Water Research X 30 (2026) 100471

Contents lists available at ScienceDirect

Water Research X

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

Comparing different Physics-Informed Neural Network models for chlorine modeling in EPANET under varying initial and boundary conditions

Shimon Komarovsky [a,*], Raghad Shamaly [a], Gopinathan R. Abhijith [b,c], Andrea Cominola [d,e], Avi Ostfeld [a]

[a] Faculty of Civil and Environmental Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel
[b] Department of Civil Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, UP, India
[c] Kotak School of Sustainability, Indian Institute of Technology Kanpur, Kanpur 208016, UP, India
[d] Chair of Smart Water Networks, Technische Universität Berlin, Berlin, 10623, Germany
[e] Einstein Center Digital Future, Berlin, 10117, Germany

ARTICLE INFO

Keywords:
PINN
Differential equation
Water distribution systems
Water quality model

ABSTRACT

Accurate water quality modeling in water distribution systems (WDS) is essential for ensuring safe and reliable drinking water. While numerical solvers such as EPANET provide robust simulations, their computational cost increases substantially for real-time or large-scale applications, particularly when boundary and initial conditions vary over time. Existing Physics-Informed Neural Network (PINN) approaches face limitations in handling such changing conditions, despite their prevalence in real WDS operations. This study focuses on enhancing the adaptability of PINNs for chlorine modeling under diverse and dynamic scenarios. The proposed framework embeds the governing Advection–Reaction (AR) equation into a deep learning architecture and introduces targeted modifications to the formulation of boundary and initial condition losses. Training data are generated using EPANET simulations, and the framework is evaluated under multiple scenarios, including constant and time-varying velocities as well as fixed and dynamic boundary and initial conditions. Results demonstrate that a PINN model explicitly designed for boundary-condition adaptability can accurately reproduce EPANET water quality simulations while reducing computational demands. Key factors influencing performance, such as proper PDE specification, loss balancing, and data preprocessing, are identified. Although the analysis is conducted on a single-pipe testbed to isolate these effects, the findings establish an essential foundation for extending adaptive PINNs to full WDS networks. The primary contribution of this work is the development and demonstration of a PINN architecture capable of reliably adapting to varying boundary and initial conditions, addressing a critical gap in current PINN-based water quality modeling research.

1. Introduction

Water distribution systems (WDS) are vital components of critical urban infrastructure, ensuring a reliable supply of potable water to households, industries, and commercial establishments (Walski et al., 2003). Modeling both the hydraulic and water quality dynamics of these systems is essential for effective planning and management strategies aimed at optimizing operations, improving efficiency, and maintaining regulatory water quality standards (Rossman, 2000; Giustolisi et al., 2008). Water quality modeling, in particular, requires accurate characterization of the transport, mixing, and decay of chemical substances within networks (Rossman, 2000), providing essential insights for contamination detection, water safety assessment, and real-time decision support (Ostfeld et al., 2008).

Traditionally, water quality dynamics in WDS are modeled using advection–diffusion–reaction (ADR) equations to describe the movement and decay of substances such as chlorine, chloramine, or contaminants (Trussell and Umphres, 1978). These processes depend strongly on boundary and initial conditions, which vary with operational states, demand fluctuations, water source changes, valve operations, and contamination events. As a result, numerical simulations must often be recomputed for each change in hydraulic or chemical conditions, leading to substantial computational burdens for large networks, real-time applications, or repeated simulation tasks.

EPANET, an open-source software developed by the U.S. Environmental Protection Agency (EPA), one of the most widely used tools

* Corresponding author.
E-mail addresses: shiman@campus.technion.ac.il (S. Komarovsky), raghad.sh@campus.technion.ac.il (R. Shamaly), abhijith@iitk.ac.in (G.R. Abhijith), andrea.cominola@tu-berlin.de (A. Cominola), ostfeld@cv.technion.ac.il (A. Ostfeld).

https://doi.org/10.1016/j.wroa.2025.100471
Received 2 October 2025; Received in revised form 1 December 2025; Accepted 13 December 2025
Available online 22 December 2025

for WDS simulation (Rossman et al., 2020), accurately handles these dynamics but inherits the computational expense inherent to traditional numerical solvers. Recent studies highlight how these costs grow when boundary and initial conditions are rapidly changing, exactly when fast simulation is most critical (Klise et al., 2017).

To alleviate these limitations, data-driven machine-learning (ML) based surrogate models have been explored for WDS analysis (Shaw et al., 2017). Among these approaches, Physics-Informed Neural Networks (PINNs) have emerged as particularly promising due to their ability to incorporate governing physical laws directly into the learning process (Raissi et al., 2019). Specifically, PINNs are neural networks that leverage deep learning architectures to solve supervised learning tasks while embedding nonlinear physical laws, such as governing partial differential equations (PDEs), directly into the loss function. PINNs have shown strong potential as fast, physically consistent surrogate models for a variety of engineering applications (Karniadakis et al., 2021), and initial studies have demonstrated their applicability to WDS hydraulics (Ashraf et al., 2024). However, a major barrier remains largely unaddressed: standard PINN formulations struggle to adapt when boundary and initial conditions change, even though such changes are pervasive in realistic WDS operations.

Most existing PINN implementations assume fixed or simple boundary conditions, limiting their ability to generalize across different scenarios or to serve as flexible real-time surrogates. In the context of water quality modeling, where velocities, source concentrations, or contamination inputs may vary dynamically, this lack of adaptability becomes a central challenge. Addressing it is essential for developing PINN-based surrogates that can replace EPANET simulations in operational or scenario-based analyses.

In this study, we address this gap by developing and analyzing PINN architectures designed explicitly to adapt to diverse and changing boundary and initial conditions for chlorine transport in pipe systems. Our framework embeds the Advection–Reaction (AR) equation within a PINN structure and evaluates how architectural choices and loss function formulations influence the model's ability to generalize across fundamentally different PDE boundary regimes. We focus on a controlled single-pipe testbed to isolate, study, and compare these effects under (i) constant vs. time-varying velocities, (ii) fixed vs. dynamic concentration boundary conditions, and (iii) varying initial conditions.

In the initial development proposed in this study, our training dataset consists of numerical solutions derived from EPANET, but future implementations will integrate sensor-based real-time measurements.

The PINN takes the spatio-temporal coordinates and flow velocity (t, x, v) as input and predicts chlorine concentration $C(t, x, v)$, enabling it to capture the evolving transport dynamics. Although prior work has proposed conceptual formulations for PINNs in WDS applications (Daniel et al., 2024), practical investigations of adaptability to boundary and initial conditions remain limited. This study provides a detailed implementation, systematic evaluation, and discussion of the model behaviors that affect such adaptability.

The main contribution of this work is a comparative analysis of PINN model variations specifically aimed at improving adaptability to changing PDE boundary and initial conditions, addressing a critical requirement for deploying PINN-based surrogates in realistic WDS applications.

The remainder of this paper is organized as follows. Section 2 describes the methods, including the formulation of the chlorine model and the PINN framework. Section 3 outlines the experimental settings and simulation scenarios. Sections 4 and 5 discuss the results and analyze the performance of PINN models with different architectures across different conditions. Finally, Section 6 summarizes the findings and discusses future directions toward full-network implementations.

2. Methods

2.1. The chlorine model formulation

We use PINNs as a computationally efficient implementation of chlorine dynamics in pipe-distribution system. We model the concentration of chlorine within a pipe of the WDN via approximation of the 1D AR PDE, in unsteady and uniform flow conditions, as given in Eq. (1).

$$\frac{\partial C}{\partial t} + v\frac{\partial C}{\partial x} = -kC \tag{1}$$

subject to:

$$C(t, x = 0) = C_{un}(t), \quad \text{if } v \geq 0 \tag{2}$$

$$C(t, x = L) = C_{dn}(t), \quad \text{if } v < 0\,, \tag{3}$$

where:

C: concentration of chlorine in (mg/L),

C_{un}: concentration of chlorine at the upstream node in (mg/L), representing the starting point of the considered pipe ($x = 0$)

C_{dn}: concentration of chlorine at the downstream node in (mg/L), representing the ending point of the considered pipe ($x = L$)

v: flow velocity (m/s),

k: decay rate (per sec),

x: spatial coordinate (m),

t: temporal coordinate (sec).

$C(t, x = 0)$ and $C(t, x = L)$, for all $t \in [t_0, t_0+T]$, define the boundary conditions (BCs), as they constrain the PDE to the spatial boundaries $x = 0$ and $x = L$. Similarly, $C(t = t_0, x)$ for all $x \in [0, L]$ represents the initial conditions (ICs), constraining the PDE at the initial time step. Here, T denotes the time period of the given PDE (or simulation horizon), while L represents the total length of the considered pipe in the network.

While solving this PDE, we assume that the velocity v is derived from the hydraulic simulation solution, i.e., it is a function of other parameters: the diameter D, the length of the pipe L, and the roughness coefficient C_{HW}. But the biggest effect on the flow velocity is the changing demand at the nodes, situated on the edges of the presented pipe. Therefore, while training the PINN on varying velocity, we can assume it is also equivalent to training over changing parameters that determine the velocity, i.e. L, D, C_{HW}, representing pipes with different lengths, diameters, and roughness coefficients.

Finally, we justify our simplified AR PDE replacing the full ADR PDE as follows. In turbulent pipe flow conditions typical of WDSs, advection and turbulent mixing dominate solute transport, while molecular diffusion and longitudinal dispersion are negligible. The high Reynolds numbers produce strong velocity fluctuations that rapidly homogenize concentrations across the pipe, effectively eliminating diffusive gradients. Consequently, the AR formulation (i.e., plug flow assumption) widely adopted in models such as EPANET provides an accurate and computationally efficient representation of water quality dynamics without the need for explicit diffusion terms.

2.2. PINN prediction task and loss functions

The main motivation for implementing PINNs for chlorine modeling in pipe-distribution system is to improve the computational efficiency of chlorine concentration prediction. Unlike traditional machine learning, which relies purely on knowledge learned from the training data, PINNs integrate additional prior domain knowledge or constraints over the outputs.

Loss is the objective we decide upon to optimize a given Neural Network (NN), to approximate the best prediction of the NN's output. The loss should be such that it allows to generalize to unseen data. While most commonly in ML a data loss prediction is used, which minimizes the difference between observations (i.e., measurements) and

the corresponding predictions, then in PINN we also include knowledge about the physics and dynamics of the system. Specifically how input coordinates influence the output variables. Full knowledge includes the dynamics equation (ordinal differential equation or PDE) and the relevant conditions to represent a specific solution for the dynamics, i.e., initial and boundary conditions.

EPANET is used as our reference (high-fidelity) model to generate training data representing reality, since its outcomes have been already thoroughly compared to real measured data in various studies (Junaid and Izinyon, 2022; Kowalska et al., 2018). Next, we train a PINN model to approximate the generated data through minimization of a weighted combination of three losses: d=data loss (L_1 or L_{data}), p=PDE loss (L_2 or L_{PDE}), and c=condition loss (L_3 or $L_{BC\&IC}$), as formulated in Eq. (4).

$$L = \sum_{i=1}^{3} w_i L_i \,, \tag{4}$$

$$\begin{aligned}
\text{where} \quad L_1 = L_{data} &= \frac{1}{N}\sum_{i=1}^{N}(C(t_i,x_i) - \hat{C}(t_i,x_i))^2, \\
L_2 = L_{PDE} &= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\partial \hat{C}}{\partial t}|_{t_i} + v\frac{\partial \hat{C}}{\partial x}|_{x_i} + k\hat{C}(t_i,x_i)\right)^2, \quad \text{and} \\
L_3 = L_{BC\&IC} &= \frac{1}{N}\sum_{i=1}^{N}\left(C(t_i=0,x_i) - \hat{C}(t_i=0,x_i)\right)^2 \\
&+ \frac{1}{N}\sum_{i=1}^{N}\left(C(t_i,x_i=0) - \hat{C}(t_i,x_i=0)\right)^2 \\
&+ \frac{1}{N}\sum_{i=1}^{N}\left(C(t_i,x_i=L) - \hat{C}(t_i,x_i=L)\right)^2
\end{aligned}$$

with N being the number of data points,
C being the true value of chlorine concentration, and
$\hat{C}$ being the predicted value.

The data loss (L_{data}) is defined as the mean squared error between predicted and observed chlorine concentrations, ensuring accurate data-driven fitting in line with standard supervised learning practices.

The physics loss (L_{PDE}) embeds the known advection–reaction dynamics governing chlorine transport and decay over time, enforcing that the model predictions comply with the underlying physical laws.

The constraint loss ($L_{BC\&IC}$) ensures adherence to the initial and boundary conditions (ICs and BCs), enabling the network to produce solutions consistent with the PDE boundaries.

During early experimentation, particularly when applying the model to more complex scenarios (see Section 4), training that included all loss components (data + physics + continuity) unexpectedly produced inferior results compared to using data loss alone. This observation indicated inconsistencies within the physics and constraint terms. Consequently, the model was employed inversely – as a physics-informed diagnostic tool – to identify and correct errors within the PDE formulation. Details of this inverse PINN-based correction are provided in the appendix in Appendix A.1.

2.3. PINN models

Several deep neural network (DNN) architectures were explored in this research to comparatively analyze their performance. More specifically, we comparatively analyzed three different architectures with increasing complexity.

Initially, we employed a basic PINN setup (see Fig. 1(a)). However, after analyzing different possible boundary and initial conditions, we determined that the model should account for varying PDEs rather than a single PDE throughout the simulation. To address this, we adjusted the time coordinate from t to relative time $t - t_0$ and incorporated IC and BC as additional inputs, yielding the improved model in Fig. 1(b).

Subsequently, we further refined the model by integrating prior knowledge through inductive biases. This was achieved by introducing separate encoder DNNs for the ICs and BCs, as depicted in Fig. 1(c).

To evaluate the effectiveness of the proposed framework, we designed and comparatively tested three neural network architectures as part of the three above models. The three NN architectures capture different learning paradigms:

- **Fully Connected Neural Network (DNN)**: a two-layer feed-forward network with 32 neurons per hidden layer and ReLU (Rectified Linear Unit) activation, followed by a Tanh-activated output neuron. This baseline configuration provides a deterministic, nonlinear function approximator for mapping spatial–temporal inputs to chlorine concentrations.
- **Transformer Network**: a sequence-oriented model beginning with a linear embedding layer (32 neurons) that projects the input vector into a latent representation, followed by two transformer encoder blocks with 4 attention heads, 0.1 dropout, and 32 hidden neurons. The output layer applies a Tanh activation to produce a one-dimensional concentration estimate. This design allows spatial–temporal dependencies to be captured effectively.
- **Hybrid Network**: a composite architecture integrating embedding sub-networks for IC and BC representations. The IC embedding consists of three layers with [32, 16, 16] neurons, and the BC embedding of two layers with [16, 16] neurons, all using Tanh activations. These learned embeddings capture context-specific features before being concatenated with the remaining input features and then processed through a main network, which is either a Transformer or a five-layer DNN with [32, 16, 16, 16, 16] neurons and alternating ReLU and Tanh activations. A final linear output layer maps to the target concentration.

This comparison allows assessing the trade-offs between fully connected and attention-based models, as well as the benefits of embedding boundary-related priors.

Model 1 and 2 use either a simple Feed-Forward Neural Network (FFNN) or a Transformer. The difference between Model 1 and 2 was only in the input dimension size, since they used different inputs. Model 3 used a hybrid architecture, since it had also embedding layers to reduce the hidden features dimension for the IC and BC.

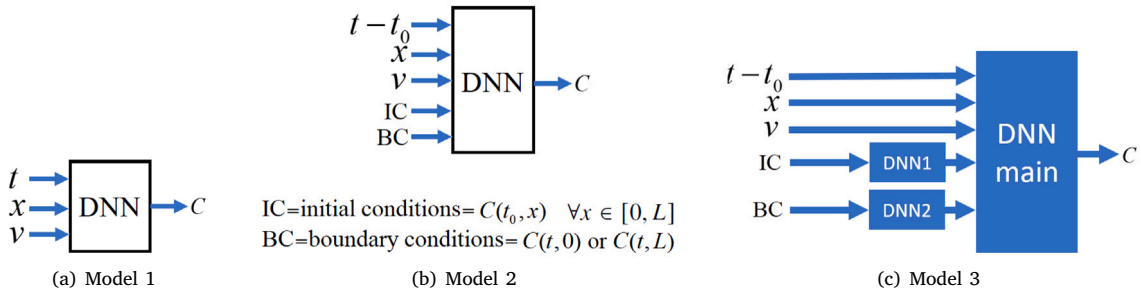
Model training followed a supervised learning framework with 80% of data used for training and 20% reserved for testing. The training utilized the Adaptive Moment Estimation (Adam) optimizer with a learning rate of $1 \cdot 10^{-5}$, and a batch size of 32. Each training instance corresponded to a one-minute water-quality simulation step, consistent with EPANET's temporal resolution. These hyperparameters were selected to balance convergence stability and computational efficiency across all architectures. Note that these are the best values that we received (for the architecture and training configuration). Many different values were examined during this research, via a random hyperparameter search method.

2.4. Evaluation metrics

We comparatively evaluate the performance of the three PINN models based on a set of state-of-the-art model error metrics. Since all experiments involve regression tasks, the Mean Square Error (MSE) serves as the primary loss function, defined in Eq. (5):

$$L = \frac{1}{n \cdot m}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(C_{i,j} - \hat{C}_{i,j}\right)^2 , \tag{5}$$

where C and $\hat{C}$ denote the actual and predicted values, respectively, while n and m represent the number of features and samples. In our case, the output is a 1-dimensional concentration variable ($m = 1$). More generally, m may have higher dimensions, which could be utilized for cases with multiple substances.



(a) Model 1 (b) Model 2 (c) Model 3

Fig. 1. Different PINN models that were comparatively analyzed throughout this research.

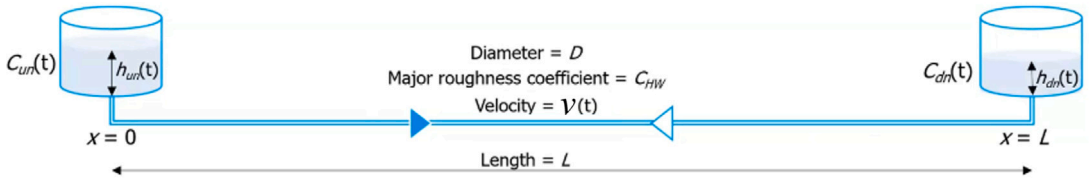


Fig. 2. The 1-pipe water distribution network used in the study.

Additional evaluation metrics include Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE), as expressed in Eq. (6):

$$\text{MAE} = \frac{1}{n \cdot m} \sum_{i=1}^{m} \sum_{j=1}^{n} |C_{i,j} - \hat{C}_{i,j}|,$$

$$\text{MAPE} = \frac{1}{n \cdot m} \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{|C_{i,j} - \hat{C}_{i,j}|}{C_{i,j}}, \tag{6}$$

$$\text{RMSE} = \sqrt{\frac{1}{n \cdot m} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(C_{i,j} - \hat{C}_{i,j}\right)^2}.$$

MSE is applied both during model training as a loss function, and during the testing phase for final model assessment. The additional metrics in Eq. (6) are used to assess performance on the test set.

Finally, the units of the different metrics are as following: MSE (mg^2/L^2), MAE (mg/L), MAPE (unitless), and RMSE (mg/L).

3. Experimental settings

For the simulations required to generate training data, we utilize the EPANET simulation tool. Specifically, hydraulic and water quality simulations are performed using the Water Network Tool for Resiliency (WNTR) Python package, which is built on EPANET version 2.2 (Klise et al., 2017; Rossman et al., 2020).

The PINN models were implemented using the Pytorch framework in Python version 3.11, via regular NNs, including the transformer package.

The water distribution network considered as a testing case in this research, as shown in Fig. 2, consists of two tanks connected by a water pipe.

In the corresponding EPANET model, this 1-pipe water distribution network is discretized by implementing 20 virtual, non-consuming nodes, separated by equally 50 meter long pipes, designed to store chemical concentration data throughout all quality steps in the simulation.

The EPANET simulation settings adopted for data generation consisted of the following temporal resolutions: hydraulic is set at 1-hour step, demand pattern is set at a 1-minute step, and the minimal possible water quality simulation step is set at 1 min. The weights of the loss function in Eq. (4) were set to $w_1 = w_2 = w_3 = 100$.

Fig. 3 illustrates the full dataset generated by EPANET for a simulation with a time horizon of 8-hours. As illustrated in the figure, the generated dataset consists of $C(t, x)$ for all $t \in [t_0, t_0 + T]$ and for all $x \in [0, L]$. Unlike traditional PINN with a single IC and BC throughout the whole simulation, we assume a varying IC or BCs at every hydraulic step, which then remain constant during the single step simulation.

Following data generation, we trained the PINN models described in the previous section considering the following different scenarios. The rationale behind this sequence of scenarios was to construct the PINN from simple to complex cases, to handle errors and issues encountered while increasing complexity at each new case:

1. **Case 1:** The velocity and BCs remain constant throughout the entire 3-day simulation. The decay rate is set to $k = 0$. Model 1 is used.
2. **Case 2:** The velocity and BC are fixed for the entire 8-day simulation, with a decay rate of $k = -\frac{10}{24 \cdot 3600} \left[\frac{1}{sec}\right]$. Model 1 is used.
3. **Case 3:** The velocity remains constant, while the BC changes every hour (at each hydraulic step) but remains fixed within that hour. The simulation runs for 10 days with $k = -\frac{10}{24 \cdot 3600} \left[\frac{1}{sec}\right]$. Model 2 is used.
4. **Case 4:** Both velocity and BC vary at each hydraulic step (1 h) but remain constant during all quality steps within the hydraulic step. The simulation duration is 3 days, with $k = -\frac{10}{24 \cdot 3600} \left[\frac{1}{sec}\right]$. Model 3 is used.
5. **Case 5:** The velocity changes at each hydraulic step (1 h), while the BC varies at every water quality time step (1 min) according to a predefined function of time. The simulation runs for 3 days with $k = -\frac{10}{24 \cdot 3600} \left[\frac{1}{sec}\right]$. Model 3 is used.
6. **Case 6:** The velocity changes at each hydraulic step (1 h), while the BC fluctuates at each quality time step (1 min) within a uniformly random range of 90% to 110% around the initial value at the start of the hydraulic step. The simulation duration is 3 days, with $k = -\frac{10}{24 \cdot 3600} \left[\frac{1}{sec}\right]$. Model 3 is used.

The different cases require different PDE assumptions. The simple cases 1 and 2 assume that the PDE and its constraints (conditions) remain constant throughout the entire simulation. Conversely, this assumption will not hold in the more complex cases, in which either v or BC were not fixed. This means that whenever any part of the PDE

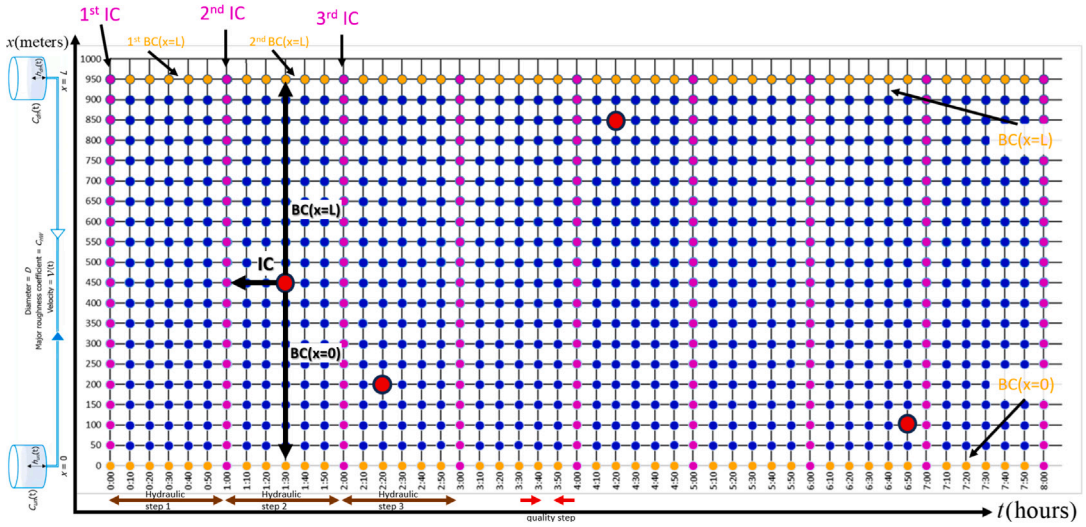


Fig. 3. Illustration of the EPANET-generated water quality dataset, with a batch of (red) points sampled from it for PINN training/testing. Each blue point represents a generated contaminant concentration data point, with coordinates (t,x). Samples marked in yellow correspond to the boundary conditions ($x = 0$) and ($x = L$). Samples marked in pink represent the data point at the beginning of each simulated hour.
Legend: A grid of $C(t, x)$ for all $t \in [0, 8\ \text{hrs}]$ and for all $x \in [0, 950\text{m}]$. x: 20 nodes 50 m apart. t: 8 h simulation, hydraulic step=1hr, quality step=10min.
Every new hydraulic step/regime starts with new IC and new BC for x=0 m and x=L=950 m. Shown 4 randomly selected data points for PINN training.
For example, point $C(x = 450\ \text{m}, t = 1{:}30\text{hrs})$ has IC $C(x, t = 1{:}00\text{hrs})$ for all $x \in [0, 950\text{m}]$, and BC: $C(x = 0\ \text{m}, t = 1{:}30\text{hrs})$ and $C(x = 950\ \text{m}, t = 1{:}30\text{hrs})$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

changes, the entire dynamics must be restarted. That is, new initial and boundary conditions must be set, reflecting the modifications in the PDE from that point onward.

In planning for case 3, a similar approach should produce a high loss, which forces us to explore alternative PINN models. We could implemented a standard fully connected neural network (FCNN) and a Transformer model, as depicted in Fig. 1(b). However, the input dimensionality would be excessively high. Specifically, the initial condition (IC) input contained $C(t_0, x)$ for all $x \in [0, L]$, with 20 nodes along this range. This mixing of inputs would yield hindered learning.

After initial evaluations, several methodological refinements were introduced to improve accuracy, stability, and physical consistency.

(a) **Embedding-based representation of boundary conditions.** Rather than processing ICs directly as raw values, embedding layers were introduced to reduce their dimensionality and extract higher-level representations. This approach allowed the model to distinguish between IC patterns while avoiding excessive sensitivity to node-specific numerical variations. The resulting model, illustrated in Fig. 1(c), processes ICs and BCs through separate embedding networks before merging them with the main input stream.

(b) **Continuity across PDE regimes.** Because flow velocity and boundary conditions can vary between PDE regimes, the transitions between regimes were reformulated to ensure smooth, continuous changes rather than abrupt discontinuities.

(c) **Correction of loss weighting.** Each data point (t, x) contributes simultaneously to all three loss components; however, IC loss is only relevant across different x values, not t values. This results in redundant IC losses. Given that the EPANET water quality simulation step is set to 1 min while each PDE regime lasts 1 h, IC loss is unnecessarily repeated 60 times. Similarly, BC loss is repeated across all 20 nodes in the pipe. To correct for this, the loss function was reweighted as:

$$L = w_1 L_{data} + w_2 L_{PDE} + \frac{w_3}{60} L_{IC} + \frac{w_3}{20} L_{BC} \tag{7}$$

(d) **Optimized data utilization for loss computation.** While the data loss term requires ground-truth labels, the PDE and constraint losses depend only on model predictions and known physical inputs. Therefore, the latter components were trained using 100% of the available data without risk of data leakage, maximizing the effectiveness of the physics-informed supervision. In other words, the PDE loss used 100% of the (t, x) tuples in the full dataset of (t, x, C), while the data loss used 80% as in original settings. BC and IC losses also used 100%, since we assume that both initial and boundary conditions are fully known.

Note that the different methodological adaptations for the different cases we proposed above are general. That is, they depend on the characteristics of ICs and BCs only. Thus, they can be applied to any pipe type, with our simple network as merely an example.

4. Results

This section is organized based on the complexity of the examined cases, beginning with simpler cases and progressing to more complex ones. The results provide insights into the model's performance and limitations under different BCs and velocity (v) configurations.

4.1. Simple cases

Initially, we examined cases with constant velocity (v) and BC. Specifically, we set $C(t, x = 0) = 0$ and $C(t, x = L) = 1$. This scenario, referred to as "Case 1" assumes $k = 0$, representing a simple spatio-temporal continuity equation. Later, in Case 2, we analyzed the scenario where $k > 0$. For both cases, we employed the simple PINN model shown in Fig. 1(a).

In these simpler cases, we obtained near-perfect results with very low test loss, as shown in Fig. 4. However, when either v or BC were not fixed, the performance deteriorated significantly, leading to a poor match between the EPANET true concentration and the predicted concentration by PINN.

This decrease in performance was due to incorrect assumptions regarding the PDE. While in the simple case we assumed a fixed PDE, the changing cases assumed a changing PDE. Consequently, the original PINN model in Fig. 1(a) was replaced with a model that accounts for

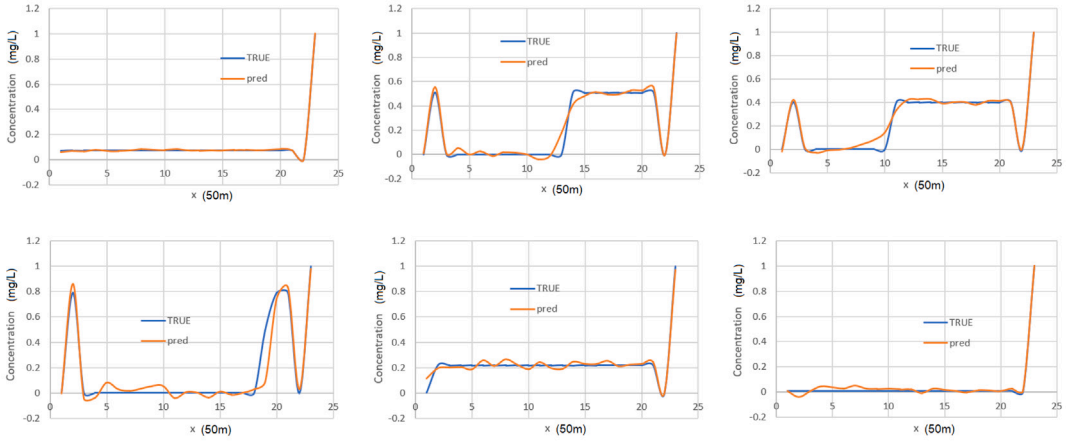


Fig. 4. Comparison between true (EPANET quality simulation) versus predicted (PINNs) concentration along the pipe (x) in Case 1 at various time steps. Note that because these are polynomial trendlines, some data points may appear as negative values.

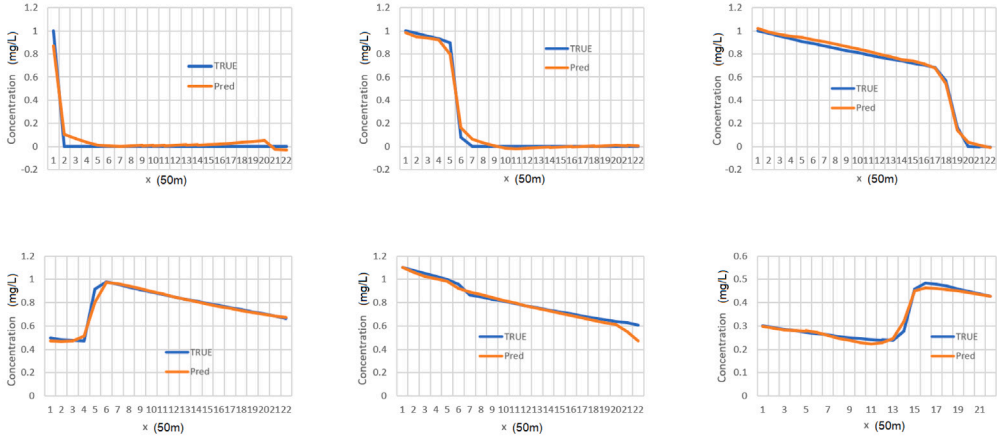


Fig. 5. Comparison between true concentrations from EPANET quality simulations and predicted concentrations from PINNs along the pipe (x) at various time steps in Case 3.

ICs, BCs, and elapsed time since the most recent PDE change, as shown in Fig. 1(b).

Alongside using a simple fully connected neural network (FCNN) for PINN, we also explored a Transformer-based model. By tuning key parameters such as the number of encoding layers, feature dimensions, and attention heads, we mitigated overfitting and achieved improved results compared to the FCNN.

4.2. Changing velocity and boundary condition cases

Next, we examined Case 3, where velocity remained fixed while BCs changed over a 10-day simulation period, with a decay coefficient of $k = -\frac{10}{24 \cdot 3600}\left[\frac{1}{sec}\right]$. All loss components (data, PDE, and constraint) were included after training for 1000 epochs. The best test loss (data loss only) was 0.003, as shown in Fig. 5.

As planned in the methodology, case 3 should use a more appropriate PINN model. Indeed, implementing the model from the simpler cases yielded a high loss. Subsequently, we used the model depicted in Fig. 1(c). However, even with this adjustment, many results showed that including PDE and constraint losses did not always improve performance. In some cases, data loss alone performed better than when additional loss terms (PDE and conditions) were included.

To handle this issue, we included continuous transitions where the velocity or BC are changed in the PDE, and weight correction of the three loss components, see Eq. (7).

Table 1
Comparison of different problem complexity cases.

Case	Velocity (v) (m/s)	BC	Simulation (days)	k (1/(3600*24) 1/s)	Test Loss (mg^2/L^2)
1	Fixed	Fixed	3	0	0.0005
2	Fixed	Fixed	8	−10	0.0001
3	Fixed	Changing	10	−10	0.0013
4	Changing	Changing	3	−10	0.0023

Legend: BC = boundary conditions, k = decay coefficient , test loss = at final epoch of training.

These corrections, along with using the exact number of repetitions for BC and IC, improved the model performance.

Since computational limitations affected training duration, we ran Case 4 (changing velocity and BCs) for 2000 epochs, which is much less epochs compared to the previous simpler cases.

We also refined our data usage for different loss components, as planned in the methodology. That is, training the PDE and constraint losses using 100% of the data.

4.2.1. Summary of cases

Table 1 summarizes different simulation cases and their test losses.

However, Case 4 was not the most complex scenario, as BCs only changed between PDE transitions, remaining constant within each PDE period (1 h or 60 water quality simulation steps).

Table 2
Comparison of test losses for varying BCs under different training losses.

Case	Data Loss (d) (mg^2/L^2)	Data + PDE (d+p) (mg^2/L^2)	Data + PDE + Conditions (d+p+c) (mg^2/L^2)
5	0.0157	0.0155	0.0116
6	0.0063	0.0055	0.0055

Finally, the comparison among the different cases was done for fixed parameters. However, we present in Table 1 additional simulations, where the number of days is different, to emphasize that the performance was almost similar. Meaning that most of the effect is due to velocity and BC configuration, and is insignificant for the simulation duration.

4.2.2. More complex cases

In the more cases we tested, BCs varied at each water quality simulation step. First, in Case 5, BCs followed a functional relationship: $C(t, x = 0) = C_1(t)$ and $C(t, x = L) = C_2(t)$ for all t. This setup performed poorly, i.e., the data losses presented in Table 2 are worse by an order of magnitude compared to previous less complex cases.

Next, in Case 6, BCs fluctuated randomly around their initial values: $C(t, x = 0) \in [0.9, 1.1] \cdot C(t_0, x = 0)$ and $C(t, x = L) \in [0.9, 1.1] \cdot C(t_0, x = L)$ for all t. This case represented system robustness to uniformly distributed noise and yielded better results.

Table 2 compares test losses across these cases with different loss functions.

Results indicate that including PDE and conditions losses generally improved performance, though the effect of conditions loss varied across experiments. Multiple runs were conducted to account for stochastic variations in neural network initialization and optimization.

Additional analyses, specifically the scalability assessment and the computation time comparison between EPANET and the PINN model, while secondary to the primary objectives of this study, are provided in Appendix in Appendix A.2.

5. Discussion

The goal of this paper was to investigate chlorine modeling with PINN in the simple 1-pipe water distribution network depicted in Fig. 2.

Our methodology involved starting with the simplest cases and gradually increasing complexity while addressing challenges encountered along the way. Several issues arose during the research, as described in Section 4.

One key challenge was constructing the most suitable DNN model for our specific task. This is a crucial step in deep learning design, where selecting an appropriate architecture enables effective data processing and minimizes loss on unseen (test) data.

Understanding the nature of the data is essential for this step. For example, convolutional neural networks (CNNs) are well-suited for image processing, recurrent neural networks (RNNs) for text, and graph neural networks (GNNs) for graph-structured data. Additionally, considerations such as parameter sharing or separation play a role in DNN design.

In our case, we reduced the input data size of the IC input to mitigate the curse of dimensionality, a common deep learning challenge where higher-dimensional data requires a significantly larger dataset to achieve comparable performance. To address this, we incorporated additional encoding or embedding layers to process IC and BC separately. This adjustment led to improved results.

Another key observation was the importance of data cleaning and monitoring. Neglecting these aspects can result in poor performance, inconsistencies, or illogical outcomes.

During our study, we encountered several such scenarios:

Table 3
Performance metrics for the runs in Fig. 6.

Case	MAE (mg/L)	MAPE (unitless)	RMSE (mg/L)	Test Loss (mg^2/L^2)
24 h d+p+c	0.05866	823.2	0.08829	0.007785
24 h d	0.06111	517	0.08995	0.008084
72 h d+p+c	0.0568	29 876	0.08625	0.007438
72 h d	0.06445	47 941	0.09514	0.009051
240 h d+p+c	0.04404	$5.62 \cdot 10^{11}$	0.0692	0.004789
240 h d	0.05095	$3.61 \cdot 10^{9}$	0.08028	0.006446

Table 4
Comparison of running time of different cases in test time.

Case	Simulation (days)	Running time PINN (s)	Running time EPANET (s)
1	1	0.3	0.17
2	3	0.77	0.48
3	10	1.59	1.63

Conditions: CPU, training data size = 80%, testing data size = 20%, quality step = 1 min.

- Excessively high test loss.
- Cases where training with only the data loss (d loss) outperformed training with data, physics, and consistency losses (d+p+c losses). This suggests that incorporating system dynamics not only failed to improve training but actually worsened it, indicating a computational error.
- Incorrectly weighting different loss components. Specifically, multiple occurrences of physics (p) and conditions (c) losses led to an imbalanced total loss, distorting training effectiveness.

Ultimately, after addressing these issues, our experiments confirmed the value of PINNs over standard data-driven models. By enforcing system dynamics during training, PINNs improved predictions both on the training set and, more importantly, on the test set.

6. Conclusion

This study explored the development of a surrogate model based on PINNs for chlorine modeling in pipe-distribution system, demonstrating its capability to approximate numerical simulations obtained with traditional hydraulic and water quality modeling effectively. Several key conclusions emerge from our analysis.

First, PINNs consistently outperformed standard data-only neural networks, highlighting the value of incorporating physical knowledge into the learning process via the loss function. However, this advantage relies heavily on the accurate formulation of the governing PDEs. Any misrepresentation in the physical process may misguide the model during training. Training with only the data loss versus combining data, physics, and constraint losses resulted in notable performance differences.

Training performance was also found sensitive to the relative weights used to balance the different components of the loss function. Additionally, careful data cleaning and monitoring were shown to significantly affect predictive accuracy.

The study further reveals that training outcomes are sensitive to stochastic or design factors such as neural network weight initialization, emphasizing the need to account for variability across different training configurations. Moreover, increasing the size of the training dataset improved generalization and reduced test errors.

This study focused on a single-pipe experiment as an initial step toward exploring the applicability of PINNs for water quality modeling. This controlled setup provided the foundation for understanding the model's behavior and validating its core mechanisms before extending the approach to a full WDS.

Future research should build on these findings by developing adaptive loss weighting strategies, hybrid architectures that integrate PINNs with traditional numerical solvers, and incorporating real-world sensor data to enhance model robustness and applicability in practical WDS scenarios.

CRediT authorship contribution statement

Shimon Komarovsky: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Raghad Shamaly:** Writing – review & editing. **Gopinathan R. Abhijith:** Writing – review & editing, Resources, Conceptualization. **Andrea Cominola:** Writing – review & editing, Validation, Methodology, Conceptualization. **Avi Ostfeld:** Writing – review & editing, Validation, Supervision, Resources, Project administration, 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 research was supported by The Israeli Water Authority under project number 2033800, and by the Bernard M. Gordon Center for Systems Engineering at the Technion.

Appendix

A.1. Correcting the PDE formulation

After examining cases 1 and 2, see Section 4.1, we observed unsatisfactory results in the more complex cases in this research.

We figured out that the problem was in our PDE. It was incorrect.

Subsequently, after multiple trials, we decided to utilize the PINN for an inverse problem or system identification task. Unlike standard prediction tasks, this approach focuses on discovering the governing PDE that best describes the system. In general, this involves solving for $\mathcal{N}$ such that $\mathcal{N}(t, x, v) = 0$, where $\mathcal{N}$ represents a function incorporating relevant coordinates and parameters. In our case, these are t, x, and velocity v.

More specifically, we assumed the overall PDE structure was correct but were uncertain about certain internal relationships. To address this, we introduced learnable parameters into the original PDE (Eq. (1)) and estimated them using data from EPANET simulations. In other words, the problem was in our suspicion about misalignment between the EPANET's model (which we assumed to be correct) and our PDE formulation. Thus our goal was to fix our PDE according to EPANET. The modified PDE with learnable parameters is given in Eq. (8):

$$\frac{\partial C}{\partial t} + v \cdot p_1 \cdot \frac{\partial C}{\partial x} = -k \cdot p_2 \cdot C \tag{8}$$

After sufficient training, the best-fit parameters obtained were $p_1 \approx 0.98$ and $p_2 \approx -0.67$. We then returned to our original prediction task, setting $p_1 = 1$ and evaluating the performance for both $p_2 = -0.5$ and $p_2 = -1$. The case with $p_2 = -1$ performed significantly better than $p_2 = -0.5$, leading us to conclude that our original PDE was correct but had an incorrect sign for the k term.

Further review of the EPANET documentation confirmed that the PDE used in EPANET for solving the chlorine model was similar to ours (Eq. (1)), except with an opposite sign for the k term. That is, EPANET employs:

$$\frac{\partial C}{\partial t} + v\frac{\partial C}{\partial x} = +kC \tag{9}$$

This discovery validated our corrected PDE formulation and improved model accuracy.

A.2. Scalability analysis

In this section, we examine how model performance depends on the size of the training dataset, focusing on the most complex scenario, case 6.

The results are summarized in Fig. 6, which compares test losses during training across different dataset sizes and training loss configurations. Additionally, Table 1 presents the best performance measures for each data scale and loss setting.

As shown in Fig. 6 and Table 3, increasing the training dataset size from 24 h to 240 h significantly reduced test loss (from 0.0078 to 0.0048). Other performance metrics also improved, emphasizing the importance of sufficient training data for PINNs.

Additionally, comparing all cases from the simplest to the most complex, we observe from Tables 1 and 2 that more complex scenarios result in higher test losses. This suggests that the model struggles to generalize effectively when dealing with increased complexity in the problem formulation.

Finally, we report a comparison in terms of computational time required by the PINN model versus the EPANET (Physics-based) model, in testing or inference, i.e. while using it in practice. See Table 4 summarizes different simulation cases and their total running time.

The computational times reported here are based on the performance of a workstation with 12th Gen Intel(R) Core(TM) i3-12100 (3.30 GHz) CPU, and 32 GB of RAM.

Note that this comparison may suggest that PINN is not significantly improve the running time, hence might not be a suitable surrogate model replacing EPANET. However, we need to consider several points that may give an advantage to PINNs over EPANET:

1. **Processing unit:** In our case we used CPU to train and test our PINN. As it well known, GPU are preferred in such cases, since they are much more efficient and faster in parallel computing and matrix multiplication. EPANET would not benefit from it, but in larger models or data GPU would accelerate PINN substantially.
2. **Network size:** The network size affects EPANET computation time, and PINN training time. However, it should not significantly affect PINN's inference time (only within ICs as inputs). We used a tiny network. Hence, a bigger network would increase the gap between EPANET and PINN computation time.
3. **Time steps:** Table 4 shows that for small simulation time EPANET is faster in producing the full dataset. However, as the simulation time increases, this gap decreases and at some point even flips, making the PINN faster. We can assume and extrapolate that at much larger temporal data size (simulation horizon) PINN becomes significantly faster. This could be realized either by a larger simulation periods or by a higher time resolution (smaller quality times).
4. **Solution technique:** The EPANET solves hydraulics for the entire network at each hydraulic step. Then, at smaller quality steps, quality is solved for the entire network, i.e. $C(t = t_i, x)$ for all $x \in [0, L]$ at time step $t = t_i$. Eventually, the simulation proceed sequentially, solving for all quality steps, with intermediate hydraulic solutions. Differently, PINN computes $C(t = t_i, x = x_i)$ directly. It skips the need to solve hydraulics or quality for the entire network, and more importantly, it avoids the necessity to compute it sequentially, representing thus an advantage over traditional physical modeling.
   A fair comparison of the computational requirements for $C(t, x)$ for all $x \in [0, L]$ and for all $t \in [t_0, t_0 + T]$ is not straightforward. While EPANET simulates outputs sequentially, in PINN it was done only over the 20% of the test data. Therefore, we derived the total running time as the total inference time over 20% of the data divided by 0.2 to represent fair comparison over the whole data in x and t.

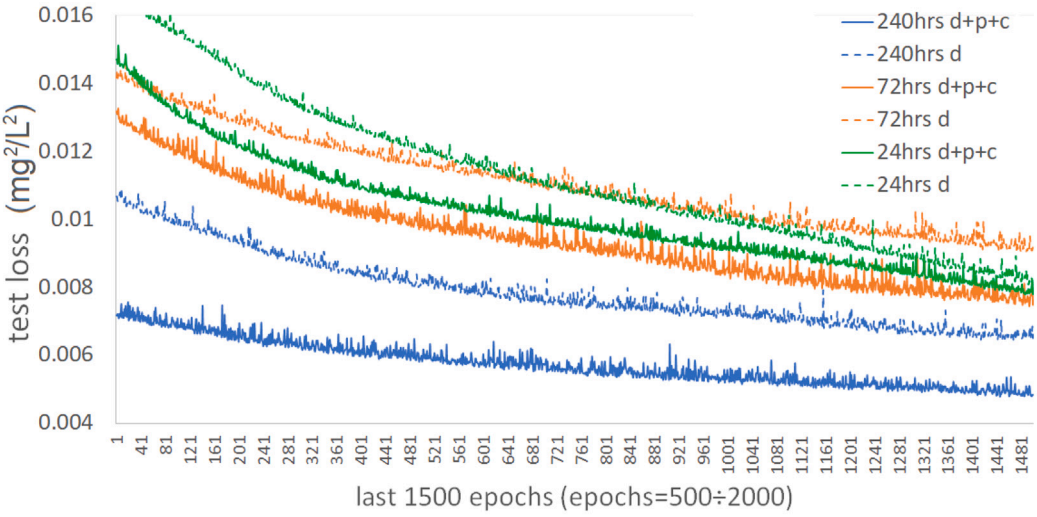


Fig. 6. Comparison of test loss across different data scales. Each dataset size includes results for both d+p+c training and d-only training.

However, as explained above, PINN can compute $C(t = t_i, x = x_i)$ pointwise. Either for specific (t_i, x_i) or a batch of those, see e.g. Fig. 3. In such cases the PINN would be greatly faster, while EPANET will still have the same computation time, since it has to account for all possible $x \in [0, L]$ and for all possible $t \in [t_0, t_0 + T]$.

Data availability

All data, models, and code that support the findings of this study are available from the corresponding author upon reasonable request.

References

Ashraf, Inaam, Strotherm, Janine, Hermes, Luca, Hammer, Barbara, 2024. Physics-informed graph neural networks for water distribution systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, (20), pp. 21905–21913.

Daniel, Ivo, Abhijith, Gopinathan R, Kutz, J Nathan, Ostfeld, Avi, Cominola, Andrea, 2024. Physics-informed machine learning for universal surrogate modelling of water quality parameters in water distribution networks. Eng. Proc. 69 (1), 205.

Giustolisi, Orazio, Savic, Dragan, Kapelan, Zoran, 2008. Pressure-driven demand and leakage simulation for water distribution networks. J. Hydraul. Eng. 134 (5), 626–635.

Junaid, C.T., Izinyon, O.C., 2022. Hydraulic and water quality modelling of water distribution networks using EPANET software. Niger. J. Env. Sci. Technol. (NIJEST) 6 (1), 172–179.

Karniadakis, George Em, Kevrekidis, Ioannis G, Lu, Lu, Perdikaris, Paris, Wang, Sifan, Yang, Liu, 2021. Physics-informed machine learning. Nat. Rev. Phys. 3 (6), 422–440.

Klise, Katherine A, Hart, David B, Moriarty, Dylan, Bynum, Michael L, Murray, Regan, Burkhardt, Jonathan, Haxton, Terra, 2017. Water network tool for resilience (WNTR) user manual. p. 50.

Kowalska, Beata, Hołota, Ewa, Kowalski, Dariusz, 2018. Simulation of chlorine concentration changes in a real water supply network using epanet 2.0 and watergems software packages. WIT Trans. Built Environ. 184, 39–48.

Ostfeld, Avi, Uber, James G, Salomons, Elad, Berry, Jonathan W, Hart, William E, Phillips, Cindy A, Watson, Jean Paul, Dorini, Gianluca, Jonkergouw, Philip, Kapelan, Zoran, et al., 2008. The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. J. Water Resour. Plan. Manag. 134 (6), 556–568.

Raissi, Maziar, Perdikaris, Paris, Karniadakis, George 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.

Rossman, L.A., 2000. EPANET 2 Users Manual. US Environmental Protection Agency, Washington, DC. Technical Report, EPA/600/R-00/057.

Rossman, Lewis A, Woo, Hyoungmin, Tryby, Michael, Shang, Feng, Janke, Robert, Haxton, Terranna, 2020. EPANET 2.2 User Manual. p. 190.

Shaw, Amelia R, Smith Sawyer, Heather, LeBoeuf, Eugene J, McDonald, Mark P, Hadjerioua, Boualem, 2017. Hydropower optimization using artificial neural network surrogate models of a high-fidelity hydrodynamics and water quality model. Water Resour. Res. 53 (11), 9444–9461.

Trussell, R. Rhodes, Umphres, Mark D., 1978. The formation of trihalomethanes. J.-Am. Water Work. Assoc. 70 (11), 604–612.

Walski, Thomas M, Chase, Donald V, Savic, Dragan A, Grayman, Walter, Beckwith, Stephen, Koelle, Edmundo, 2003. Advanced Water Distribution Modeling and Management. Haestad Press.