.org/10.1016/j.ces.2023.118676.
[37] A. Mersmann, Zum Flutpunkt in Flüssig/Flüssig–Gegenstromkolonnen, Chem. Ing. Tech. 52 (12) (1980) 933–942, http://dx.doi.org/10.1002/cite.330521203.
[38] J. Kampwerth, B. Weber, J. Rußkamp, S. Kaminski, A. Jupke, Towards a holistic solvent screening: On the importance of fluid dynamics in a rate-based extraction model, Chem. Eng. Sci. 227 (2020) 115905, http://dx.doi.org/10.1016/j.ces.2020.115905.
[39] R.L. Iman, J.C. Helton, J.E. Campbell, An approach to sensitivity analysis of computer models: Part I—Introduction, input variable selection and preliminary variable assessment, J. Qual. Technol. 13 (3) (1981) 174–183, http://dx.doi.org/10.1080/00224065.1981.11978748.
[40] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Y.W. Teh, M. Titterington (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, vol. 9, PMLR, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256, URL https://proceedings.mlr.press/v9/glorot10a.html.
[41] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017, arXiv preprint arXiv:1412.6980.
[42] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. 45 (1) (1989) 503–528, http://dx.doi.org/10.1007/BF01589116.
[43] S. Markidis, The old and the new: Can physics-informed deep-learning replace traditional linear solvers? Front. Big Data 4 (2021) http://dx.doi.org/10.3389/fdata.2021.669097.
[44] S.S. Du, J.D. Lee, H. Li, L. Wang, X. Zhai, Gradient descent finds global minima of deep neural networks, 2019, arXiv preprint arXiv:1811.03804.
[45] S. Maddu, D. Sturm, C.L. Müller, I.F. Sbalzarini, Inverse Dirichlet weighting enables reliable training of physics informed neural networks, Mach. Learn.: Sci. Technol. 3 (1) (2022) 015026, http://dx.doi.org/10.1088/2632-2153/ac3712.
[46] F.M. Dekking, C. Kraaikamp, H.P. Lopuhaä, L.E. Meester, A Modern Introduction to Probability and Statistics: Understanding Why and How, Springer, London, 2005.
[47] R.E. Kalman, A new approach to linear filtering and prediction problems, J. Basic Eng. 82 (1) (1960) 35–45, http://dx.doi.org/10.1115/1.3662552.
[48] A. Mohamed, K. Schwarz, Adaptive Kalman filtering for INS/GPS, J. Geod. 73 (4) (1999) 193–203, http://dx.doi.org/10.1007/s001900050236.
[49] L. Zhang, D. Sidoti, A. Bienkowski, K.R. Pattipati, Y. Bar-Shalom, D.L. Kleinman, On the identification of noise covariances and adaptive Kalman filtering: A new look at a 50 year-old problem, IEEE Access 8 (2020) 59362–59388, http://dx.doi.org/10.1109/ACCESS.2020.2982407.
[50] D. Wackerly, W. Mendenhall, R.L. Scheaffer, Mathematical Statistics with Applications, Seventh ed., Brooks/Cole, 2008.
[51] R.S. Bucy, P.D. Joseph, Filtering for Stochastic Processes with Applications to Guidance, vol. 326, American Mathematical Soc., 2005, http://dx.doi.org/10.1109/TAC.1972.1099917.