Progress in Nuclear Energy 197 (2026) 106411

Contents lists available at ScienceDirect

Progress in Nuclear Energy

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

Node assigned physics-informed neural networks for thermal–hydraulic system simulation: CVH/FL/HS modules

Jeesuk Shin [a,1], Cheolwoong Kim [b,1], Sunwoong Yang [c,d], Minseo Lee [a], Donggyun Seo [e], Sung Joong Kim [b,*], Joongoo Jeon [a,*]

[a] Division of Advanced Nuclear Engineering, Pohang University of Science and Technology, Pohang-si, Republic of Korea
[b] Department of Nuclear Engineering, Hanyang University, Seoul, Republic of Korea
[c] Cho Chun Shik Graduate School of Mobility, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea
[d] Department of Mechanical Engineering, Hanyang University, Ansan, Republic of Korea
[e] Mueunjae School of Undergraduate Studies, Pohang University of Science and Technology, Pohang-si, Republic of Korea

ARTICLE INFO

Keywords:
FDM
PINN
Thermal-hydraulics
Control-volume approach

ABSTRACT

Severe accidents (SAs) in nuclear power plants have been analyzed using thermal–hydraulic (TH) system codes such as MELCOR and MAAP. These codes efficiently simulate the progression of SAs, while they still have inherent limitations due to their inconsistent finite difference schemes. The use of empirical schemes incorporating both implicit and explicit formulations inherently induces unidirectional coupling in multi-physics analyses. The objective of this study is to develop a novel numerical method for TH system codes using physics-informed neural network (PINN). They have shown strength in solving multi-physics due to the innate feature of neural networks—automatic differentiation. We propose a node-assigned PINN (NA-PINN) that is suitable for the control volume approach-based system codes. NA-PINN addresses the issue of spatial governing equation variation by assigning an individual network to each nodalization of the system code, such that spatial information is excluded from both the input and output domains, and each subnetwork learns to approximate a purely temporal solution. In this phase, we evaluated the accuracy of the PINN methods for the hydrodynamic module. In the 6 water tank simulation, PINN and NA-PINN showed maximum absolute errors of 1.908146 and 0.003757, respectively. It should be noted that only NA-PINN demonstrated acceptable accuracy. The numerical feasibility of NA-PINN was also verified for a longer-term heat transfer case study. To the best of the authors' knowledge, this is the first study to successfully implement a system code using PINN. Our future work involves extending NA-PINN to a multi-physics solver and developing it in a surrogate manner.

1. Introduction

Due to the extremely low frequency of severe accident (SA) in nuclear power plants (NPPs) and the limited availability of real-world accident data, SA-related research inevitably relies on the use of system codes to simulate hypothetical accident scenarios and assess the potential safety concerns. Widely used system codes, such as RELAP5/SCDAP, MAAP, and MELCOR, model the physical behavior of NPP components and simulate accident progression by accounting for complex thermal–hydraulic (TH) and physicochemical interactions arising under SA conditions. These simulations cover a broad spectrum of phenomena, including inter-component TH coupling and the release, transport, and deposition of fission products originating from fuel degradation during accident progression (International Atomic Energy Agency, 2002, 2007).

Such codes allow users to gain insights into accident phenomena either by analyzing simulation results or validating these results against data acquired from actual accident events (International Atomic Energy Agency, 2019). Each system code was developed by a different organization: RELAP5/SCDAPSIM by Innovative Systems Software (ISS) (U.S. Nuclear Regulatory Commission, 2001), MAAP by the Electric Power Research Institute (EPRI) (Electric Power Research Institute (EPRI), 2014), and MELCOR by Sandia National Laboratories (SNL) (Gauntt et al., 2021). For cross-validation purposes, numerous studies have been conducted to compare the outputs of these system codes (Sandia National Laboratories, 2021, 2019; Song et al., 2023).

Among the available system codes, MELCOR was selected for this study owing to its flexibility in components customization and its suitability for modeling simplified severe accident scenarios (Gauntt

* Corresponding authors.
E-mail addresses: sungjkim@hanyang.ac.kr (S.J. Kim), jgjeon41@postech.ac.kr (J. Jeon).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.pnucene.2026.106411
Received 9 May 2025; Received in revised form 10 March 2026; Accepted 16 April 2026

Table 1
Related research on AI and thermal–hydraulic system code

Publication year	Author	Used AI	Data-Driven
2021	Lu et al. (2021)	ANN	O
2022	Song and Ha (2022)	LSTM	O
2023	Chae et al. (2023)	PINN	O
2023	Wang and Ma (2023)	ANN	O
2023	Song and Kim (2023)	LSTM	O
2023	Antonello et al. (2023)	PINN	O
2024	Lee et al. (2024)	CNN with LSTM	O
2024	Song et al. (2024)	CNN with LSTM	O
2025	Wang and Ma (2025)	ANN	O
2025	Baraldi et al. (2025)	PINN	O
2025	Wei et al. (2025)	PINN	X

et al., 2021). This distinguished feature makes it a widely used tool for comparative assessments and parametric studies. Although MELCOR provides flexibility through explicit nodalization, this modeling paradigm inherently requires substantial engineering effort and computational resources when applied to large-scale or highly coupled analyses. Such characteristics highlight the potential value of mesh- or node-independent surrogate approaches as complementary tools for efficient uncertainty and parametric studies.

Recently, deep learning has emerged as a powerful tool for tackling the numerical solution of partial differential equations (PDEs) (Khanal et al., 2025; Jeon et al., 2022). Among the advances, physics-informed neural network (PINN) have gained particular prominence for capturing the behavior of complex, nonlinear physical systems to this end (Raissi et al., 2019; Yang et al., 2024; Mao et al., 2020). By embedding the governing physics directly into the loss function in the form of PDEs, PINN differs from conventional black-box deep learning models by offering physically and theoretically interpretable solutions. Notably, as they inherently do not require training data, they can be classified as not only machine learning algorithm but with a numerical method. Due to their versatility and effectiveness, PINN has garnered considerable attention in a variety of fields, including heat transfer (Cai et al., 2021b; Li et al., 2022), structural (Lai et al., 2021; Bacsa et al., 2023; Abbasi et al., 2025a) and fluid dynamics (Vinuesa and Brunton, 2022; Cai et al., 2021a; Wang et al., 2017; Pang et al., 2020; Abbasi et al., 2025b; Jeon et al., 2024), solid mechanics (Haghighat et al., 2021; Okazaki et al., 2022), and nuclear reactor kinetics (Prantikos et al., 2022), where they have shown promising and efficient alternatives to traditional numerical methods. Once trained, PINN offers notably faster computation compared to traditional numerical methods. However, vanilla PINN still suffer from a key limitation: typically trained for a single instance, meaning any change in input conditions requires retraining from the scratch. Go et al. (2023) demonstrated that if trained in the manner of surrogate-PINN, near real-time inference can be utilized within untrained conditions.

Although numerous studies have attempted to integrate AI techniques with SA codes, as summarized in Table 1, most research has been based on data-driven methodologies and lacks explicit incorporation of physical principles. Lu et al. (2021) utilized ANN for predicting the thermal–hydraulic parameters in KLT-40S, a type of nuclear reactor core, and tube-in-tube once-through steam generators. The model developed was trained and validated with RELAP5/SCDAPSIM demonstrating good agreement and accuracy in rapidly predicting thermal–hydraulic parameters. Song and Ha (2022) developed a machine-learning-based simulation model using LSTM to predict severe accident progression at the Fukushima Daiichi Nuclear power plant. The model was trained and validated with MELCOR using observable parameter in main control room such as liquid levels and pressures of reactor core. The study implied that the time series forecasting was appropriately made and the predictions accuracy remained even under the extreme harsh condition such as corrupted measurement device. Chae et al. (2023) developed PINN based on the data-driven simulation with the data obtained from MARS-KS, one of the system code developed in South Korea based on RELAP5 code (Chung et al., 2010). The study suggests the efficiency of the predefined knowledge and the simulation framework could easily be updated. Wang and Ma (2023) constructed uncertainty analysis tool through bootstrapped ANN by evaluating the severe accidents with the data obtained from MELCOR. Criticizing the heavy load of conventional method of utilizing system code, the bootstrapped ANN demonstrated effective estimation of designated figure of merits such as hydrogen generation and vessel failure timing. Song and Kim (2023) evaluated the importance of utilizing machine learning especially under the situation of non-available observable parameters. LSTM were utilized in the study. By selecting elimination technique with RandomForestRegressor, the ML model was able to predict the target variable within the true values range. Antonello et al. (2023) analyzed the feasibility of application of PINN with regards to a nuclear battery, a type of unique microreactor, under the case of loss of heat sink accident scenario. The study compared the two models, DNN and PINN by training the data retrieved from the system code, RELAP-5 3D. The results showed that training the neural network with the domain knowledge provided numerous benefits such as reducing error and avoiding over or underfitting. Lee et al. (2024) developed a surrogate model for forecasting the progression of the severe accident. The surrogate model was developed based on CNN with LSTM combined. The surrogate model was trained with time series data with thermal–hydraulic behavior. Song et al. (2024) developed a reinforcement learning with the simulation ran by the surrogate model which was trained with the data obtained from MAAP. The developed reinforcement learning model identified the worst-case scenario for the timing of the high pressure injection failure event occurred. Wang and Ma (2025) aimed to specifically analyze the quenching process of conical debris beds for Nordic boiling water reactor. The analysis was conducted by coupling the MELCOR system code with an artificial neural network surrogate model. Baraldi et al. (2025) developed a PINN–based surrogate model enhanced with allocation points to accurately estimate key safety parameters during a Loss of Heat Sink (LOHS) scenario in a nuclear microreactor. While the study successfully incorporated multiphysics modeling within a single system, it did not address the multiple control volume conditions. Wei et al. (2025) proposed a CV-based PINN that improves physical consistency by extending collocation points to control volumes. However, its finite-volume-based formulation limits direct application to lumped-parameter system codes such as MELCOR.

Consequently, most data-driven surrogate models rely heavily on large simulation datasets, thereby implicitly learn system behavior without explicitly incorporating the governing physical laws. As a result, their applicability to untrained scenarios and the interpretability at the equation or term level is inherently constrained. To address these limitation, this study focuses on developing a physics-informed neural network framework based on the governing equations implemented in MELCOR. By explicitly incorporating the conservation equations under multiple control volume and flow path conditions, the proposed approach aims to improve physical consistency and generality for thermal–hydraulic analysis. While the concept of physics-informed

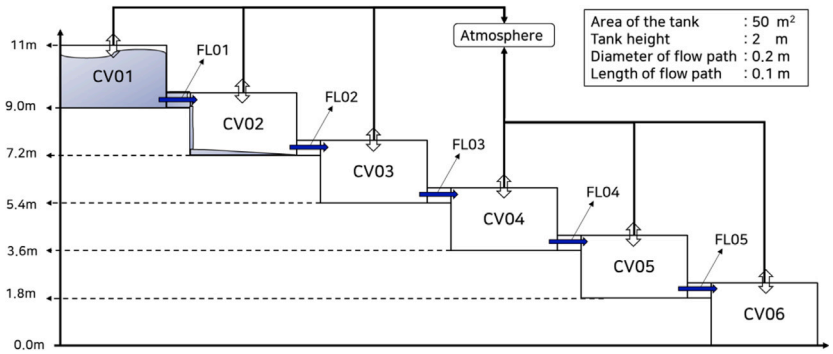


Fig. 1. Nodalization for the scenario model.

learning itself is not new, its systematic formulation and application to MELCOR-based severe accident analysis under generalized control volume conditions constitute the primary contribution of this work.

We present the node-assigned PINN (NA-PINN), a system-code-aligned PINN formulation that naturally matches MELCOR's control volume/flow path nodalization. Unlike conventional surrogate models, our approach requires no simulation data, as it embeds the governing equations directly into the loss function. This enables the model to approximate solutions based solely on physical laws, offering a theoretically grounded alternative to traditional numerical solvers. In addition, the resulting predictions are physically interpretable and consistent with fundamental conservation principles. To evaluate the feasibility of the proposed method, we apply it to a simplified but representative scenario involving the MELCOR control volume hydraulic (CVH) / flow path (FL) package, governed by mass and momentum conservation laws. Furthermore, we extend this application to MELCOR's Heat Structure (HS) package, which is governed by the heat conduction equation and Newton's law of cooling. Focusing specifically on the governing physical laws, we assess the convergence characteristics of the PINN architecture within a controlled test environment. The results serve as a foundational step toward constructing physics-informed surrogate models that are compatible with MELCOR-based TH simulations. The emulator experiments were conducted on a CPU-based computer with 13th Gen Intel(R) Core(TM) i7-13700K CPU and 64 GB RAM. The Physics-Informed Neural Network (PINN) simulations were executed separately using GPU resources (NVIDIA A100-PCIE-40G).

The remainder of this paper is organized as follows. Section 2 describes the emulator for MELCOR's CVH/FL Module, which emulates MELCOR's thermal–hydraulic calculations, serving as a reference for model validation. Section 3 introduces the application of PINN to MELCOR simulations, including the formulation and evaluation of both the vanilla PINN and the lumped-parameter PINN (LP-PINN) for capturing transient flow behavior. Section 4 proposes the NA-PINN as an alternative framework to address the limitations of LP-PINN, detailing its architectural modifications and validating its effectiveness through numerical experiments, which include extending the framework to the HS module to accurately capture heat transfer dynamics. Section 5 discusses potential directions for future research, and Section 6 concludes the study by summarizing the key findings.

2. Developing emulator for MELCOR CVH/FL module

A Python-based emulator model is constructed for the evaluation of comprehension for the calculation in the governing equation of MELCOR, implemented solely using Python code. This emulator facilitates comprehension on the calculation logic and assess the accuracy of the governing equations thereby provide the base equation PINN model could train. As a pilot study, the emulator is developed exclusively targeting the CVH and FL modules, intentionally excluding the HS module. Although thermal aspects are typically significant when performing MELCOR calculations, the thermal module was omitted to reduce numerical complexity and clearly demonstrate the feasibility of this methodology before developing it to more complicated scenarios.

The selected scenario is demonstrated in Fig. 1. The water tanks are demonstrated as control volumes (CVs), and the pipes connecting the CVs are demonstrated as FL. All the CVs are assumed to be open on the top thereby neglect the pressure terms in the analysis and isolate the effect of gravity. Each tank has an area of 50 m^2 with a height of 2 m. The diameter of the FL is 0.2 m with length of 0.1 m, respectively.

In this scenario, six CVs are located with different elevation levels of 1.8 m and five FLs connect the bottom of the donor tank to the top of the receiver tank. At the beginning of the scenario, the highest water tank, CV01 in Fig. 1, is fully filled with water. As the simulation begins, the water flows toward the bottom of the tank due to gravitational force. The scenario is terminated when the momentum of the water no longer exists, that is, when the water levels of the last two CVs are equal.

To evaluate the accuracy of the comprehension, a Python-based emulator model was developed, hereafter referred to as the *emulator model*. This section provides a detailed explanation of the governing equations implemented in MELCOR and their corresponding computational logic. For clarity, the section is organized into four parts. Section 2.1 introduces the governing equations and defines the associated terminology, focusing on the height and momentum equations used in MELCOR's CVH and FL modules. Section 2.2 discusses the computational logic employed in the *emulator model*. Section 2.3 presents the verification results of the *emulator model* by comparing the result with that from MELCOR. Finally, Section 2.4 conducts the sensitivity analysis for the equation terms and summarizes the key findings with modeling considerations.

2.1. MELCOR CVH/FL package: governing equations

Two of the equations used in this case scenario are demonstrated in this section. First is the mass conservation equation in partial differential equation form.

$$\frac{\partial M_{i,m}}{\partial t} = \sum_j \sigma_{ij}\alpha_{j,\phi}\rho^d_{j,m}v_{j,\phi}F_jA_j + \dot{M}_{i,m} \tag{1}$$

The Eq. (1), subscript i and j denote the CV of interest and the corresponding FLs respectively. Thus, the summation over subscript j represents the total mass flow through all associated FLs into or out of the CV_i. The terminology $\dot{M}_{i,m}$ represent non-flow mass generation processes, including condensation, evaporation, and fog precipitation to name a few. This partial differential equation form can be discretized into the form shown in Eq. (2). Each term in Eq. (2) is described in Table 2.

$$M^n_{i,m} = M^o_{i,m} + \sum_j \sigma_{ij}\alpha^n_{j,\phi}\rho^d_{j,m}v^n_{j,\phi}F_jA_j\Delta t + \delta M_{i,m} \tag{2}$$

Table 2
Description of terms in Eq. (2).

Term	Description	Term	Description
$M^n_{i,m}$	New mass of material m in CV_i	$\rho^d_{j,m}$	Material m density of donor
$M^o_{i,m}$	Old mass of material m in CV_i	$v^n_{j,\phi}$	Flow velocity calculated from Eq. (4)
σ_{ij}	Direction of flow in FL j	F_j	Fraction of area opened
$\alpha^n_{j,\phi}$	Volume fraction of ϕ in FL j	A_j	FL area
ϕ	Phase of the material (liquid, air, solid)	$\delta M_{i,m}$	Net external sources

Table 3
Description of terms in Equation 4.

Term	Description	Term	Description
$v^n_{j,\phi}$	New velocity in FL_j	$(\Delta z)^n_{j,\phi}$	Height of the water
$v^{o+}_{j,\phi}$	Old velocity in FL_j	ΔP_j	Pump head pressure
$P^{\tilde{n}}_i$	Predicted value of pressure of donor volume at end of iteration	$K^*_{j,\phi}$	Net form- and wall-loss coefficient
$P^{\tilde{n}}_k$	Predicted value of pressure of receiver volume at end of iteration	$f_{2,j}$	Momentum exchange coefficient
L_j	Inertial length of the pipe	$L_{2,j}$	Effective length over the interphase force
$v'_{j,\phi}$	Tangent linearization	$v^{n^-}_{j,\phi}$	Old value of $v^n_{j,\phi}$ inside the iteration

Table 4
Velocity terms in momentum equations.

No.	Term	Description
1	$(\rho g\Delta z)^{\tilde{n}}_{j,\phi}$	Static head difference
2	$v^o_{j,\phi}(\rho\Delta v)^o_{j,\phi}$	Advection of momentum
3	$\frac{K^*_{j,\phi}\Delta t}{2L_j}\left(\left\|v^{n^-}_{j,\phi}+v'_{j,\phi}\right\|v^n_{j,\phi}-\left\|v'_{j,\phi}\right\|v^{n^-}_{j,\phi}\right)$	Net form- and wall-loss effect
4	$\frac{\alpha_{j,-\phi}f_{2,j}L_{2,j}\Delta t}{\rho_{j,\phi}L_j}\left(v^n_{j,\phi}-v^n_{j,-\phi}\right)$	Interphase force coefficient (momentum exchange)

In this scenario, the cross-sectional areas of the CVs and the density of water are assumed to be constant. Therefore, the mass in each CV can be represented in terms of water level. The water level is also used in the momentum equation, specifically to evaluate the static head in the water tank. The partial differential form of the momentum equation is presented in Eq. (3).

$$\alpha_{j,\phi}\rho_{j,\phi}L_j\frac{\partial v_{j,\phi}}{\partial t}=\alpha_{j,\phi}(P_i-P_k)+\alpha_{j,\phi}(\rho g\Delta z)_{j,\phi}+\alpha_{j,\phi}\Delta P_j+\alpha_{j,\phi}\rho_{j,\phi}v_{j,\phi}(\Delta v_{j,\phi})-\frac{1}{2}K^*_{j,\phi}\alpha_{j,\phi}\rho_{j,\phi}|v_{j,\phi}|v_{j,\phi}-\alpha_{j,\phi}\alpha_{j,-\phi}f_{2,j}L_{2,j}(v_{j,\phi}-v_{j,-\phi}) \quad (3)$$

In Eq. (3), the subscript i denotes the donor CV, k represents the receiver CV, ϕ implies the state of the target material such as liquid or gas, and j refers to the associated FL. Since MELCOR employs a FDM to solve the momentum equation, the Eq. (3) is integrated to derive an approximate discretized form, suitable for numerical implementation. Based on this discretized form, the *emulator model* is developed, and validated by comparing its results with those obtained from MELCOR.

$$v^n_{j,\phi}=v^{o+}_{j,\phi}+\frac{\Delta t}{\rho_{j,\phi}L_j}\left(P^{\tilde{n}}_i+\Delta P_j-P^{\tilde{n}}_k+(\rho g\Delta z)^{\tilde{n}}_{j,\phi}+v^o_{j,\phi}(\rho\Delta v)^o_{j,\phi}\right)-\frac{K^*_{j,\phi}\Delta t}{2L_j}\left(\left|v^{n^-}_{j,\phi}+v'_{j,\phi}\right|v^n_{j,\phi}-\left|v'_{j,\phi}\right|v^{n^-}_{j,\phi}\right)-\frac{\alpha_{j,-\phi}f_{2,j}L_{2,j}\Delta t}{\rho_{j,\phi}L_j}\left(v^n_{j,\phi}-v^n_{j,-\phi}\right) \quad (4)$$

In Eq. (4), the form- and wall-loss term is derived using a linear approximation based on a Taylor series expansion. Specifically, the loss term in Eq. (3), originally expressed as $|v|v$, is approximated in a linearized form. As a result, the friction loss term in Eq. (4) is expressed as $\left|v^{n^-}_{j,\phi}+v'_{j,\phi}\right|v^n_{j,\phi}-\left|v'_{j,\phi}\right|v^{n^-}_{j,\phi}$.

The choice of $v'_{j,\phi}$ in this equation depends on the flow direction. If $v^{n^-}_{j,\phi}$ is positive, $v'_{j,\phi}=v^{n^-}_{j,\phi}$; otherwise, $v'_{j,\phi}$ is set to be zero. In this scenario, reverse flow is not considered, as no countercurrent flow occurs.

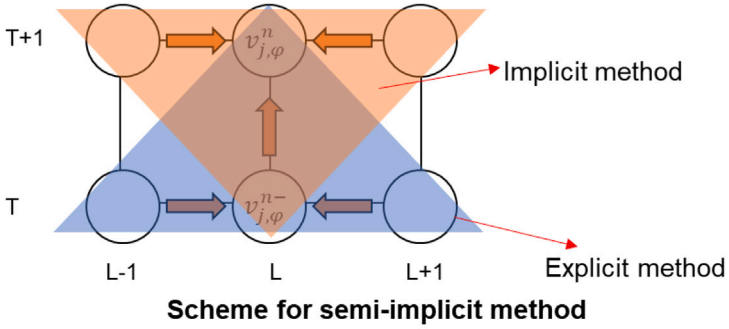
Table 3 explains the terminologies for Eqs. (3) and (4). As per the assumptions, the pressure difference is neglected owing to the open tank. Since there are no force applied in this scenario, the pressure created by pump could also be neglected. Table 4 demonstrates the effective terms for the velocity calculation. For static head difference term, the elevation difference between the two tanks need to be updated as the time flows during the calculation. To be more specific, $\tilde{n}$ indicates the predicted value at the end of calculation step and the static head difference term is explained in the following form:

$$(\rho g\Delta z)^{\tilde{n}}_{j,\phi}=(\rho g\Delta z)^{o+}_{j,\phi}+\frac{\partial(\rho g\Delta z)_{j,\phi}}{\partial M_{i,P}}\left(M^n_{i,P}-M^{o+}_{i,P}\right)+\frac{\partial(\rho g\Delta z)_{j,\phi}}{\partial M_{k,P}}\left(M^n_{k,P}-M^{o+}_{k,P}\right) \quad (5)$$

In this scenario, due to its simplicity, the mass lost from the donor CV is equal to the mass gained by the receiver CV, ensuring the flow consistency. Under this condition, the representative density $\tilde{n}$ can be replaced by o. Consequently, the static head term depends solely on the elevation difference Δz between the donor and receiver CVs. However, the formulation of this methodology relies on implicit assumptions and procedural evaluations, making it difficult to incorporate directly into equation based loss terms within a PINN framework.

2.2. MELCOR CVH/FL package: numerical schemes

Within MELCOR calculation, the terms are retrieved either with implicit method or explicit method. The numerical scheme is demonstrated in Fig. 2. From the figure, implicit method implies calculating the future value from the future surrounding values whereas explicit method is calculating the future value purely with the current values. From Table 4, term 1 and 4 are calculated implicitly, term 2 is calculated explicitly, and term 3 is calculated with both explicitly and implicitly due to mixture of current value and the future value. Because the term 3 in Eq. (3) is nonlinear, the calculation requires iteration until the convergence of the specific criteria. The calculation process is described in Fig. 3. The absolute difference between new and old value inside the iteration loop is required to converge within the 9%. Following the MELCOR CVH/FL formulation (Gauntt et al., 2021), a convergence tolerance of 9% is employed, which serves to preserve the validity of the linearized friction term while preventing unphysical velocity reversal during iteration. This criterion only regards the local friction linearization while enabling the conservation of mass or energy, which are ensured independently by the governing equations.



Scheme for semi-implicit method

$$v_{j,\varphi}^{n} = v_{j,\varphi}^{o+} + \frac{\Delta t}{\rho_{j,\varphi} L_j}\left(P_i^n + \Delta P_j - P_k^n + (\rho g \Delta z)_{j,\varphi}^{n} + v_{j,\varphi}^{o}(\rho \Delta v)_{j,\varphi}^{o}\right)$$
$$- \frac{K *_{j,\varphi} \Delta t}{2L_j}\left(|v_{j,\varphi}^{n-}| + v'_{j,\varphi}|v_{j,\varphi}^{n} - |v'_{j,\varphi}|v_{j,\varphi}^{n-}\right) - \frac{\alpha_{j,-\varphi} f_{2,j} L_{2,j} \Delta t}{\rho_{j,\varphi} L_j}\left(v_{j,\varphi}^{n} - v_{j,-\varphi}^{n}\right)$$

Momentum equation

Fig. 2. Numerical scheme of calculation process utilized in MELCOR.

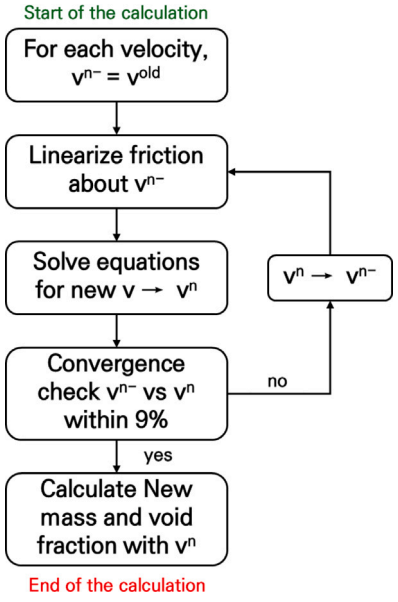


Fig. 3. Velocity calculation process in MELCOR.

The velocity calculation begins by adopting the previously computed value, denoted as v^{n^-}, and linearizing the nonlinear friction term about this reference state. Specifically, the term $v|v|$ in Eq. (3) is approximated through a first order linearization with respect to v^{n^-}, resulting in the expression given in Eq. (6). This formulation enables a stable finite-difference implementation of the nonlinear momentum equation.

$$v^n|v^n| = v^{n^-}|v^{n^-}| + (|v^{n^-}| \pm v^{n^-})(v^n - v^{n^-}) \tag{6}$$

Fig. 3 illustrates the iterative procedure used to evaluate the updated velocity $v_{j,\phi}^{n}$. Convergence is assessed based on the difference between successive velocity estimates, and the linearization point is updated iteratively until the prescribed criterion is satisfied. The resulting velocity and mass updates are implemented via the procedures summarized in Algorithms 1 and 2, with the water level calculations applied to the connected donor and receiver CVs (CV_d and CV_r), and the velocity iteration performed for each FL $j = 1, \ldots, 5$. In the CVH/FL packages, MELCOR employs a semi-implicit finite difference scheme to maintain numerical stability while preserving computational efficiency. Rather than assembling and solving a global system of equations, the discretized momentum equations are solved locally for coupled FLs using iterative techniques. This combination of linearized and iterative formulations yields the final velocity solution from the discretized momentum balance.

Algorithm 1 Mass Update for Coupled Tank System (CVH/FL)

Require: v^o, m_d, m_r, Δt, A_p, A_t, elev, etc.
Ensure: m_d^n, m_r^n, v^n, α

Compute donor height:

$$H_d = \frac{m_d}{\rho A_t} + \texttt{elev}$$

if $H_d - \texttt{elev} \le \varepsilon$ **then**
  $v^n \leftarrow 0, \quad \alpha \leftarrow 1.0$
  $m_d^n \leftarrow m_d, \quad m_r^n \leftarrow m_r$
else
  Compute the heights to determine the static head:

$$H'_d = H_d - \texttt{elev}, \quad H_r = \frac{m_r}{\rho A_t}$$

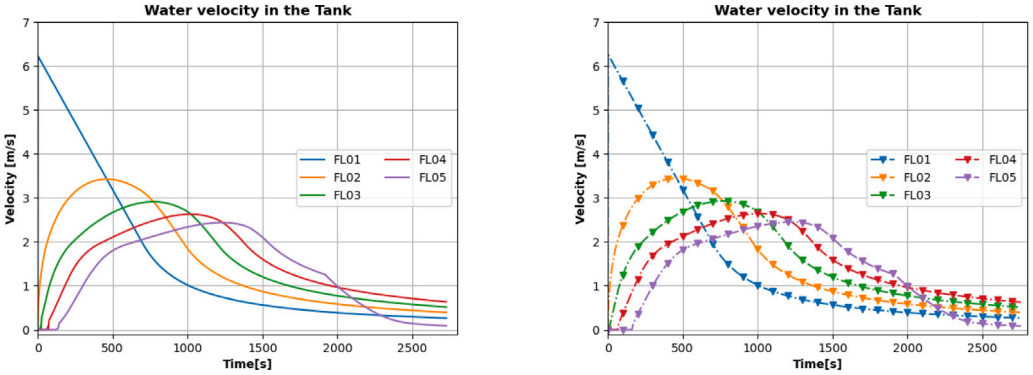
  if $H_r < \texttt{elev}$ **then**
    $\Delta z \leftarrow H_d - \texttt{elev}$ ▷ Flow into initially empty tank
  else
    $\Delta z \leftarrow H_d - H_r$ ▷ Post-alignment inter-tank flow
  end if
  Compute void fraction:

$$\alpha = \begin{cases} 0, & H_d \ge 0.2 \\ 1 - \frac{H_d}{0.2}, & 0 \le H_d < 0.2 \end{cases}$$

  Compute velocity v^n using Algorithm 2
  $\Delta m = \rho \cdot v^n \cdot \Delta t \cdot A_p \cdot (1 - \alpha)$
  Update:

$$m_d^n = m_d - \Delta m, \quad m_r^n = m_r + \Delta m$$

end if
return m_d^n, m_r^n, v^n, α, Δz



(a) Water level result from MELCOR (b) Water level result from Emulator model

Fig. 4. Velocity profiles in the FL computed by MELCOR (left) and emulator model (right).

Algorithm 2 Velocity Iteration for Given Static Head and Void Fraction

Require: v^o, α, α_o, Δz, Δt, L, K, A_p, A_t, ρ
Ensure: Final velocity v^n

1: $v^{n^-} \leftarrow v^o$, $\quad v_{o_2} \leftarrow v^o + (\alpha_o - \alpha)(-v^o)$
2: **repeat**
3: $\quad v^{n^-} \leftarrow v^{n+}$
4: $\quad$ Compute:

$$v^{n+} = \frac{v_{o_2} + \frac{\Delta t}{L}\left(g\Delta z + v_{o_2}^2 \cdot \frac{A_p}{A_t}\right) + \frac{K\Delta t}{2L} \cdot (v^{n^-})^2}{1 + \frac{K\Delta t}{2L}(v^o + v^{n^-}) + \frac{\alpha f L_2 \Delta t}{\rho L} + \frac{\Delta t}{L} \cdot v_{o_2} \cdot \frac{A_p}{A_t}}$$

5: **until** $|v^{n+} - v^{n^-}| < 0.09 \cdot |v^{n^-}|$
6: **return** $v^n \leftarrow v^{n+}$

2.3. Development of Python emulator model and verification

Prior to developing the PINN-based model, it is essential to establish a comprehensive understanding of the governing equations and numerical scheme implemented in MELCOR. Based on the analysis in Sections 2.1 and 2.2, a Python-based *emulator model* was developed to replicate the numerical behavior of MELCOR. The *emulator model* follows the algorithmic structures summarized in Algorithms 1 and 2, which represent the velocity iteration and mass update procedures implemented in the CVH/FL packages of MELCOR. The *emulator model* mimics the velocity calculation scheme where the state variables at time step $n+1$ are predicted from the converged solution at time step n. The time step is set to $\Delta t = 1$ s, which is consistent with the temporal resolution of the MELCOR output data used emulator evolve on an identical temporal grid. MELCOR internally employs a semi-implicit formulation with iterative velocity linearization, as illustrated in Fig. 2. The *emulator model* reproduces this behavior in an operator-splitting manner by sequentially applying the velocity iteration and mass update procedures at each time step. This design preserves the essential numerical characteristics of MELCOR while maintaining computational efficiency and transparency. The *emulator model* also offers greater flexibility in manipulating individual terms within the equations, enabling a sensitivity study to identify the dominant contributors. This capability facilitates model simplification by eliminating negligible terms and lays the foundation for efficient training in the subsequent PINN development. The comparison between the MELCOR results and the *emulator model* predictions focuses on the two key parameters: FL velocity and water level in the tank. A time-step pairwise comparison is performed using the two standard error metrics; mean absolute error (MAE) and mean squared error (MSE) which are defined by the following equations:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i(t) - \hat{y}(t)\right| \tag{7}$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i(t) - \hat{y}(t)\right)^2 \tag{8}$$

Where N is the number of data, $y_i(t)$ is the desired value (MELCOR values) and $\hat{y}(t)$ is the predicted value.

Table 5
MAE and MSE of velocity and height predictions.

Metric	Velocity	Height
MAE	0.01002	0.00248
MSE	0.00023	1.6816×10^{-5}

A velocity comparison was conducted between MELCOR (target values) and the *emulator model* (predicted values). Fig. 4(a) shows the velocity profile obtained from MELCOR, while Fig. 4(b) presents the corresponding prediction from the *emulator model*. The qualitative comparison from Fig. 4 indicates the emulator reproduces the overall velocity trends of the MELCOR results with reasonable agreement.

To quantitatively evaluate the level of agreement, the MAE and MSE over the entire simulation were computed as 0.01002 and 0.00023, respectively. These metrics supports a close agreement between the emulator predictions and the MELCOR results.

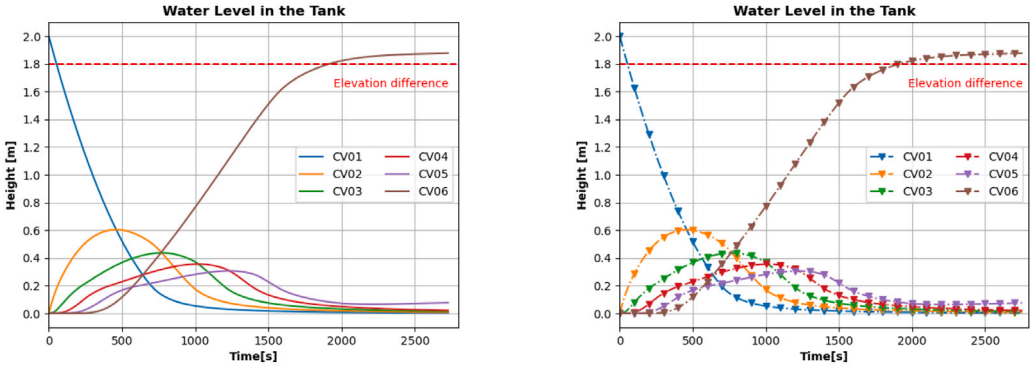
Fig. 5 depicts the evolution of the water level in each tank. Graph (a) presents the reference results from MELCOR, while graph (b) shows the corresponding predictions from the *emulator model*. As the water level decreases, a corresponding reduction in the velocity through the connecting FL05 is observed. This trend is primarily governed by Term 1 in Table 4, which defines the velocity as a function of the height difference between connected CVs.

The red dotted line in Fig. 5(b) indicates the elevation at which the water levels in the two tanks approach equilibrium, resulting in a substantial reduction of the static head difference. This transition is consistently reflected in the FL05 velocity profile shown in Fig. 1.

Over the whole scenario progression, the emulator demonstrated MAE of 0.00248 and MSE of 1.6816×10^{-5}, indicating reasonable agreement with MELCOR calculation results in terms of mass balance. A quantitative comparison of the error metrics is provided in Table 5.

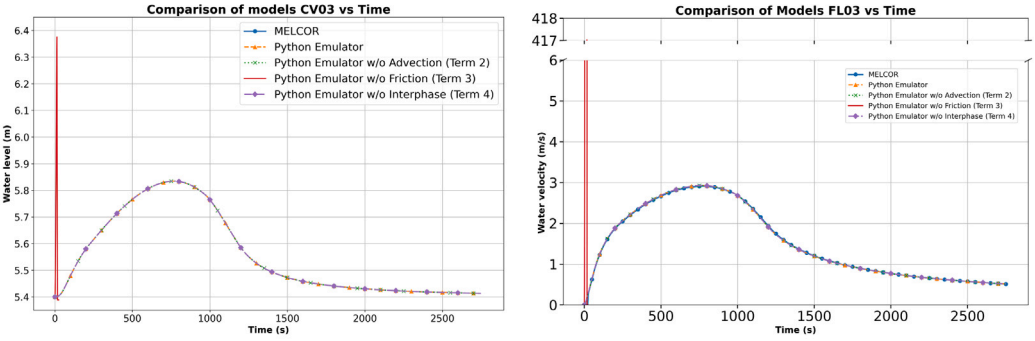
2.4. Sensitivity analysis for the equation

While MELCOR provides a detailed and validated simulation environment, the modification of individual terms in its governing equations is restricted to the code developers, which limits the possibility of



(a) Water level result from MELCOR (b) Water level result from Emulator model

Fig. 5. Water level profiles in the water tanks computed by MELCOR (left) and emulator model (right).



(a) Sensitivity analysis for water level at CV03 (b) Sensitivity analysis on velocity at FL03

Fig. 6. Comparison between the models for water level (left) and velocity (right).

performing term-wise analyses. However, the emulator, as an in-house code, enables sensitivity analyses by selectively removing or modifying individual terms. In this section, a sensitivity analysis is conducted to identify non-dominant terms that exhibit negligible influence on flow behavior. This process is a critical step toward optimizing the governing equation for a PINN framework, which is expected to reduce computational cost and enhance model robustness. Based on Table 4, three terms are selected for sensitivity analysis: the advection momentum term, net form- and wall-loss term, and interphase force term. Term 1, representing the static head difference, is excluded from the sensitivity analysis because it is the primary driving force for flow generation, and the velocity equation cannot be evaluated in its absence.

As shown in Fig. 4(b), the emulator demonstrates comparable behavior to MELCOR. To further investigate the governing equation, the velocity-related terms defined in Table 4 are analyzed, focusing on CV03 and FL03. Fig. 6(a) and (b) present the water level and velocity responses, respectively, under the selective exclusion of terms 2, 3, and 4 from Table 4. Each term was independently removed from the equation to evaluate its influence on the velocity profile. To quantitatively assess the impact of each term, MSE and MAE of the water level at CV03 and the velocity at FL03 were calculated with respect to the MELCOR results, as summarized in Table 6. The baseline emulator model, in which all governing terms are retained, serves as the reference for evaluating the reduced models.

The results in Table 6 indicate that removing term 2 and 4 produces negligible changes to the full model, suggesting that their contribution is insignificant. In contrast, the exclusion of term 3 leads to a drastic increase in both MSE and MAE for velocity and water level predictions. This confirms that the form- and wall-loss term is the dominant contributor governing the flow behavior in the considered scenario. Specifically, when this term is omitted, the velocity peaks at 417.65 m/s, and the water level rapidly reaches 0.977 m before declining. The lack of observable differences when excluding Term 2 and 4 further underscores that Term 3 is the sole dominant factor in this scenario.

This sensitivity analyses allows simplification of the governing equations by eliminating terms with negligible influence. Removing these insignificant terms prevents the accumulation of unnecessary errors within the PINN model and reduces computational complexity. Based on the remaining dominant terms, the equations are reformulated as loss functions for PINN training. Accordingly, the final equations for mass and momentum conservation are given as follows:

$$f_1 = \sum_j \alpha_{j,\phi}\rho^d_{j,m} v_{j,\phi} A_j - \frac{\partial M_{i,m}}{\partial t} \tag{9}$$

$$f_2 = \alpha_{j,\phi}\rho_{j,\phi} L_j \frac{\partial v_{j,\phi}}{\partial t} - \alpha_{j,\phi}(\rho g \Delta z)_{j,\phi} + \frac{1}{2}\alpha_{j,\phi}\rho_{j,\phi} K^*_{j,\phi} |v_{j,\phi}| v_{j,\phi} \tag{10}$$

Additionally, a substantial difference in computation time is observed between MELCOR and the emulator. While MELCOR requires 0.5625 s to complete the simulation, the emulator model completed the task in 0.03688 s. The difference of calculation speed comes from the scope of the calculation. MELCOR contains broader range of tasks such as the heat transfer between the physical models and the behavior of the radionuclides are also included in the scope of MELCOR's work. In contrast, the emulator model has narrowed its focus of calculating the mass and momentum equation, thereby achieving faster computation.

Table 6
Quantitative MSE and MAE comparisons between the reference emulator and reduced emulator models with respect to MELCOR.

Case	CV03 H3 MSE	CV03 H3 MAE	FL03 V3 MSE	FL03 V3 MAE
Reference emulator	8.36×10^{-6}	1.77×10^{-3}	2.60×10^{-4}	9.96×10^{-3}
No Advection (Term 2)	8.36×10^{-6}	1.77×10^{-3}	2.60×10^{-4}	9.96×10^{-3}
No Friction (Term 3)	1.98×10^{-1}	2.97×10^{-1}	4.14×10^{4}	1.32×10^{2}
No Interphase (Term 4)	8.36×10^{-6}	1.78×10^{-3}	2.60×10^{-4}	9.96×10^{-3}

3. PINN-based CVH/FL module

3.1. Background on physics-informed neural network (PINN)

PINN represents a pioneering approach that incorporates physical laws directly into neural network architectures by embedding PDEs into the loss function (Raissi et al., 2019). The fundamental concept of PINN lies in their ability to solve PDEs through neural networks, where the network's loss function includes PDE-based terms that approach zero as the solution converges to satisfy the governing equations. For a well-posed problem, these PDEs are typically complemented by boundary conditions (BCs) and initial conditions (ICs) to ensure solution uniqueness. PINN can be broadly categorized into two frameworks based on their training approach. The primary focus of this study is on data-free PINN, which solve PDEs without labeled solution data, particularly suitable as alternatives to conventional computational solvers such as MELCOR. In this framework, the neural network is trained solely using the physical constraints imposed by the PDEs and their associated BCs/ICs. While an alternative framework known as data-driven PINN exists, which incorporates additional loss terms from labeled data to enhance prediction accuracy (Yang et al., 2024), our investigation concentrates on data-free PINN due to their potential as standalone computational solvers. The data-free PINN framework takes spatiotemporal coordinates $(\mathbf{x}, t)$ as input and predicts the quantities of interest at these points. The physics-based loss function comprises PDE losses and IC/BC losses. For a general PDE of the form:

$$\mathcal{F}(\mathbf{u}(\mathbf{x},t), \nabla\mathbf{u}(\mathbf{x},t), \nabla^2\mathbf{u}(\mathbf{x},t), \ldots) = 0, \quad (\mathbf{x},t) \in \Omega \times [0,T] \tag{11}$$

the PDE loss is defined as:

$$\mathcal{L}_{PDE} = \|\mathcal{F}(\mathbf{u}_{NN}(\mathbf{x},t), \nabla\mathbf{u}_{NN}(\mathbf{x},t), \nabla^2\mathbf{u}_{NN}(\mathbf{x},t), \ldots)\|, \quad (\mathbf{x},t) \in \Omega \times [0,T] \tag{12}$$

where $\mathbf{u}_{NN}$ represents the predicted output values from neural network, Ω is the spatial domain, and $[0,T]$ is the time interval of interest. Conventionally, initial and boundary conditions are imposed through additional loss terms. For ICs of the form $\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x})$, the IC loss is:

$$\mathcal{L}_{IC} = \|\mathbf{u}_{NN}(\mathbf{x},0) - \mathbf{u}_0(\mathbf{x})\|, \quad \mathbf{x} \in \Omega \tag{13}$$

Similarly, for BCs of the form $\mathbf{u}(\mathbf{x},t) = \mathbf{g}(\mathbf{x},t)$ on $\partial\Omega$, the BC loss is:

$$\mathcal{L}_{BC} = \|\mathbf{u}_{NN}(\mathbf{x},t) - \mathbf{g}(\mathbf{x},t)\|, \quad (\mathbf{x},t) \in \partial\Omega \times [0,T] \tag{14}$$

The total PINN loss is then:

$$\mathcal{L}_{PINN} = \mathcal{L}_{PDE} + \mathcal{L}_{IC} + \mathcal{L}_{BC} \tag{15}$$

And there is an alternative way to enforce BCs directly in the network architecture, known as the hard constraint approach (Lu et al., 2021b). In this case, the neural network output $\mathbf{u}_{NN}$ is modified to automatically satisfy the boundary conditions. For BCs of the form $\mathbf{u}(\mathbf{x},t) = \mathbf{g}(\mathbf{x},t)$ on $\partial\Omega$, the modified output is:

$$\mathbf{u}(\mathbf{x},t) = h(\mathbf{x}) \cdot \mathbf{u}_{NN}(\mathbf{x},t) + \mathbf{g}(\mathbf{x},t) \tag{16}$$

where $h(\mathbf{x})$ is a function that equals 0 on $\partial\Omega$. With the hard constraint approach for BCs/ICs, the PINN's loss only includes the PDE term since they already satisfy BCs/ICs:

$$\mathcal{L}_{PINN} = \mathcal{L}_{PDE} \tag{17}$$

In contrast to the multiplicative formulation presented in Eq (16), this study adopts an additive strategy, hereafter referred to as the "shifting" method. Further details on this formulation are provided in Section 3.2

All partial derivatives in the equations for the loss functions of the PINN are computed using automatic differentiation within the neural network (Raissi et al., 2019). The loss terms are evaluated at spatiotemporal collocation points: PDE losses within the domain $\Omega \times [0,T]$ and IC/BC losses at their respective locations. Naturally, the strategic selection of these collocation points is known to significantly influence the PINN's performance (Lu et al., 2021a; Nabian et al., 2021; Mao et al., 2020; Yang et al., 2024). After training, PINN provide significantly faster computational performance than conventional numerical methods. Although vanilla PINN is typically limited to single-instance applications, this speed advantage can be leveraged when they are trained in the style of surrogate PINN.

3.2. Reformulation of PINN for MELCOR

The vanilla PINN architecture is formulated as a fully connected neural network (FCNN), where the input variables are the time variable t and the discrete system index i, introduced to emulate spatial information. Unlike standard PINN, which requires continuous spatiotemporal inputs as discussed in Section 3.1, MELCOR calculates solution variables individually for each CV. As a result, spatial coordinates are not inherently required in this context. The index i is introduced solely to adapt the formulation to a standard PINN structure, and the network predicts the solution variables as functions of (i,t).

The network outputs are defined in a two-dimensional form, consisting of the water level $h(i,t)$ and the velocity $v(i,t)$ associated with each control volume index i. For example, when the index i is set to 1, in Fig. 1 the network output corresponds to the water level h_1 in CV01 and the velocity v_1 in FL01. Hence, when $i = 6$, the output variable v_6 does not correspond to a valid FL and is therefore excluded from the training and verification. The overall vanilla PINN framework used in this study is illustrated in Fig. 7. Alternatively, a one-dimensional output formulation, in which indices were assigned separately to velocities and water levels, was also tested but showed inferior performance. This study therefore adopts only the two-dimensional output formulation.

To strictly enforce the initial condition, while no boundary conditions are considered in this study, a shifting method based on an additive formulation was adopted. While standard PINN approaches often enforce initial conditions via soft constraints Eq. (15) or multiplicative hard constraints Eq. (16), these methods have limitations. Soft constraints introduce additional optimization objectives, often leading to unbalanced training where the network struggles to satisfy both the PDE and the initial condition simultaneously. Multiplicative hard constraints resolve this but can introduce unnecessary functional complexity. Multiplying the neural network output by a weighting function $h(t)$ imposes an artificial non-linearity, which can complicate the optimization landscape and hinder the network's ability to approximate the solution's dynamics effectively.

In contrast, the shifting method adopted in Eq. (18) offers distinct advantages. By employing an additive formulation, it strictly satisfies the initial condition $u(i,0) = u_{i,0}$ via a simple linear transformation. This approach preserves the inherent functional properties of the neural network without introducing structural distortions from multiplicative

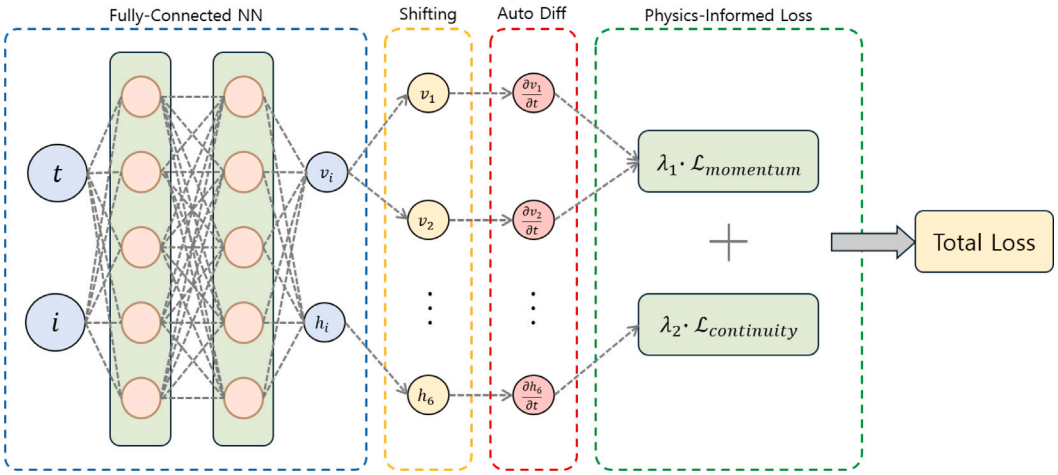


Fig. 7. Architecture of the vanilla PINN model.

terms. Consequently, it ensures that the network focuses solely on learning the system dynamics governed by the PDE, resulting in a more stable and efficient training process compared to the forms discussed in Section 3.1.

In this formulation, $\mathbf{u}_{NN}(t)$ denotes the original network output at time t, $\mathbf{u}_{NN}(0)$ represents the network prediction at the initial time $t = 0$, and $\mathbf{u}_0$ is the given initial condition. By directly incorporating the initial condition into the network architecture, it is inherently satisfied throughout the training process.

$$\mathbf{u}(i,t) = \mathbf{u}_{NN}(i,t) - \mathbf{u}_{NN}(i,0) + \mathbf{u}_{i,0}. \tag{18}$$

Since the initial condition is strictly satisfied through the hard constraint formulation and no boundary conditions are imposed, the total loss function $\mathcal{L}_{PDE}$ is defined solely by the momentum conservation loss $\mathcal{L}_{momentum}$ and the continuity equation loss $\mathcal{L}_{continuity}$. This is expressed in Eq (19) as

$$\mathcal{L}_{PDE} = \lambda_1 \cdot \mathcal{L}_{momentum} + \lambda_2 \cdot \mathcal{L}_{continuity} \tag{19}$$

where the weighting parameters are set to $\lambda_1 = 1$ and $\lambda_2 = 0.001$. This scaled weighting is adopted to prevent the continuity residual from dominating the optimization and to maintain a balanced contribution of the governing constraints during training. In practice, this choice yields consistently improved performance across the test cases considered in this work, as reflected by the training records summarized in Appendix A.2.

The momentum conservation loss $\mathcal{L}_{momentum}$, derived from the governing momentum equation, is calculated as shown in Eq (20). In this expression, N_j denotes the number of pipes, defined as $N_i - 1$, and $v_{j,\phi}$ represents the velocity of the jth FL. As noted above, CV-based approach was used in MELCOR code. In addition, N_t refers to the number of collocation points at which the residuals of both the momentum and continuity equations are evaluated during training.

$$\mathcal{L}_{momentum} = \frac{1}{N_t}\sum_{j=1}^{N_j}\left(L\frac{\partial v_{j,\phi}}{\partial t} - g\Delta z + \frac{1}{2}K^*_{j,\phi}|v_{j,\phi}|v_{j,\phi}\right)^2 \tag{20}$$

For mass conservation, the continuity equation is reformulated under the incompressible flow assumption. The mass $M_{i,m}$ within the ith CV is expressed in Eq (21), where $H_{i,m}$ represents the water level in the CV, A_i is the cross-sectional area of the CV base, and $\rho^d_{i,m}$ denotes the fluid density. Using this formulation, the continuity residual for each CV is defined by Eq (22), where σ_{ij} indicates the flow direction, $\alpha_{j,\phi}$ is the flow area ratio, F_j is the flow coefficient, and A_j is the cross-sectional area of the FL.

$$M_{i,m} = H_{i,m}A_i\rho^d_{i,m} \tag{21}$$

$$A_i\rho^d_{i,m}\frac{\partial H_{i,m}}{\partial t} = \sum_j \sigma_{ij}\alpha_{j,\phi}\rho^d_{j,m}v_{j,\phi}F_jA_j \tag{22}$$

To account for CV-location-dependent flow characteristics, the continuity residual is classified into three cases, and a node-specific loss function is constructed accordingly. Specifically, in the serial tank–pipe topology, each control volume CV_i is connected to at most two adjacent FLs, so the summation term in Eq. (22) reduces according to whether CV_i is the first, the last, or an intermediate volume. Consequently, Eq. (22) yields Eqs. (23)–(25) for $i = 1$, $i = N_i$, and $2 \le i \le N_i - 1$, respectively.

1. In the first CV, water can only exit through the connected pipe, leading to Eq. (23).

2. In the last CV, water can only enter through the connected pipe, satisfying Eq. (24).

3. For an intermediate CV (neither the first nor the last), water flows in from the preceding CV and exits to the next CV, which results in Eq. (25).

$$\mathcal{R}_1 = A_i\rho^d_{i,m}\frac{\partial H_{i,m}}{\partial t} + \rho^d_{j,m}v_{j,\phi}F_jA_j \tag{23}$$

(derived from Eq. (22) for $i = 1$)

$$\mathcal{R}_i = A_i\rho^d_{i,m}\frac{\partial H_{i,m}}{\partial t} - \rho^d_{j-1,m}v_{j-1,\phi}F_{j-1}A_{j-1} \tag{24}$$

(derived from Eq. (22) for $i = N_i$)

$$\mathcal{R}_2 \sim \mathcal{R}_{i-1} = A_i\rho^d_{i,m}\frac{\partial H_{i,m}}{\partial t} + \rho^d_{j,m}v_{j,\phi}F_jA_j - \rho^d_{j-1,m}v_{j-1,\phi}F_{j-1}A_{j-1} \tag{25}$$

(derived from Eq. (22) for $2 \le i \le N_i - 1$)

The total continuity loss $\mathcal{L}_{continuity}$ is calculated as the mean squared sum of the local continuity residuals across all CVs, where each residual $\{\mathcal{R}_i\}_{i=1}^{N_i}$ is defined according to the flow characteristics of the first, last, and intermediate CVs in Eqs. (23)–(25). The final form of the continuity loss is given in Eq (26).

$$\mathcal{L}_{continuity} = \frac{1}{N_t}\sum_{i=1}^{N_i}\mathcal{R}_i^2 \tag{26}$$

For network training, the ELU activation function (Clevert, 2015) was used to prevent vanishing gradient issues, and He initialization was applied for the initialization of the weight. The learning rate was set to 0.0001, with a scheduler reducing the learning rate by 99.99% at each epoch. In addition, Min–max scaling is applied to the time variable t to maintain numerical stability during training by normalizing the input range to [0, 1], as shown in Eq (27), where $t_{\min}$ and $t_{\max}$ denote the minimum and maximum time values of the collocation points.

$$t' = \frac{t - t_{\min}}{t_{\max} - t_{\min}}, \tag{27}$$

Table 7
Hyperparameters for each case according to the number of CVs.

N_i	End time	Collocation points	Epochs	Hidden layers ×nodes
2	1000	2500	30 000	10 × 192
3	1400	3000	40 000	10 × 256
6	2800	6000	50 000	10 × 368

Through this formulation, the proposed PINN architecture is designed to capture the coupled dynamics of CVs and FLs while strictly adhering to the governing physical laws. However, the effectiveness of this approach must be verified across various system configurations to assess its accuracy and scalability. Therefore, in the following section, a series of case studies are conducted under different CV settings to evaluate the performance of the proposed model.

3.3. Case study matrix

In this section, the performance and scalability of the vanilla PINN architecture are evaluated through a series of numerical experiments. The objective of these case study is to assess the model's ability to generalize across systems with varying complexity due to differing numbers of CVs. To this end, simulations are conducted for systems comprising $N_i = 2$, $N_i = 3$, and $N_i = 6$, where N_i denotes number of CVs.

All case studies are conducted under identical physical conditions as defined in Section 2. However, the termination criteria differ due to the fundamental differences in how the models handle temporal evolution. The Python-based emulator model follows a step-by-step computation from the initial condition, where the simulation progresses sequentially in time until a physical convergence condition is met—for example, when the water levels of the final CV and its preceding CV become sufficiently close.

In contrast, the PINN model learns the solution over the entire time domain simultaneously, rather than advancing in discrete time steps. As a result, it requires an explicitly predefined termination time to define the temporal learning domain. Furthermore, as the number of CVs increases, the scenario duration naturally becomes longer, and the termination time for PINN must be extended accordingly.

To ensure consistency between the models, the termination time for each PINN case was set by rounding up the final simulation time of the emulator model to the nearest hundred. For example, in the case with six CVs, the emulator's simulation terminated at 2727 s, so the corresponding termination time for PINN was set to 2800 s.

To account for the increased system complexity associated with larger numbers of CVs, the network architecture and training parameters are scaled accordingly. Specifically, the number of collocation points, training epochs, and the size of the neural network increase proportionally with N_i. Collocation points are uniformly placed at intervals of approximately 0.4 s to ensure sufficient temporal resolution. The detailed hyperparameter settings for each configuration are summarized in Table 7, with corresponding results discussed in the subsequent section.

3.4. PINN module verification

A comparative analysis was conducted to evaluate the accuracy of the proposed PINN by examining its predictions against those of the emulator model. The evaluation considers water level and velocity profiles across three different CV configurations ($N_i = 2$, 3, and 6), allowing for a systematic investigation of the model's limitations under increasing system complexity.

The accuracy of the PINN model is quantitatively assessed using the mean absolute error (MAE) and mean squared error (MSE), calculated according to Eqs. (7) and (8), respectively. These metrics provide a quantitative measure of the discrepancies between the vanilla PINN predictions and the reference solutions.

Fig. 8 presents the comparison results for the cases with $N_i = 2$, 3, and 6 CVs. The figure displays the time histories of water level and velocity for each case, where the surrogate model results are represented by dotted lines and the vanilla PINN predictions by solid lines. The vanilla PINN results indicate that, while the model partially captures the transient behavior of the system, its overall predictive performance remains limited.

The corresponding MAE and MSE values confirm the insufficient accuracy of the vanilla PINN across all cases, as summarized in Table 8. A detailed analysis of the vanilla PINN results indicates that, while the model captures the overall trend of the system dynamics to some extent, it fails to resolve the detailed temporal variations required to accurately represent the transient behavior. The observed behavior suggests that the vanilla PINN struggles to learn the complex interactions between CVs. As a result, the network fails to establish an appropriate mapping between inputs and outputs, leading to a solution where the predicted state exhibits insufficient temporal variability.

In addition, Fig. 9 presents the loss histories during training for each case. In all cases, the loss functions exhibit an apparently adequate convergence trend. However, the losses do not converge to sufficiently small values, and the resulting predictions fail to reflect the expected physical behavior, as confirmed in Fig. 8. This limitation is likely due to MELCOR, as a lumped-parameter system, does not require explicit spatial information as input, whereas an artificial index was introduced into the vanilla PINN architecture to incorporate spatial information similar to standard PINNs. These results highlight the difficulty of attaining MELCOR-level accuracy using the vanilla PINN approach.

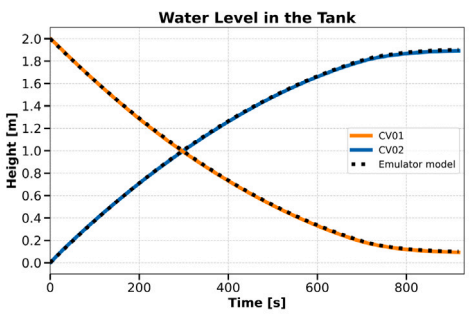
These results demonstrate that the LP-PINN formulation exhibits clear limitations in modeling the transient TH behavior of the system. One possible contributing factor is the use of a single neural network to simultaneously predict multiple governing PDEs, which can give rise to gradient conflicts during training. However, it remains unclear whether the observed performance limitations is primarily caused by the mismatch between the vanilla PINN architecture and the lumped-parameter formulation of MELCOR, or by gradient conflicts inherent to multi-equation learning. Therefore, it is necessary to develop a model that explicitly addresses gradient conflict issues and to compare its performance against the vanilla PINN in order to isolate the dominant source of error.

3.5. Architecture of lumped-parameter PINN

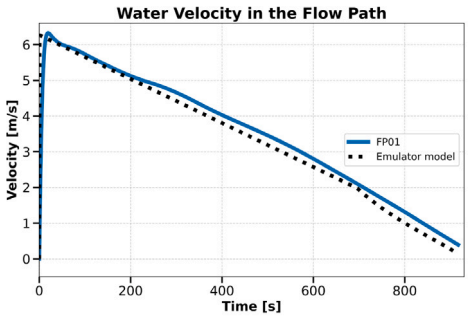
Although the loss function of the vanilla PINN is formulated to enforce the governing equations of MELCOR, its input–output structure does not align with the lumped-parameter nature of the MELCOR system. In particular, the vanilla PINN introduces an artificial spatial index as an input variable in order to conform to the standard PINN formulation, even though such spatial information does not exist in the MELCOR equations. This mismatch between the loss formulation and the input representation leads to degraded predictive performance.

To address this limitation, a lumped-parameter PINN (LP-PINN) architecture is developed, as illustrated in Fig. 10, that strictly follows the system-level representation employed in MELCOR. In the proposed LP-PINN, the input is defined solely as the one-dimensional time variable t, reflecting the fact that MELCOR does not require explicit spatial coordinates. Consequently, the outputs are not parameterized by a spatial index i; instead, the LP-PINN is formulated to directly predict the complete set of nodal variables for all CVs and FLs, allowing all nodes to be trained jointly without introducing artificial spatial information.

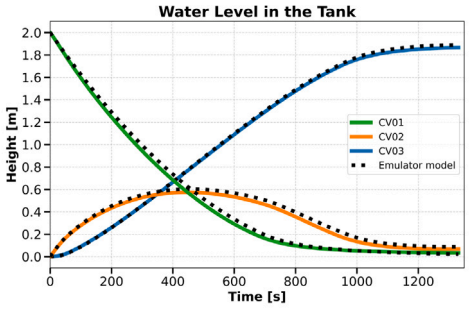
In addition, the output dimension of the network is defined as $2N_i - 1$, where N_i denotes the number of CVs. Specifically, the network outputs N_i variables corresponding to the water levels of the CVs, together with $N_i - 1$ variables representing the velocities between



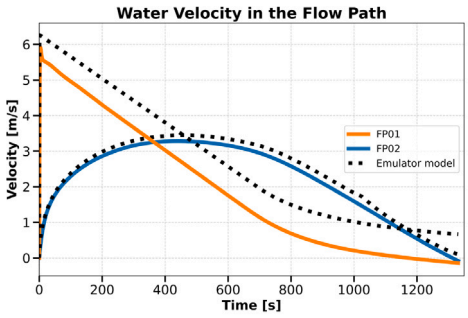
(a) Height profile when $N_i = 2$



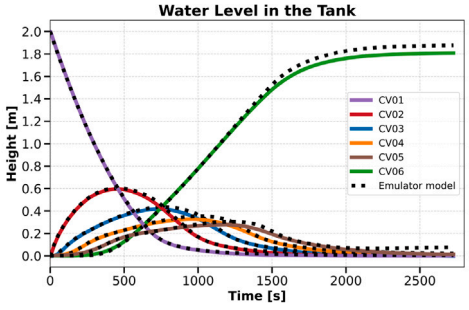
(b) Velocity profile when $N_i = 2$



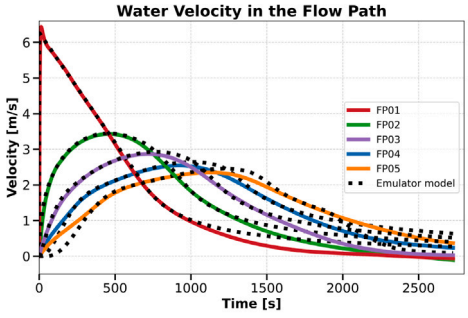
(c) Height profile when $N_i = 3$



(d) Velocity profile when $N_i = 3$

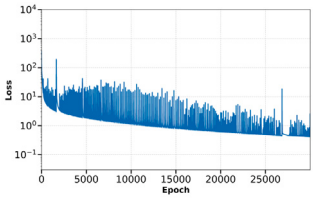


(e) Height profile when $N_i = 6$

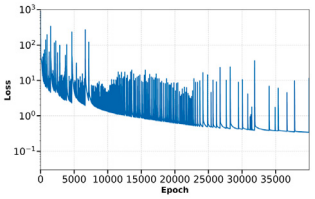


(f) Velocity profile when $N_i = 6$

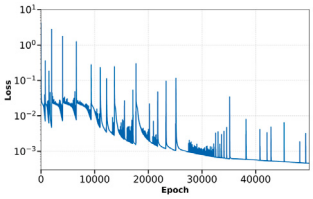
Fig. 8. Comparison of water height (left) and velocity (right) for $N_i = 2$, 3, and 6 CV cases. Dotted lines represent the emulator model results, and solid lines indicate the PINN predictions.



(a) Training loss at $N_i = 2$



(b) Training loss at $N_i = 3$



(c) Training loss at $N_i = 6$

Fig. 9. Training loss histories for $N_i = 2$, 3, and 6 CV cases. The total loss is plotted over the training epochs, showing no meaningful convergence in any case.

adjacent CVs, as illustrated in Fig. 1. This formulation enables the LP-PINN to represent the coupled system dynamics in a manner that is fully consistent with the lumped-parameter structure of MELCOR, while avoiding the need for artificial spatial inputs. The complete training algorithm is also provided in Algorithm 3. Since the material m and phase φ do not change in this scenario, they are excluded for simplicity.

3.6. LP-PINN module verification

This section evaluates the performance and convergence of the LP-PINN under the same settings and criteria as in Section 3.3. Fig. 11 summarizes the comparison results of the predicted system responses for the cases with $N_i = 2$, 3, and 6 CVs. As shown in the Figure, the LP-PINN fails to adequately capture the transient behavior in all

Table 8
Comparison of MAE, MSE, and the number of trainable parameters between the vanilla PINN, the LP-PINN, and the NA-PINN.

N_i	Architecture	MAE (H)	MSE (H)	MAE (V)	MSE (V)	Parameters
2	PINN	0.005218	0.000032	0.225312	0.144070	334,466
2	LP-PINN	1.3281475	2.435083	3.252238	13.065767	334,467
2	NA-PINN	0.001222	2.113807e−6	0.008827	2.39661e−04	337,907
3	PINN	0.021679	0.000712	0.470970	0.321667	593,410
3	LP-PINN	0.661866	0.806748	2.519508	8.475054	593,925
3	NA-PINN	0.001280	2.147004e−6	0.004914	1.122552e−4	579,845
6	PINN	0.013875	0.000397	0.174725	0.062661	1,223,970
6	LP-PINN	0.315291	0.332821	1.250000	2.685334	1,226,923
6	NA-PINN	0.001118	3.613825e−6	0.004344	5.66337e−5	1,275,659

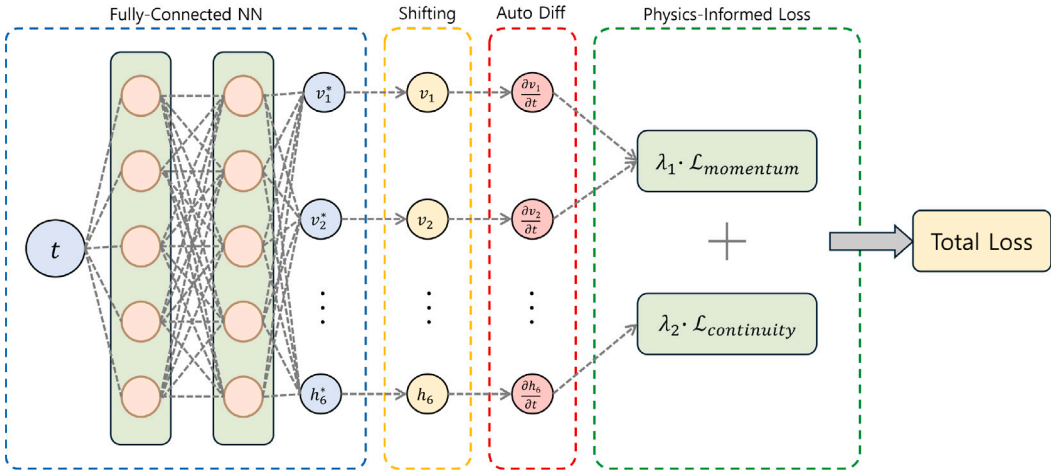


Fig. 10. Architecture of the LP-PINN model.

cases, exhibiting significant discrepancies compared to the reference solutions. The errors remain substantial regardless of the system complexity, indicating a fundamental limitation in the PINN's ability to learn the governing dynamics even in the simplest case, $N_i = 2$.

The MAE and MSE values in Table 8 indicate the poor accuracy of the LP-PINN across all cases. A closer inspection of the LP-PINN results indicates that, in certain cases, the predicted solutions remain nearly constant over time, failing to capture the expected transient behavior of the system. This issue was observed consistently across all tested configurations, where the network does not exhibit meaningful variations despite changing input conditions. These results indicate that the LP-PINN fails to adequately capture the overall system behavior, likely due to its single-network architecture, which must simultaneously predict multiple interdependent variables as in Fig. 10. As a result, the network fails to establish an appropriate mapping between inputs and outputs, leading to a trivial solution where the predicted state remains unchanged.

In addition, Fig. 12 presents the loss histories during training for each case. For $N_i = 2$, the loss curve shows no noticeable decrease, indicating a clear failure to converge. In contrast, the $N_i = 3$ and $N_i = 6$ cases exhibit a reduction in loss by approximately two orders of magnitude from the initial epochs. However, these cases appear to converge to local minima, as the solutions fail to reflect the expected physical behavior. This limitation is likely due to the increased number of PDEs—five for $N_i = 3$ and eleven for $N_i = 6$—which poses a significant challenge for the LP-PINN architecture based on a single neural network. These results highlight the difficulty of solving highly coupled multi-equation systems using the LP-PINN approach.

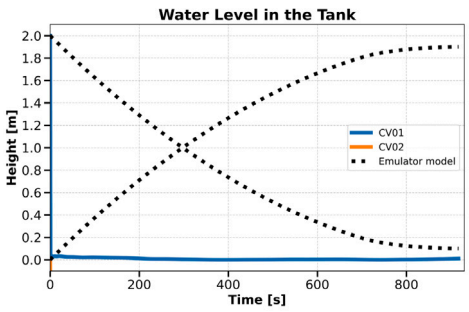
These results demonstrate that the LP-PINN formulation exhibits clear limitations in modeling the transient TH behavior of the system. One possible contributing factor is the use of a single neural network to simultaneously predict multiple governing PDEs, which can give rise to gradient conflicts during training. However, it remains unclear whether the observed performance limitations stem from fundamental challenges associated with applying PINN frameworks to lumped-parameter formulations, or from gradient conflicts inherent to multi-equation learning. Therefore, it is necessary to develop a model that explicitly addresses gradient conflict issues and to compare its performance against LP-PINN in order to isolate the dominant source of error.

4. Node-assigned PINN for thermal-hydraulics analysis code

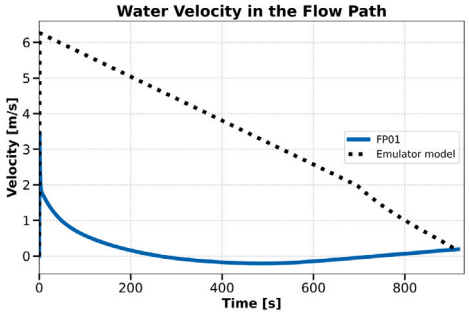
4.1. Concept and architecture

The LP-PINN architecture exhibits significant limitations when applied to transient TH simulations. These limitations arise because a single neural network is used to predict multiple PDEs simultaneously, which can lead to gradient conflicts and convergence issues due to multiple local minima and differences in the output variable scales (Zou and Karniadakis, 2023). Moreover, strong interdependencies between CVs and FLs can result in convergence failure and large prediction errors. We concluded that a novel PINN architecture suitable for CV-based system codes needs to be developed.

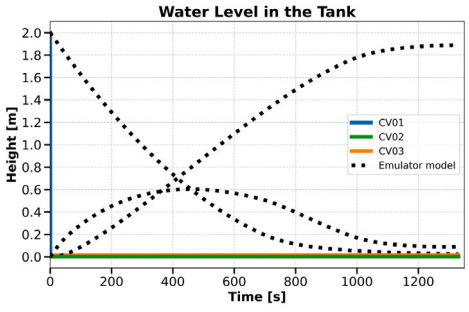
To address these challenges, we develop NA-PINN, which aligns the PINN representation with MELCOR's CV/FL nodalization. In this framework, each output variable, including water level and velocity, is assigned an independent neural network. In other words, separate networks were assigned for modeling individual nodes of CVs and FLs. It is named node, not CV, because it also includes FL objects. Although the networks are structurally independent, they are trained simultaneously via a shared loss function that enforces the governing physical laws, ensuring consistency among variables during training. A schematic of the NA-PINN architecture is shown in Fig. 13. The complete training



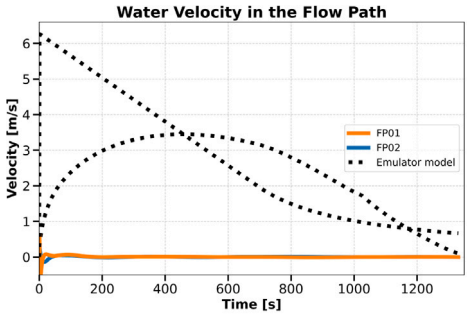
(a) Height profile when $N_i = 2$



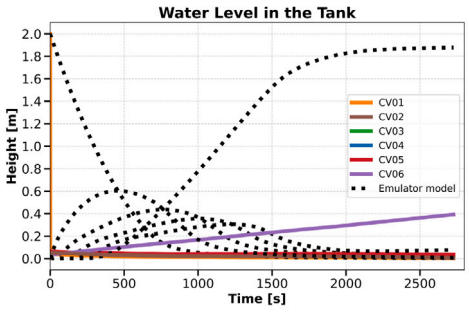
(b) Velocity profile when $N_i = 2$



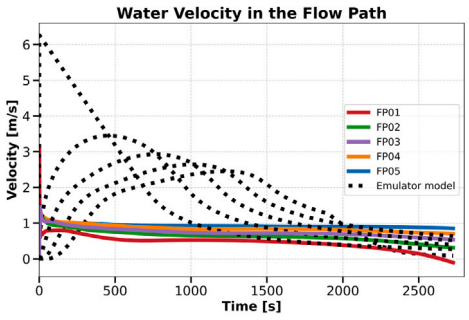
(c) Height profile when $N_i = 3$



(d) Velocity profile when $N_i = 3$

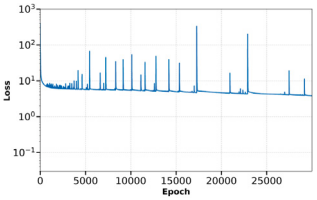


(e) Height profile when $N_i = 6$

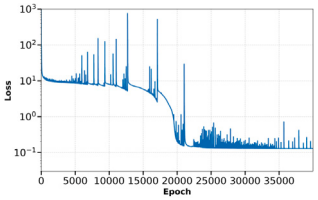


(f) Velocity profile when $N_i = 6$

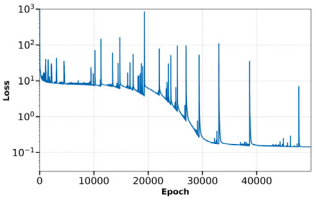
Fig. 11. Comparison of water height (left) and velocity (right) for $N_i = 2$, 3, and 6 CV cases using the LP-PINN. Dotted lines represent the Python-based emulator model results, and solid lines indicate the LP-PINN predictions.



(a) Training loss at $N_i = 2$



(b) Training loss at $N_i = 3$



(c) Training loss at $N_i = 6$

Fig. 12. Training loss histories of the LP-PINN module for $N_i = 2$, 3, and 6 CV cases.

algorithm is also provided in Algorithm 4. The overlapping parts with LP-PINN have been omitted for brevity, and it should be noted that a neural network and trainable parameters have been specified for each variable.

In the LP-PINN, only temporal information is used as an input, a single neural network must infer spatial–temporal functions although spatially discrete. By contrast, in the NA-PINN architecture each sub-network is associated with a specific spatial node, so that, given only the temporal input, it predicts a purely temporal function. We attribute the superior performance of NA-PINN to the deliberate exclusion of spatial information from both its inputs and outputs, which simplifies the learning task.

4.2. Node-assigned PINN module verification

The architecture of NA-PINN addresses the limitations associated with using a single network as in Section 3.4 by assigning a individual neural network to each output variable. This approach eliminates

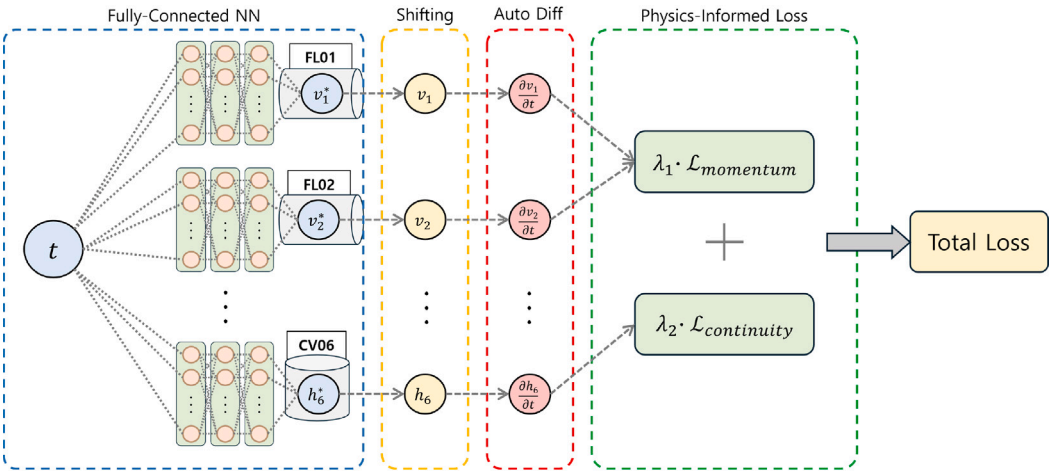


Fig. 13. Architecture of the NA-PINN. Each network is responsible for predicting one CV or FL variable, as illustrated inside each block.

Algorithm 3 Training procedure of LP-PINN

Require: Time domain $[0, T]$, number of collocation points N_t, initial condition $\mathbf{u}_0$
Require: Trainable parameters $\boldsymbol{\theta}$, number of tanks n, number of epochs N_{epoch}, learning rate scheduler η_{sched}
Initialize neural network parameters $\boldsymbol{\theta}$
Apply shifting method:

$$\mathbf{u}(t) = \mathbf{u}_{NN}(t) - \mathbf{u}_{NN}(0) + \mathbf{u}_0$$

for each epoch $k = 1, \ldots, N_{\text{epoch}}$ **do**
  Sample N_t collocation points $t \in [0, T]$ using uniform (equally spaced) discretization
  Normalize time input using Min-Max scaling:

$$t' = \frac{t - t_{\min}}{t_{\max} - t_{\min}}$$

  Predict $\{\hat{v}_j(t')\}_{j=1}^{n-1}$ and $\{\hat{H}_i(t')\}_{i=1}^{n}$ using the neural network $\mathbf{u}(t)$
  Compute temporal derivatives via automatic differentiation:

$$\frac{\partial \hat{v}_j}{\partial t}, \quad \frac{\partial \hat{H}_i}{\partial t}$$

  Compute void fraction at the inlet of pipe j (connecting donor tank j):

$$\alpha_j = \begin{cases} 1 - \dfrac{\hat{H}_j}{0.2}, & 0 \le \hat{H}_j < 0.2 \\ 0, & \text{otherwise} \end{cases}$$

  Compute height difference for each pipe j (connecting donor tank j and receiver tank $j+1$):

$$\Delta z_j = \begin{cases} \hat{H}_j - \hat{H}_{j+1} + 1.8, & \hat{H}_j < 0.2 \wedge \hat{H}_{j+1} \ge 1.8 \\ \hat{H}_j, & \text{otherwise} \end{cases}$$

  Compute momentum residuals for each pipe j:

$$\mathcal{R}_{\text{momentum},j} = L \cdot \frac{\partial \hat{v}_j}{\partial t} - g \cdot \Delta z_j + \frac{K^*}{2} |\hat{v}_j| \hat{v}_j$$

  Compute continuity residuals for each tank i (considering the upstream connected pipe i):

$$\mathcal{R}_{\text{continuity},i} = \rho A_t \cdot \frac{\partial \hat{H}_i}{\partial t} + \begin{cases} \rho A_p F \cdot \hat{v}_1 (1 - \alpha_1), & i = 1 \\ \rho A_p F \cdot \left[\hat{v}_i (1 - \alpha_i) - \hat{v}_{i-1} (1 - \alpha_{i-1})\right], & 2 \le i \le n-1 \\ -\rho A_p F \cdot \hat{v}_{n-1} (1 - \alpha_{n-1}), & i = n \end{cases}$$

  Compute total loss:

$$\mathcal{L}_{\text{PDE}} = \lambda_1 \cdot \frac{1}{N_t} \sum_{j=1}^{n-1} \mathcal{R}^2_{\text{momentum},j} + \lambda_2 \cdot \frac{1}{N_t} \sum_{i=1}^{n} \mathcal{R}^2_{\text{continuity},i}$$

  Update parameters using a scheduled learning rate:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_{\text{sched}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{PDE}}$$

end for
return trained model $\mathbf{u}(t)$

Algorithm 4 Training procedure of NA-PINN

Require: Time domain $[0, T]$, number of collocation points N_t, initial condition $\mathbf{U}_0$
Require: Trainable parameter sets $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_{\hat{v}_1}, \ldots, \boldsymbol{\theta}_{\hat{v}_{n-1}}, \boldsymbol{\theta}_{\hat{H}_1}, \ldots, \boldsymbol{\theta}_{\hat{H}_n}\}$, number of tanks n, number of epochs N_{epoch}, learning rate scheduler η_{sched}
Initialize neural network parameters $\boldsymbol{\Theta}$
Apply shifting method $\mathbf{U} = \{\mathbf{u}_{\hat{v}_1}, \ldots, \mathbf{u}_{\hat{v}_{n-1}}, \mathbf{u}_{\hat{H}_1}, \ldots, \mathbf{u}_{\hat{H}_n}\}$:

$$\mathbf{U}(t) = \mathbf{U}_{NN}(t) - \mathbf{U}_{NN}(0) + \mathbf{U}_0$$

for each epoch $k = 1, \ldots, N_{\text{epoch}}$ **do**
  Predict $\{\hat{v}_j(t')\}_{j=1}^{n-1}$ and $\{\hat{H}_i(t')\}_{i=1}^{n}$ using each neural network $\mathbf{U}$
  Compute temporal derivatives via automatic differentiation:

$$\frac{\partial \hat{v}_j}{\partial t}, \quad \frac{\partial \hat{H}_i}{\partial t}$$

  Compute momentum residuals for each pipe j:

$$\mathcal{R}_{\text{momentum},j} = L \cdot \frac{\partial \hat{v}_j}{\partial t} - g \cdot \Delta z_j + \frac{K^*}{2} |\hat{v}_j| \hat{v}_j$$

  Compute continuity residuals for each tank i (considering the upstream connected pipe i):

$$\mathcal{R}_{\text{continuity},i} = \rho A_t \cdot \frac{\partial \hat{H}_i}{\partial t} + \begin{cases} \rho A_p F \cdot \hat{v}_1 (1 - \alpha_1), & i = 1 \\ \rho A_p F \cdot \left[\hat{v}_i (1 - \alpha_i) - \hat{v}_{i-1} (1 - \alpha_{i-1})\right], & 2 \le i \le n-1 \\ -\rho A_p F \cdot \hat{v}_{n-1} (1 - \alpha_{n-1}), & i = n \end{cases}$$

  Compute total loss:

$$\mathcal{L}_{\text{PDE}} = \lambda_1 \cdot \frac{1}{N_t} \sum_{j=1}^{n-1} \mathcal{R}^2_{\text{momentum},j} + \lambda_2 \cdot \frac{1}{N_t} \sum_{i=1}^{n} \mathcal{R}^2_{\text{continuity},i}$$

  Update parameters using a scheduled learning rate:

$$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \eta_{\text{sched}} \nabla_{\boldsymbol{\Theta}} \mathcal{L}_{\text{PDE}}$$

end for
return trained model $\mathbf{u}(t)$

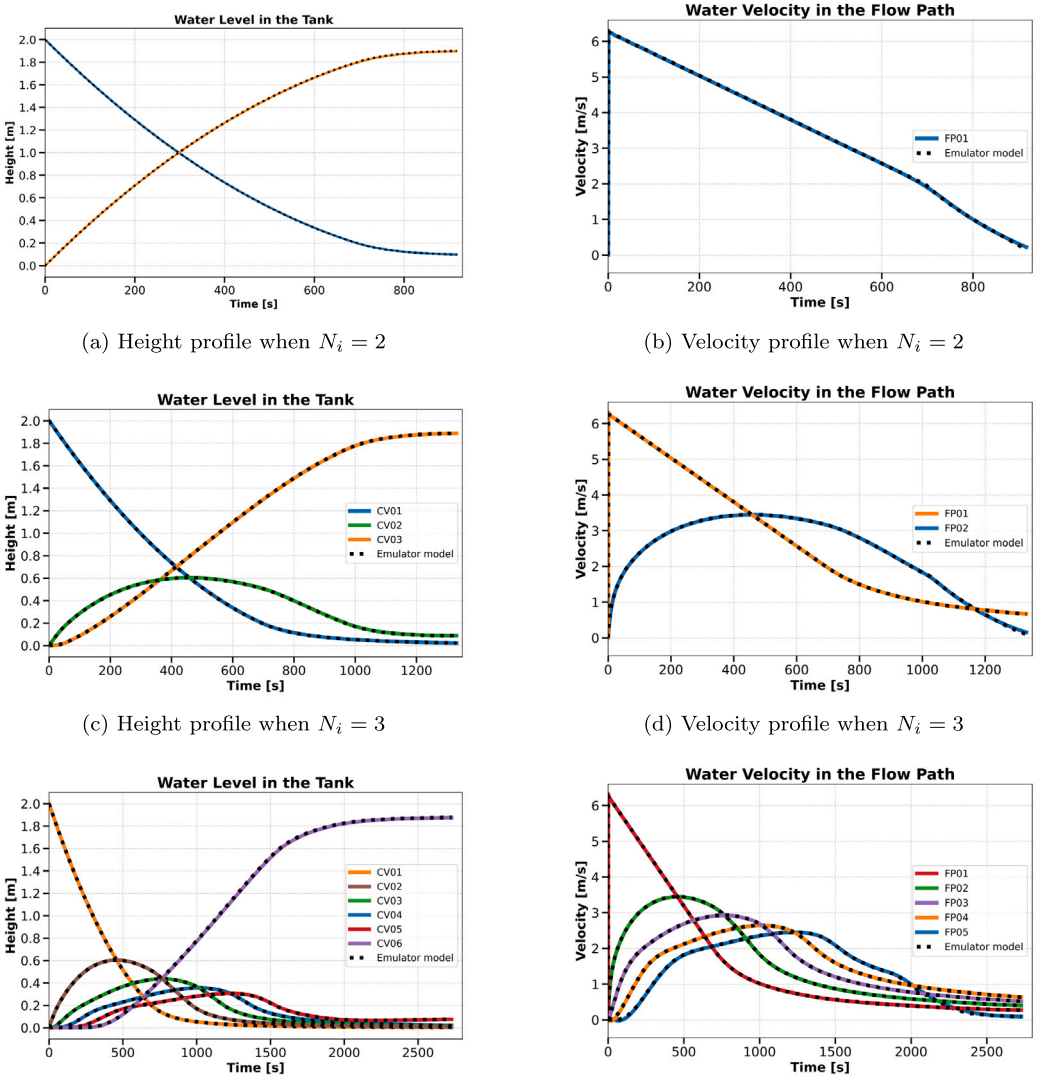
cross-output interference and enables each network to specialize in its respective variable. Moreover, the shared loss function ensures that the interdependence among variables governed by physical laws is effectively captured during training (Zhang and Li, 2023). Apart from this structural modification, the training settings, including the optimizer, activation function, and learning rate, are consistent with those used in the vanilla PINN.

The performance of the NA-PINN module is validated through numerical experiments conducted on systems with $N_i = 2, 3$, and 6 CVs—same as vanilla PINN. To ensure a fair comparison, the total number of trainable parameters in NA-PINN is kept similar to that of the vanilla PINN. The hyperparameters applied in each case are summarized in Table 9, where the number of networks in each case is set equal to the number of output variables to be predicted—that is, the number of PDEs associated with water levels and velocities.

Fig. 14 presents their results, showing that NA-PINN closely follows the reference solutions obtained from the Python-based emulator model. In contrast to the LP-PINN, which struggled with learning meaningful dynamics, NA-PINN accurately captures transient variations. This improvement can be attributed to its decoupled network architecture,

Table 9
Hyperparameters used for the NA-PINN module.

N_i	Networks	End time	Collocation points	Epochs	Hidden layers × nodes
2	3	1000	2500	30 000	8 × 128
3	5	1400	3000	40 000	8 × 128
6	11	2800	6000	50 000	8 × 128



(a) Height profile when $N_i = 2$

(b) Velocity profile when $N_i = 2$

(c) Height profile when $N_i = 3$

(d) Velocity profile when $N_i = 3$

(e) Height profile when $N_i = 6$

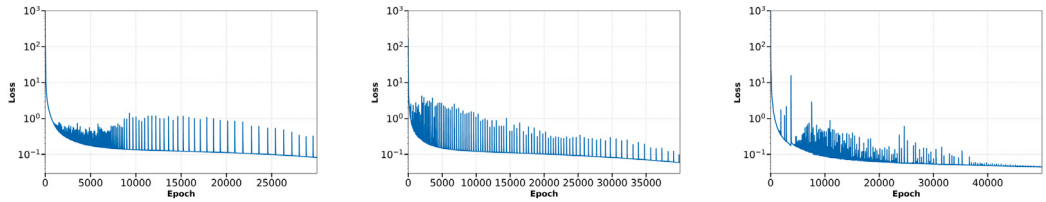
(f) Velocity profile when $N_i = 6$

Fig. 14. Comparison of water height (left) and velocity (right) for $N_i = 2$, 3, and 6 CV cases using the NA-PINN. Dotted lines represent the Python-based emulator model results, and solid lines indicate the NA-PINN predictions.

which eliminates interference between outputs and allows each neural network to specialize in learning its assigned variable.

Table 8 quantitatively confirms that NA-PINN significantly reduces MAE and MSE across all test cases. Unlike the LP-PINN, which failed to learn transient behavior, NA-PINN effectively models dynamic variations by leveraging its independent network structure. This structural improvement enables more stable gradient updates, leading to more accurate predictions in complex simulations.

Additionally, Fig. 15 demonstrates the robust convergence of the NA-PINN architecture, showing a consistent reduction in loss across all cases—a marked improvement over the stagnation observed in Fig. 9. To mitigate optimization instabilities in PINN training that can arise from stiff gradient-flow dynamics, we employed an exponential learning-rate decay schedule. Small late-stage oscillations in the loss for the $N_i = 2$ and $N_i = 3$ cases can occur when fewer epochs result in less accumulated decay (i.e., a relatively higher final learning rate), leading to mild overshooting and fluctuations around a low-loss neighborhood. Such oscillations are a known behavior in gradient-based training under non-vanishing step sizes and do not necessarily indicate loss of convergence (Wang et al., 2021). Importantly, the final loss values remain at very low magnitudes and the predicted transients stay accurate, which is consistent with convergence to a stable, low-residual solution.



(a) Training loss at $N_i = 2$ (b) Training loss at $N_i = 3$ (c) Training loss at $N_i = 6$

Fig. 15. Training loss histories of the NA-PINN module for $N_i = 2$, 3, and 6 CV cases.

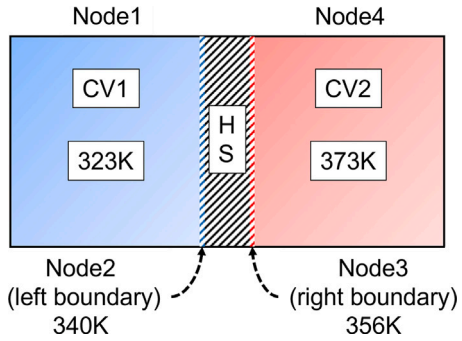


Fig. 16. Schematic of the two-CV system coupled by a heat structure via interfacial convection and internal conduction. The global structure defines CV1 (Node 1) and CV2 (Node 4), as well as the left (Node 2) and right (Node 3) boundary interfaces of the HS.

The computational cost of NA-PINN is assessed in terms of the total training time and the inference time per transient. The total training time is 624, 1383, and 3466 s for $N_i = 2$, $N_i = 3$, and $N_i = 6$, respectively, and the corresponding inference times are 0.01067, 0.02035, and 0.01750 s. For the $N_i = 6$ configuration, the emulator requires 0.03688 s to generate the same transient scenario, whereas NA-PINN produces the result in 0.01750 s, corresponding to an inference speedup of approximately 2×. This pronounced runtime gap supports extending the proposed approach to surrogate modeling, where rapid repeated evaluations are essential. As accident scenarios become more complex and the emulator runtime increases accordingly, the potential acceleration offered by surrogate-based inference is expected to become even more substantial.

4.3. Extension of NA-PINN to Heat Structure coupling

This section extends the NA-PINN formulation validated in Section 4.2 to a coupled thermal problem that explicitly includes a Heat Structure (HS). The target scenario consists of two CVs thermally coupled through an intervening solid, where heat exchange at each CV–HS interface occurs via convection and heat transport within the HS is governed by conduction. The objective of this section is to evaluate whether NA-PINN can accurately predict the temperatures of both the CVs and the HS in a thermally coupled CV–HS heat-transfer scenario.

The coupled CV–HS–CV configuration and the adopted nodalization are shown in Fig. 16. To ensure consistency between the physical model and the NA-PINN architecture, four thermal nodes are defined on the CV and HS side of the two interfaces: the left CV temperature T_{CV1}, the left HS boundary temperature T_{HS1}, the right HS boundary temperature T_{HS2}, and the right CV temperature T_{CV2}. The initial temperatures of the left and right CVs are set to 323 K and 373 K, respectively, while the corresponding HS boundary interfaces are initialized at 340 K (left) and 356 K (right). The transient response is simulated over a long time horizon, $t \in [0,\ 65,000]$.

The governing equations comprise HS conduction and interface convection. Heat conduction in the HS is described by the transient one-dimensional conduction equation, given in Eq. (28):

$$C_p \frac{\partial T}{\partial t} = \frac{1}{A} \frac{\partial}{\partial x}\left(kA \frac{\partial T}{\partial x}\right). \tag{28}$$

Here, C_p denotes the volumetric heat capacity, T the temperature, A the heat-transfer area, and k the thermal conductivity.

At the CV–HS interfaces, convective heat transfer is modeled by Newton's law of cooling. The left and right interface heat transfer relations are written in Eqs. (29) and (30), respectively:

$$q_L = h_L A_L \left(T_{\mathrm{HS1}} - T_{\mathrm{CV1}}\right), \tag{29}$$

$$q_R = h_R A_R \left(T_{\mathrm{HS2}} - T_{\mathrm{CV2}}\right), \tag{30}$$

where q (i.e. q_L and q_R) denotes the convective heat transfer rate, h the convective heat transfer coefficient, and T the temperature at the corresponding node.

As a reference for validating NA-PINN, we implement a Python emulator of the coupled conduction–convection model following the procedure in Section 2. The emulator provides the reference solution, while the remainder of this section derives the node-wise residuals and corresponding loss terms from the governing equations.

Following the NA-PINN architecture in Fig. 13, the present HS-coupled system is modeled using node-specific networks, with one dedicated subnetwork assigned to each thermal node. Accordingly, because the coupled CV–HS–CV configuration is represented by four thermal nodes $\{T_{\mathrm{CV1}}, T_{\mathrm{HS1}}, T_{\mathrm{HS2}}, T_{\mathrm{CV2}}\}$, the NA-PINN model employs four subnetworks to predict the corresponding nodal temperatures.

Using the nodal representation, we define four residual equations—one for each node. Let $C_{p,\mathrm{CV1}}$ and $C_{p,\mathrm{CV2}}$ denote the volumetric heat capacitances of the left and right CVs, and $C_{p,\mathrm{HS1}}$ and $C_{p,\mathrm{HS2}}$ those of the HS boundary nodes. The left CV residual is given in Eq. (31):

$$\mathcal{R}_{\mathrm{CV1}} = V_{\mathrm{CV1}} C_{p,\mathrm{CV1}} \frac{dT_{\mathrm{cv1}}}{dt} - h_L A_L \left(T_{\mathrm{HS1}} - T_{\mathrm{CV1}}\right). \tag{31}$$

Similarly, the right CV residual is defined by Eq. (32):

$$\mathcal{R}_{\mathrm{CV2}} = V_{\mathrm{CV2}} C_{p,\mathrm{CV2}} \frac{dT_{\mathrm{CV2}}}{dt} - h_R A_R \left(T_{\mathrm{HS2}} - T_{\mathrm{CV2}}\right). \tag{32}$$

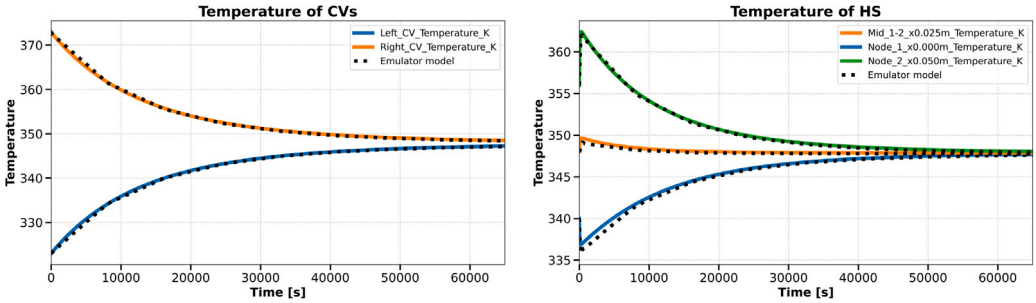
For the HS boundary nodes, the residual must include both the internal conductive exchange and the interface convection. The left HS boundary residual is written in Eq. (33):

$$\mathcal{R}_{\mathrm{HS1}} = V_{\mathrm{HS1}} C_{p,\mathrm{HS1}} \frac{dT_{\mathrm{HS1}}}{dt} - \frac{k_{\mathrm{HS1}} A_L}{dx}\left(T_{\mathrm{HS2}} - T_{\mathrm{HS1}}\right) - h_L A_L \left(T_{\mathrm{CV1}} - T_{\mathrm{HS1}}\right), \tag{33}$$

and the right HS boundary residual is written in Eq. (34):

$$\mathcal{R}_{\mathrm{HS2}} = V_{\mathrm{HS2}} C_{p,\mathrm{HS2}} \frac{dT_{\mathrm{HS2}}}{dt} - \frac{k_{\mathrm{HS2}} A_R}{dx}\left(T_{\mathrm{HS1}} - T_{\mathrm{HS2}}\right) - h_R A_R \left(T_{\mathrm{CV2}} - T_{\mathrm{HS2}}\right). \tag{34}$$

Here, dx is set to the HS thickness (i.e., the distance between the HS boundary nodes). Since spatial information is not explicitly provided to



(a) Temperature of CV nodes (b) Temperature of HS boundary nodes

Fig. 17. Long-horizon temperature histories. Dotted lines denote the reference emulator, and solid lines denote NA-PINN.

the NA-PINN as an input, the HS conduction term is expressed using the nodal temperature difference over dx.

These four residuals explicitly preserve the physical meaning of each node's energy balance. For the CV nodes, the residuals include transient storage and interfacial convection; for the HS boundary nodes, they include transient storage, internal conduction, and interfacial convection.

Based on the residual definitions, we construct the physics-informed training objective in a compact node-wise form. For a generic node residual R_{node}, the node loss is defined as the mean-squared residual evaluated at temporal collocation points $\{t_k\}_{k=1}^{N_t}$, as written in Eq. (35):

$$\mathcal{L}_{\text{node}} = \frac{1}{N_t}\sum_{k=1}^{N_t}\left(R_{\text{node}}(t_k)\right)^2, \qquad \text{node} \in \{\text{CV1}, \text{HS1}, \text{HS2}, \text{CV2}\}. \tag{35}$$

The total loss is then formed by summing the four node losses, as shown in Eq. (36):

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CV1}} + \mathcal{L}_{\text{HS1}} + \mathcal{L}_{\text{HS2}} + \mathcal{L}_{\text{CV2}} \tag{36}$$

By summing the node-wise losses, the NA-PINN model enforces the energy-balance residuals at all four nodes. The coupled conduction–convection residuals link the subnetworks and preserve the one-network-per-node design principle. Building on this formulation, the next section evaluates the HS-coupled NA-PINN on the coupled CV–HS transient problem and validates the predicted nodal temperatures against the reference emulator.

4.4. NA-PINN verification for the coupled CV–HS transient

We now present verification results for the HS-coupled NA-PINN on the coupled CV–HS–CV transient heat-transfer problem. The evaluation focuses on the temperature evolution at the CV nodes and HS boundary nodes over $T_{\text{end}} = 65,000$ s.

Because the temporal domain is extensive, direct training on a single continuous interval can lead to practical optimization and stability difficulties. To address this issue while leaving the governing equations and node-wise physics constraints unchanged, we employ a time-marching training strategy (Chen et al., 2024). The full interval $t \in [0, T_{\text{end}}]$ is partitioned into consecutive time windows, training proceeds sequentially from the first window to the last, and the predicted terminal state of one window is propagated as the initial condition for the next. Importantly, this procedure is introduced purely to facilitate learning over a long time span; it does not alter the residual definitions or the physical coupling mechanisms established in Section 4.3.

Fig. 17 compares the temperature histories predicted by NA-PINN with those produced by the reference emulator for (a) the CV nodes and (b) the HS boundary nodes. In Fig. 17(a), T_{CV1} and T_{CV2} relax from the prescribed initial thermal imbalance toward equilibrium as heat is transferred through the HS. In Fig. 17(b), T_{HS1} and T_{HS2} exhibit consistent long-term behavior and reflect the combined effects of interfacial convection and internal conduction within the solid. Across all four nodes, NA-PINN predictions agree closely with the reference trajectories throughout the simulation, indicating that the coupled conduction–convection dynamics are accurately captured and that the HS–CV coupling is preserved under the one-network-per-node modeling strategy.

Fig. 18 reports the total training-loss evolution under time marching. Within each time window, the loss decreases as the node-wise residual constraints are optimized, while transitions between windows introduce discrete shifts associated with advancing the training interval and updating the propagated initial condition. The overall decreasing trend demonstrates stable optimization across the sequential windows.

For quantitative assessment, prediction errors are summarized using MAE and MSE as defined earlier in the manuscript. Over the full evaluation horizon, the aggregated CV errors are MAE = 0.1681 and MSE = 0.0565, while the aggregated HS errors are MAE = 0.1618 and MSE = 0.0501. These results confirm that the proposed NA-PINN formulation remains accurate for both CV and HS temperatures in the coupled heat-transfer setting, thereby validating the extension of the NA-PINN framework to HS-coupled thermal problems. Because scaling for variable types is not included in the MAE and MSE calculation in this study, the heat transfer cases involving temperature variable have a higher MAE and MSE scale than tank gravity case.

5. Future work

While the proposed NA-PINN architecture demonstrates improved performance over the vanilla PINN and LP-PINN, several directions remain for future investigation. First, the scenarios used in this study assume a simplified flow pattern along CVs and FLs. Therefore, further validation is needed for more complex and realistic transient scenarios. Second, the current implementation focuses solely on MELCOR's CVH/FL/HS packages. Future work could involve extending the framework to include the fission product module to develop a more comprehensive multi-physics solver. Third, the current model requires retraining whenever the scenario changes, which limits its applicability in real-time or fast-response simulations. This limitation could be addressed by developing a surrogate PINN framework capable of generalizing across a range of accident scenarios after a single training process.

6. Conclusion

Severe accident analysis code enables the investigation on the progression of the accident by modeling the postulated scenarios. One of the most representative code, MELCOR is obtained as a target analysis

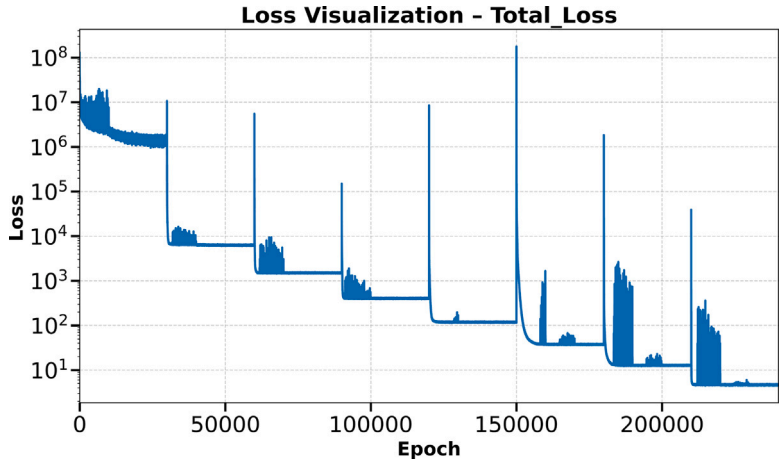


Fig. 18. Training loss history under time marching, showing a staircase-like pattern due to sequential time windows.

system code in this study. MELCOR has a wide range of application from solving numerical solutions for TH and heat structure to modeling complex severe accident progression. However, MELCOR still faces a few challenges such as manually constructing the complex inputs and being unable to solve the multiphysics problem. The goal of this study is to address these system codes difficulties by employing artificial intelligence. As a pioneering study, the CVH/FL package, a simple yet crucial component, was simulated using a Python-based emulator. to thoroughly assess the comprehension of the logic behind the calculation in the package. The comparison results indicated that the system code and the emulator well agreed between the two. With the emulator being developed, reduction in complexity of the loss term for PINN model could be conducted thereby finding the optimized loss equation for PINN model to train.

We proposed a NA-PINN architecture to overcome the limitations of vanilla PINNs in modeling transient TH systems by enabling a lumped-parameter representation while avoiding the gradient conflicts observed in LP-PINN. By assigning independent neural networks to each PDE and enforcing a shared loss function, the proposed method effectively captures the coupled dynamics of CVs and FLs while removing spatial information from its inputs and outputs, which simplifies the learning task into a purely temporal approximation and results in superior performance. Numerical experiments demonstrated that NA-PINN achieved significantly better convergence and accuracy compared to LP-PINN, particularly in systems with multiple interacting PDEs. At $N_i = 2$, the MSE for height decreased from 2.44 to 2.11×10^{-6}, and that for velocity decreased from 0.81 to 2.15×10^{-6}. At $N_i = 3$, the MSE for height decreased from 2.42 to 6.44×10^{-6}, while the velocity MSE decreased from 16.95 to 2.25×10^{-4}. When $N_i = 6$, the MSE for height decreased from 0.33 to 3.61×10^{-6}, and for velocity from 13.43 to 2.83×10^{-4}. These results demonstrate that NA-PINN achieved substantially lower errors than LP-PINN for both height and velocity across all tested scenarios. Furthermore, the NA-PINN achieved notably low MSE values of 0.0565 for the CV and 0.0501 for the HS in the thermal validation case.

Although the current framework requires retraining when applied to new accident scenarios, the gradual loss reduction and physically meaningful predictions highlight the advantages of the architecture. These results highlight the potential of NA-PINN as a scalable and reliable approach that could be extended to complex multi-physics problems in nuclear THs.

Overall, NA-PINN preserves a lumped-parameter representation while resolving gradient conflicts through architectural decomposition, indicating that it is well-suited for integration with system-level TH codes such as MELCOR.

CRediT authorship contribution statement

Jeesuk Shin: Writing – original draft, Software, Methodology, Investigation. **Cheolwoong Kim:** Writing – original draft, Visualization, Software, Investigation. **Sunwoong Yang:** Writing – original draft, Investigation, Conceptualization. **Minseo Lee:** Investigation, Conceptualization. **Donggyun Seo:** Writing – review & editing, Software. **Sung Joong Kim:** Resources, Conceptualization. **Joongoo Jeon:** Writing – original draft, Software, Investigation, Funding acquisition, Conceptualization.

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 supported by the Nuclear Safety Research Program through the Korea Foundation of Nuclear Safety (KoFONS) and the (Regulatory Research Management Agency for SMRs (RMAS) using the financial resource granted by the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea (Nos. RS-2024-00403364 and RS-2024-00509653).

Appendix. Training record tables for LP-PINN and NA-PINN

This appendix summarizes the training configurations and the resulting error metrics for the LP-PINN and NA-PINN. The MAE and MSE follow the same definitions used in the main text. For compact reporting, the tables list the key training settings together with the final MAE/MSE outcomes.

A.1. LP-PINN

See Table A.10.

A.2. NA-PINN

See Table A.11.

Table A.10
Hyperparameter and performance summary for LP-PINN.

End Time	Collocation	Epochs	Node	Hidden Layers	Parameters	Activation	Optimizer	Hard	Learning Rate	Weight	MAE	MSE
1000	2500	30 000	128	8	337 907	ELU	Adam	Shifting	10^{-4}	[1, 1]	0.017265	0.008639
1000	2500	30 000	128	8	337 907	ReLU	Adam	Shifting	10^{-4}	[1, 1]	0.784249	1.188567
1000	2500	30 000	128	8	337 907	Tanh	Adam	Shifting	10^{-4}	[1, 1]	0.025540	0.007674
1000	2500	30 000	128	8	337 907	ELU	Adam	Shifting	10^{-4}	[1, 0.001]	0.003757	0.000081
1000	2500	30 000	128	8	337 907	ReLU	Adam	Shifting	10^{-4}	[1, 0.001]	0.378512	0.332556
1000	2500	30 000	128	8	337 907	Tanh	Adam	Shifting	10^{-4}	[1, 0.001]	0.004544	0.000105
1000	2500	30 000	128	8	337 907	ELU	Adam	Shifting	10^{-3}	[1, 0.001]	0.003651	0.000136
1000	2500	30 000	128	8	337 907	ReLU	Adam	Shifting	10^{-3}	[1, 0.001]	0.375381	0.329195
1000	2500	30 000	128	8	337 907	Tanh	Adam	Shifting	10^{-3}	[1, 0.001]	0.006282	0.000153
1000	2500	30 000	128	8	337 907	ELU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.005136	0.000120
1000	2500	30 000	128	8	337 907	ReLU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.377284	0.331008
1000	2500	30 000	128	8	337 907	Tanh	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.038111	0.002707
1000	2500	30 000	128	8	337 907	ELU	Adam	HardBC	10^{-4}	[1, 0.001]	0.003522	0.000083
1000	2500	30 000	128	8	337 907	ReLU	Adam	HardBC	10^{-4}	[1, 0.001]	0.030188	0.003987
1000	2500	30 000	128	8	337 907	Tanh	Adam	HardBC	10^{-4}	[1, 0.001]	0.151350	0.279221
1400	3000	40 000	128	8	602 067	ELU	Adam	Shifting	10^{-4}	[1, 1]	0.013492	0.003455
1400	3000	40 000	128	8	602 067	ReLU	Adam	Shifting	10^{-4}	[1, 1]	0.874493	1.352109
1400	3000	40 000	128	8	602 067	Tanh	Adam	Shifting	10^{-4}	[1, 1]	0.018139	0.002952
1400	3000	40 000	128	8	602 067	ELU	Adam	Shifting	10^{-4}	[1, 0.001]	0.002734	0.000046
1400	3000	40 000	128	8	602 067	ReLU	Adam	Shifting	10^{-4}	[1, 0.001]	0.265491	0.153057
1400	3000	40 000	128	8	602 067	Tanh	Adam	Shifting	10^{-4}	[1, 0.001]	0.004911	0.000084
1400	3000	40 000	128	8	602 067	ELU	Adam	Shifting	10^{-3}	[1, 0.001]	0.002834	0.000040
1400	3000	40 000	128	8	602 067	ReLU	Adam	Shifting	10^{-3}	[1, 0.001]	0.300578	0.180490
1400	3000	40 000	128	8	602 067	Tanh	Adam	Shifting	10^{-3}	[1, 0.001]	0.005398	0.000094
1400	3000	40 000	128	8	602 067	ELU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.003970	0.000063
1400	3000	40 000	128	8	602 067	ReLU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.265350	0.152387
1400	3000	40 000	128	8	602 067	Tanh	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.004810	0.000086
1400	3000	40 000	128	8	602 067	ELU	Adam	HardBC	10^{-4}	[1, 0.001]	0.003345	0.000053
1400	3000	40 000	128	8	602 067	ReLU	Adam	HardBC	10^{-4}	[1, 0.001]	0.036600	0.007399
1400	3000	40 000	128	8	602 067	Tanh	Adam	HardBC	10^{-4}	[1, 0.001]	0.114495	0.162984
2800	6000	50 000	128	8	1 275 659	ELU	Adam	Shifting	10^{-4}	[1, 1]	0.026654	0.003352
2800	6000	50 000	128	8	1 275 659	ReLU	Adam	Shifting	10^{-4}	[1, 1]	0.583278	0.733660
2800	6000	50 000	128	8	1 275 659	Tanh	Adam	Shifting	10^{-4}	[1, 1]	0.020502	0.002127
2800	6000	50 000	128	8	1 275 659	ELU	Adam	Shifting	10^{-4}	[1, 0.001]	0.002584	0.000028
2800	6000	50 000	128	8	1 275 659	ReLU	Adam	Shifting	10^{-4}	[1, 0.001]	0.292425	0.196904
2800	6000	50 000	128	8	1 275 659	Tanh	Adam	Shifting	10^{-4}	[1, 0.001]	0.003266	0.000042
2800	6000	50 000	128	8	1 275 659	ELU	Adam	Shifting	10^{-3}	[1, 0.001]	0.008458	0.000652
2800	6000	50 000	128	8	1 275 659	ReLU	Adam	Shifting	10^{-3}	[1, 0.001]	0.264639	0.162088
2800	6000	50 000	128	8	1 275 659	Tanh	Adam	Shifting	10^{-3}	[1, 0.001]	0.004265	0.000068
2800	6000	50 000	128	8	1 275 659	ELU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.002613	0.000028
2800	6000	50 000	128	8	1 275 659	ReLU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.282002	0.188456
2800	6000	50 000	128	8	1 275 659	Tanh	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.156636	0.057620
2800	6000	50 000	128	8	1 275 659	ELU	Adam	HardBC	10^{-4}	[1, 0.001]	0.002648	0.000032
2800	6000	50 000	128	8	1 275 659	ReLU	Adam	HardBC	10^{-4}	[1, 0.001]	0.064952	0.018942
2800	6000	50 000	128	8	1 275 659	Tanh	Adam	HardBC	10^{-4}	[1, 0.001]	0.120705	0.096197

Table A.11
Hyperparameter and performance summary for NA-PINN.

End Time	Collocation	Epochs	Node	Hidden layers	Parameters	Activation	Optimizer	Hard	Learning rate	Weight	MAE	MSE
1000	2500	30 000	128	8	337 907	ELU	Adam	Shifting	10^{-4}	[1, 1]	0.017265	0.008639
1000	2500	30 000	128	8	337 907	ReLU	Adam	Shifting	10^{-4}	[1, 1]	0.784249	1.188567
1000	2500	30 000	128	8	337 907	Tanh	Adam	Shifting	10^{-4}	[1, 1]	0.025540	0.007674
1000	2500	30 000	128	8	337 907	ELU	Adam	Shifting	10^{-4}	[1, 0.001]	0.003757	0.000081
1000	2500	30 000	128	8	337 907	ReLU	Adam	Shifting	10^{-4}	[1, 0.001]	0.378512	0.332556
1000	2500	30 000	128	8	337 907	Tanh	Adam	Shifting	10^{-4}	[1, 0.001]	0.004544	0.000105
1000	2500	30 000	128	8	337 907	ELU	Adam	Shifting	10^{-3}	[1, 0.001]	0.003651	0.000136
1000	2500	30 000	128	8	337 907	ReLU	Adam	Shifting	10^{-3}	[1, 0.001]	0.375381	0.329195
1000	2500	30 000	128	8	337 907	Tanh	Adam	Shifting	10^{-3}	[1, 0.001]	0.006282	0.000153
1000	2500	30 000	128	8	337 907	ELU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.005136	0.000120
1000	2500	30 000	128	8	337 907	ReLU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.377284	0.331008
1000	2500	30 000	128	8	337 907	Tanh	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.038111	0.002707
1000	2500	30 000	128	8	337 907	ELU	Adam	HardBC	10^{-4}	[1, 0.001]	0.003522	0.000083
1000	2500	30 000	128	8	337 907	ReLU	Adam	HardBC	10^{-4}	[1, 0.001]	0.030188	0.003987
1000	2500	30 000	128	8	337 907	Tanh	Adam	HardBC	10^{-4}	[1, 0.001]	0.151350	0.279221
1400	3000	40 000	128	8	602 067	ELU	Adam	Shifting	10^{-4}	[1, 1]	0.013492	0.003455
1400	3000	40 000	128	8	602 067	ReLU	Adam	Shifting	10^{-4}	[1, 1]	0.874493	1.352109
1400	3000	40 000	128	8	602 067	Tanh	Adam	Shifting	10^{-4}	[1, 1]	0.018139	0.002952
1400	3000	40 000	128	8	602 067	ELU	Adam	Shifting	10^{-4}	[1, 0.001]	0.002734	0.000046
1400	3000	40 000	128	8	602 067	ReLU	Adam	Shifting	10^{-4}	[1, 0.001]	0.265491	0.153057
1400	3000	40 000	128	8	602 067	Tanh	Adam	Shifting	10^{-4}	[1, 0.001]	0.004911	0.000084
1400	3000	40 000	128	8	602 067	ELU	Adam	Shifting	10^{-3}	[1, 0.001]	0.002834	0.000040
1400	3000	40 000	128	8	602 067	ReLU	Adam	Shifting	10^{-3}	[1, 0.001]	0.300578	0.180490
1400	3000	40 000	128	8	602 067	Tanh	Adam	Shifting	10^{-3}	[1, 0.001]	0.005398	0.000094
1400	3000	40 000	128	8	602 067	ELU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.003970	0.000063
1400	3000	40 000	128	8	602 067	ReLU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.265350	0.152387
1400	3000	40 000	128	8	602 067	Tanh	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.004810	0.000086
1400	3000	40 000	128	8	602 067	ELU	Adam	HardBC	10^{-4}	[1, 0.001]	0.003345	0.000053
1400	3000	40 000	128	8	602 067	ReLU	Adam	HardBC	10^{-4}	[1, 0.001]	0.036600	0.007399
1400	3000	40 000	128	8	602 067	Tanh	Adam	HardBC	10^{-4}	[1, 0.001]	0.114495	0.162984
2800	6000	50 000	128	8	1 275 659	ELU	Adam	Shifting	10^{-4}	[1, 1]	0.026654	0.003352
2800	6000	50 000	128	8	1 275 659	ReLU	Adam	Shifting	10^{-4}	[1, 1]	0.583278	0.733660
2800	6000	50 000	128	8	1 275 659	Tanh	Adam	Shifting	10^{-4}	[1, 1]	0.020502	0.002127
2800	6000	50 000	128	8	1 275 659	ELU	Adam	Shifting	10^{-4}	[1, 0.001]	0.002584	0.000028
2800	6000	50 000	128	8	1 275 659	ReLU	Adam	Shifting	10^{-4}	[1, 0.001]	0.292425	0.196904
2800	6000	50 000	128	8	1 275 659	Tanh	Adam	Shifting	10^{-4}	[1, 0.001]	0.003266	0.000042

(*continued on next page*)

Table A.11 (*continued*).

End Time	Collocation	Epochs	Node	Hidden layers	Parameters	Activation	Optimizer	Hard	Learning rate	Weight	MAE	MSE
2800	6000	50 000	128	8	1 275 659	ELU	Adam	Shifting	10^{-3}	[1, 0.001]	0.008458	0.000652
2800	6000	50 000	128	8	1 275 659	ReLU	Adam	Shifting	10^{-3}	[1, 0.001]	0.264639	0.162088
2800	6000	50 000	128	8	1 275 659	Tanh	Adam	Shifting	10^{-3}	[1, 0.001]	0.004265	0.000068
2800	6000	50 000	128	8	1 275 659	ELU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.002613	0.000028
2800	6000	50 000	128	8	1 275 659	ReLU	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.282002	0.188456
2800	6000	50 000	128	8	1 275 659	Tanh	Adam+L-BFGS	Shifting	10^{-4}	[1, 0.001]	0.156636	0.057620
2800	6000	50 000	128	8	1 275 659	ELU	Adam	HardBC	10^{-4}	[1, 0.001]	0.002648	0.000032
2800	6000	50 000	128	8	1 275 659	ReLU	Adam	HardBC	10^{-4}	[1, 0.001]	0.064952	0.018942
2800	6000	50 000	128	8	1 275 659	Tanh	Adam	HardBC	10^{-4}	[1, 0.001]	0.120705	0.096197

Data availability

Data will be made available on request.

References

Abbasi, J., Moseley, B., Kurotori, T., Jagtap, A.D., Kovscek, A.R., Hiorth, A., Andersen, P.Ø., 2025a. History-matching of imbibition flow in fractured porous media using physics-informed neural networks (PINNs). Comput. Methods Appl. Mech. Engrg. 437, 117784.

Abbasi, J., Moseley, B., Kurotori, T., Jagtap, A.D., Kovscek, A.R., Hiorth, A., Andersen, P.Ø., 2025b. History-matching of imbibition flow in fractured porous media using physics-informed neural networks (PINNs). Comput. Methods Appl. Mech. Engrg. 437, 117784.

Antonello, F., Buongiorno, J., Zio, E., 2023. Physics informed neural networks for surrogate modeling of accidental scenarios in nuclear power plants. Nucl. Eng. Technol. 55 (9), 3409–3416. http://dx.doi.org/10.1016/j.net.2023.04.012.

Bacsa, K., Lai, Z., Liu, W., Todd, M., Chatzi, E., 2023. Symplectic encoders for physics-constrained variational dynamics inference. Sci. Rep. 13 (1), 2643.

Baraldi, P., Battistini, C., Antonello, F., Buongiorno, J., Zio, E., 2025. Physics-informed neural networks for the safety analysis of nuclear reactors. Prog. Nucl. Energy 185, 105745. http://dx.doi.org/10.1016/j.pnucene.2025.105745.

Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E., 2021a. Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mech. Sin. 37 (12), 1727–1738.

Cai, S., Wang, Z., Wang, S., Perdikaris, P., Karniadakis, G.E., 2021b. Physics-informed neural networks for heat transfer problems. J. Heat Transf. 143 (6), 060801.

Chae, Y.H., Kim, H., Bang, J., Seong, P.H., 2023. Development of a data-driven simulation framework using physics-informed neural network. Ann. Nucl. Energy 189, 109840.

Chen, Z., Lai, S.-K., Yang, Z., 2024. AT-PINN: Advanced time-marching physics-informed neural network for structural vibration analysis. Thin-Walled Struct. 196, 111423.

Chung, B.D., do Kim, G., Bae, S., Jung, J., 2010. MARS CODE MANUAL VOLUME 1: Code Structure, System Model, and Solution Methods. Korea Atomic Energy Research Institute, Available from KAERI upon request.

Clevert, D.-A., 2015. Fast and accurate deep network learning by exponential linear units (ELUs). arXiv preprint arXiv:1511.07289.

Electric Power Research Institute (EPRI), 2014. Modular accident analysis program (MAAP5) version 5.03 – windows. Available from EPRI upon request.

Gauntt, R.O., et al., 2021. MELCOR Computer Code Manuals, Volume 2: Reference Manual, Version 2.2.9496. Sandia National Laboratories, Albuquerque, NM, SAND2021-0241, available from NRC or Sandia National Labs.

Go, M.-S., Lim, J.H., Lee, S., 2023. Physics-informed neural network-based surrogate model for a virtual thermal sensor with real-time simulation. Int. J. Heat Mass Transfer 214, 124392. http://dx.doi.org/10.1016/j.ijheatmasstransfer.2023.124392, URL https://www.sciencedirect.com/science/article/pii/S0017931023005380.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., Juanes, R., 2021. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. Comput. Methods Appl. Mech. Engrg. 379, 113741.

International Atomic Energy Agency, 2002. Accident Analysis for Nuclear Power Plants. In: IAEA Safety Reports Series, (23), IAEA, Vienna.

International Atomic Energy Agency, 2007. Application of safety analysis codes for nuclear power plants. Tech. Rep. IAEA-TECDOC-1538, IAEA, Vienna, Available at: https://www.iaea.org/publications/7577/.

International Atomic Energy Agency, 2019. Status and Evaluation of Severe Accident Simulation Codes for Water Cooled Reactors. Tech. Rep. IAEA-TECDOC-1872, IAEA, Vienna, Available at: https://www.iaea.org/publications/13438/.

Jeon, J., Lee, J., Kim, S.J., 2022. Finite volume method network for the acceleration of unsteady computational fluid dynamics: Non-reacting and reacting flows. Int. J. Energy Res. 46 (8), 10770–10795.

Jeon, J., Lee, J., Vinuesa, R., Kim, S.J., 2024. Residual-based physics-informed transfer learning: A hybrid method for accelerating long-term CFD simulations via deep learning. Int. J. Heat Mass Transfer 220, 124900.

Khanal, S., Baral, S., Jeon, J., 2025. Comparison of CNN-based deep learning architectures for unsteady cfd acceleration on small datasets. arXiv preprint arXiv:2502.06837.

Lai, Z., Mylonas, C., Nagarajaiah, S., Chatzi, E., 2021. Structural identification with physics-informed neural ordinary differential equations. J. Sound Vib. 508, 116196.

Lee, Y., Song, S.H., Bae, J.Y., Song, K., Seo, M.R., Kim, S.J., Lee, J.I., 2024. Surrogate model for predicting severe accident progression in nuclear power plant using deep learning methods and rolling-window forecast. Ann. Nucl. Energy 208, 1–4. http://dx.doi.org/10.1016/2024/110816.

Li, Y., Liu, T., Xie, Y., 2022. Thermal fluid fields reconstruction for nanofluids convection based on physics-informed deep learning. Sci. Rep. 12 (1), 12567.

Lu, L., Meng, X., Mao, Z., Karniadakis, G.E., 2021a. DeepXDE: A deep learning library for solving differential equations. SIAM Rev. 63 (1), 208–228.

Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., Johnson, S.G., 2021b. Physics-informed neural networks with hard constraints for inverse design. SIAM J. Sci. Comput. 43 (6), B1105–B1132.

Lu, Q., Yuan, Y., Li, F., Yang, B., Li, Z., Ma, Y., Gu, Y., Liu, D., 2021. Prediction method for thermal-hydraulic parameters of nuclear reactor system based on deep learning algorithm. Appl. Therm. Eng. 196, 117272.

Mao, Z., Jagtap, A.D., Karniadakis, G.E., 2020. Physics-informed neural networks for high-speed flows. Comput. Methods Appl. Mech. Engrg. 360, 112789.

Nabian, M.A., Gladstone, R.J., Meidani, H., 2021. Efficient training of physics-informed neural networks via importance sampling. Computer-Aided Civ. Infrastruct. Eng. 36 (8), 962–977.

Okazaki, T., Ito, T., Hirahara, K., Ueda, N., 2022. Physics-informed deep learning approach for modeling crustal deformation. Nat. Commun. 13 (1), 7092.

Pang, G., D'Elia, M., Parks, M., Karniadakis, G.E., 2020. Npinns: Nonlocal physics-informed neural networks for a parametrized nonlocal universal Laplacian operator. Algorithms and applications. J. Comput. Phys. 422, 109760.

Prantikos, K., Tsoukalas, L.H., Heifetz, A., 2022. Physics-informed neural network solution of point kinetics equations for a nuclear reactor digital twin. Energies 15 (20), 7697.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707.

Sandia National Laboratories, 2019. MAAP-MELCOR Crosswalk Phase 1 Study. Tech. Rep. SAND2019-xxxx, Sandia National Laboratories.

Sandia National Laboratories, 2021. Modular Accident Analysis Program (MAAP) – MELCOR Crosswalk: Phase II, Analyzing a Partially Recovered Accident Scenario. Tech. Rep. SAND2017-11975, Sandia National Laboratories.

Song, J., Ha, K., 2022. A simulation and machine learning informed diagnosis of the severe accidents. Nucl. Eng. Des. 395, 111881.

Song, J., Kim, S.J., 2023. A machine learning informed prediction of severe accident progressions in nuclear power plants. Nucl. Eng. Technol. 56 (6), 2266–2273.

Song, S.H., Lee, Y., Bae, J.Y., Song, K., Seo, M.R., Kim, S.J., Lee, J.I., 2024. Application of reinforcement learning to deduce nuclear power plant severe accident scenario. Ann. Nucl. Energy 205, 1–4. http://dx.doi.org/10.1016/2024/110605.

Song, J., Son, D., Ham, J., Bae, J., Bae, S., Ha, K., Kim, B., Yoon, S.H., Chung, B., Park, S., Park, C., Park, J., Choi, Y., 2023. A comparative simulation of severe accident progressions by CINEMA and MAAP5. Nucl. Eng. Des. 404, 112181. http://dx.doi.org/10.1016/j.nucengdes.2023.112181, URL https://www.sciencedirect.com/science/article/pii/S0029549323000304.

U.S. Nuclear Regulatory Commission, 2001. SCDAP/RELAP5/MOD3.3 Code Manual, Volume I–IV. Tech. rep., NUREG/CR-6150, Available at: https://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr6150/.

Vinuesa, R., Brunton, S.L., 2022. Enhancing computational fluid dynamics with machine learning. Nat. Comput. Sci. 2 (6), 358–366.

Wang, W., Ma, W., 2023. Bootstrapped artificial neural network model for uncertainty analysis in MELCOR simulation of severe accident. Prog. Nucl. Energy 157, 104556.

Wang, W., Ma, W., 2025. Coupling of MELCOR with surrogate model for quench estimation of conical debris beds. Ann. Nucl. Energy 211, 110933.

Wang, S., Teng, Y., Perdikaris, P., 2021. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM J. Sci. Comput. 43 (5), A3055–A3081.

Wang, J.-X., Wu, J.-L., Xiao, H., 2017. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. Phys. Rev. Fluids 2 (3), 034603.

Wei, C., Fan, Y., Zhou, Y., Liu, X., Li, C., Li, X., Wang, H., 2025. Physics-informed neural network based on control volumes for solving time-independent problems. Phys. Fluids 37 (3).

Yang, S., Kim, H., Hong, Y., Yee, K., Maulik, R., Kang, N., 2024. Data-driven physics-informed neural networks: A digital twin perspective. Comput. Methods Appl. Mech. Engrg. 428, 117075.

Zhang, W., Li, J., 2023. CPINNs: A coupled physics-informed neural networks for the closed-loop geothermal system. Comput. Math. Appl. 132, 161–179.

Zou, Z., Karniadakis, G.E., 2023. L-HYDRA: Multi-head physics-informed neural networks. arXiv preprint arXiv:2301.02152.