Energy and AI 24 (2026) 100721

Contents lists available at ScienceDirect

Energy and AI

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

Graph neural network-based surrogate modeling for fast and scalable simulations of meshed district heating networks

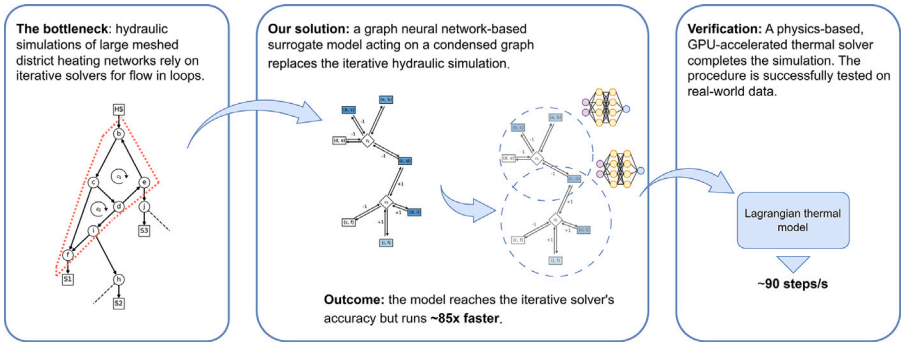
Roberto Boghetti *, Jean-Marc Odobez, Jérôme H. Kämpf

Idiap Research Institute, Rue Marconi 19, Martigny, 1920, Switzerland
EPFL, L'IDIAP, Station 14, Lausanne, 1015, Switzerland

HIGHLIGHTS

- A graph neural network surrogate enables fast simulation of meshed heating networks.
- Achieves 85x speedup over iterative solvers with 97% convergence rate.
- Shows negligible discrepancies with physics-based tools on monitoring data.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
District heating networks
Graph neural networks
Surrogate modeling
Simulation
Machine learning

ABSTRACT

District heating networks are key to the decarbonization of heating, yet their high capital costs and long payback periods hinder investments. Simulation-based applications such as real-time control optimization, which rely on scenario analysis or uncertainty quantification, can reduce costs and risks but face prohibitive computational burdens when numerous forward evaluations are required. Graph neural network-based surrogates are promising for improving simulation speed, but their local receptive fields cannot resolve the global interdependencies governing pressure balance across internal loops in meshed networks. Moreover, most existing surrogates are fully data-driven, sacrificing physical guarantees on thermal dynamics. This work presents a model that overcomes these limitations by reformulating the problem in terms of cycle flows and using a condensed graph representation that allows shallower architectures to capture global dynamics. This learned hydraulic solver is coupled with a physics-based Lagrangian thermal model optimized for parallel execution. The framework is validated on the district heating network of Verbier, Switzerland, comprising 165 substations and six internal loops per line. On synthetic test data spanning the full operational envelope, the surrogate matches physics-based solver accuracy in over 97% of cases, maintaining pressure residuals below 100 Pa, while achieving an 85-fold speedup (2187 versus 25 hydraulic steps per second). Validation against real-world monitoring data shows thermal predictions comparable to high-fidelity simulation tools with the speed of fully data-driven models. The resulting throughput and accuracy make the proposed framework suitable for applications requiring high simulation speed while maintaining physical fidelity.

* Corresponding author.
E-mail addresses: roberto.boghetti@epfl.ch, roberto.boghetti@idiap.ch (R. Boghetti).

https://doi.org/10.1016/j.egyai.2026.100721
Received 14 November 2025; Received in revised form 5 February 2026; Accepted 12 March 2026
Available online 13 March 2026

Nomenclature

Deep Learning

$\bigoplus$	Aggregation operator in GNN
$\boldsymbol{\Theta}_e$	Learnable weight matrix for edge features
$\boldsymbol{A}'$	Incidence matrix of the condensed graph
$\boldsymbol{e}$	Edge feature vector for the GNN
$\boldsymbol{h}_u^t$	Hidden representation of node u at GNN layer t
$\boldsymbol{X}$	Node feature matrix for the GNN
$\boldsymbol{x}_u$	Feature vector for node u
$\mathcal{G}'$	Condensed graph for the GNN model
$\mathcal{L}$	Loss function
σ	Sigmoid function
k	Normalization constant for physics loss
$M_{\boldsymbol{\Theta}}$	Message function in a GNN
$U_{\boldsymbol{\Theta}}$	Update function in a GNN
w_1, w_2	Dynamic weights for loss components

Generic

$\boldsymbol{A}$	Vertex-edge incidence matrix
$\boldsymbol{B}$	Cycle-edge incidence matrix
δ	Tolerance threshold for residuals
ϵ	Small constant for numerical stability
$\mathcal{C}$	Set of cycles in a cycle basis
$\mathcal{E}$	Set of edges in a graph
$\mathcal{G}$	Directed graph representing the DHN
$\mathcal{I}_i$	Set of incoming edges at node i
$\mathcal{N}(u)$	Set of neighbors of node u
$\mathcal{O}_i$	Set of outgoing edges at node i
$\mathcal{V}$	Set of vertices in a graph
$\odot$	Hadamard (element-wise) product
N_c, N_e, N_p, N_s, N_V	Counts (cycles, epochs, pipes, samples, vertices)
n_e	Current training epoch

Physics Modeling

α	Blending factor for temperature calculation
$\bar{\theta}$	Volume-weighted average temperature
$\boldsymbol{\phi}$	Function mapping mass flow to pressure difference
$\boldsymbol{\Delta p}$	Vector of edge pressure differences
$\dot{\boldsymbol{m}}$	Vector of mass flows
$\boldsymbol{\ell}$	Vector of pipe lengths
$\tilde{\boldsymbol{m}}$	Vector of cycle flows
$\boldsymbol{d}$	Vector of pipe diameters
$\boldsymbol{f}_D$	Vector of Darcy–Weisbach friction factors
ψ	Temperature transfer function
ρ	Fluid density
θ	Temperature
r_{ij}	Pressure residual of cycle j in sample i
V	Volume

Subscripts

b	Indicating a boundary loop-related entity
$t(j)$	Indicating the starting node of edge j (tail)
in	Indicating an inlet or incoming value
l	Indicating an internal loop-related entity
max	Indicating a maximum value
out	Indicating an outlet or outgoing value
p	Indicating an entity related to pipes in internal loops
$phys$	Indicating the physics-informed component of the loss
reg	Indicating the regression component of the loss
$scaled$	Indicating a scaled value
$slack$	Indicating slack nodes

Superscripts

$pred$	Indicating a value related to the surrogate model's predictions
$'$	Indicating an entity related to the condensed graph $\mathcal{G}'$
(1)	Indicating an output from the first stage of the model
t	Indicating a GNN layer index

Abbreviations

Acc	Accuracy
CVRMSE	Coefficient of Variation of the RMSE
DACVRMSE	Daily amplitude Coefficient of Variation of the RMSE
DHC	District Heating and Cooling
DHN	District Heating Network
ELU	Exponential Linear Unit
GNN	Graph Neural Network
HCVRMSE	Hourly-averaged Coefficient of Variation of the RMSE
HEM	Holomorphic Embedding Method
LHS	Latin Hypercube Sampling
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multi-Layer Perceptron
MPS	Message-Passing Stack
NMBE	Normalized Mean Bias Error
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
SM	Surrogate Model

1. Introduction

District Heating and Cooling (DHC) systems deliver thermal energy to multiple buildings (consumers) through underground networks of pipes. These networks typically circulate fluid through two primary lines: a supply line delivers it from producers (e.g., heat plants, data centers or chillers) to consumers, and a parallel return line brings it back. Both lines branch out to serve individual buildings, where substations transfer heat via non-mixing heat exchangers.

DHC systems are critical for sustainable development scenarios [1] due to their high energy efficiency, integration with renewable sources and suitability for densely urbanized areas where individual heat pumps are impractical [2]. Several studies propose new methods for optimizing their design [3] and operations [4], showing significant cost and energy savings opportunities. These include the introduction of flexibility strategies, such as thermal energy storage [5] and demand-side management [6]. Such studies rely on simulation models to evaluate the performance of different design or operational choices

under a wide range of conditions, such as varying demand profiles [7], technologies and constraints [8]. The validity of these optimizations directly depends on the underlying simulation model, which imposes trade-offs between speed and accuracy, as evidenced by analyses of model fidelity [9], simplification strategies [10], and recent reviews on modeling approaches [11].

A simulation covers a defined period of time, discretized into time steps. Each step typically decouples into sequential subproblems: a steady-state fluid dynamics simulation to estimate mass flow and pressure, followed by a transient thermal simulation to compute temperatures. For meshed networks (i.e., with internal loops), the hydraulic simulation requires the solution of a non-linear equation system, typically approached with iterative methods. Because of its steady-state nature, each time step can be solved independently, without relying on previous results, allowing parallel computation of multiple time steps. In contrast, the thermal simulation addresses a linear problem, but requires sequential time step computation due to time-dependence [12].

Simulation systems often face speed-accuracy trade-off. Faster simulation models use simplifications that increase uncertainty but allow broader exploration during optimization, while accurate models are computationally intensive, thus limiting the variety of scenarios that can be evaluated, especially in complex systems [10]. To this end, the iterative nature of hydraulic solvers represents a bottleneck: while a single steady-state hydraulic simulation of a network with several hundred nodes typically completes within seconds, this cost scales poorly for simulations with a long time horizon and when embedded in optimization or uncertainty quantification loops. In such applications, exhaustive problem space exploration requires a large number of forward evaluations, making the cumulative computational burden prohibitive even when individual solves are relatively inexpensive. This trade-off is particularly relevant for real-time operational optimization, where the operational parameters of DHC systems are continuously adjusted [13].

A common strategy to reduce the computational burden is to use network reduction methods that merge several pipes or branches into a single element thus reducing the number of equations to be dealt with [14]. However, this approach does not directly address the bottlenecks of the simulation, which are its iterative and time-dependent nature, and negatively impacts the simulation accuracy.

Recent works have also explored alternative solution strategies aimed at avoiding iterative solves, most notably the Holomorphic Embedding Method (HEM), which avoids the convergence sensitivity of iterative methods. HEM has been successfully applied to power flow problems, where the governing equations admit a holomorphic embedding in the complex voltage variables [15]. However, a direct extension to the hydraulics of DHC networks is not trivial, as pressure loss relations violate the holomorphicity assumptions underlying the method [16]. Consequently, existing applications have only been restricted to radial topologies, as in [17,18], or relied on warm guesses through Newton–Raphson [16], substantially diminishing the computational advantages.

Alternatively, surrogate models based on Machine Learning (ML) have been proposed in several applications related to energy simulations, providing lower computational effort than physics-based models while retaining high accuracy [19]. For DHC systems specifically, ML has mostly been applied to demand forecasting [20], fault detection [21], and automatic control [22], while surrogate modeling of distribution networks remains under-researched.

1.1. Literature review

Among early works on DHC surrogates, Guelpa and Verda [12] proposed a simplified model for the hydraulic simulation of a meshed District Heating Network (DHN). They removed one pipe from each internal loop of the network, effectively decomposing its topology into a tree and a co-tree structure, which are then solved separately. The co-tree is solved first: the mass flows in the removed pipes are estimated using a linear model based on the processed mass flows at the heat plants. These estimated flows are then used as additional boundary conditions to solve the tree part. While achieving three orders of magnitude speed-up, this approach is inherently network-specific and cannot generalize, as the linear model learns relationships valid only for that specific topology and data distribution. If the network layout is altered, or a different set of pipes is removed for the decomposition, the entire model must be rebuilt.

Rodrigue et al. [23] proposed to reduce the DHN topology by replacing predefined clusters of substations with single nodes, and estimate the temperatures of connecting pipe water using Artificial Neural Networks, while modeling remaining dynamics physics-based. Applied to a synthetic network, the proposed hybrid approach reduced simulation time by 27%, with negligible power estimation errors at heat sources. However, performance gains are not on the same magnitude as other methods due to partial physics-based reliance, and pipe-level outputs in clusters are not computed.

Boussaid et al. [24] investigated the use of a surrogate model based on Graph Neural Networks (GNNs) [25] with graph attention [26] for predicting the return temperature of nodes in a toy network with 1 heat plant and 6 consumers. The model achieved a Root Mean Square Error (RMSE) of 0.3 K on the test data while reducing the computational time by a factor of 10^4. A followup work [27] extended the model to predict the nodal temperature and pressure for both the supply and return nodes, improving the architecture with recurrent layers (Gated Recurrent Units, [28]), graph convolutional layers [29], and skip connections. The new architecture achieved a RMSE below 0.5 K on the temperature prediction and an average RMSE of 0.002 and 0.06 bar (200 and 6000 Pa) on the pressure estimation of supply and return nodes respectively. Scaling the model to larger networks showed increased prediction error with node count growth. Increasing the complexity of the model partly mitigated the loss of performance, at the cost of inference speed. For a 100-node DHN simulated over 54 days with 20-minute time steps, the inference time was approximately 42 s, or around 92 steps per second. A notable limitation is that the GNN architecture provides only localized network views, which may be inadequate for simulating large meshed networks where hydraulic behavior depends on entire loop topology, as evidenced by the relatively high errors reported for pressure estimation. While the size of the local view could be expanded by adding more GNN layers, this would come at the cost of slower inference and a potential loss of detail in the learned representations. Despite these limitations, GNN-based surrogate models offer key advantages over standard ANNs. First, they explicitly encode relational inductive biases by operating directly on the DHN graph, thereby making the network topology an integral part of the model rather than an implicit feature of the data. Second, their parameterization is independent of the number of nodes, allowing a single architecture to be applied across networks of varying size and structure. This property makes GNNs particularly attractive for transferable or reusable surrogate modeling, where the same model class, if not the same trained weights, can be deployed across different case studies, thus moving toward generalized surrogate solvers.

Finally, de Giuli et al. [30] proposed PI-RNN, an architecture interconnecting distinct Recurrent Neural Networks (RNNs) to form a graph resembling DHN topology, requiring separate RNNs for each consumer (load) plus one for the return line. The model only has outputs for the load nodes, thus not resolving the hydraulic of the piping network. Furthermore, while promising on the nine-node synthetic example, this approach is highly topology-coupled, making scaling or adaptation difficult without re-designing the entire model.

1.2. Aim and contributions

The literature review reveals a gap in surrogate models for DHN simulation. A key challenge is capturing the global nature of hydraulic physics in meshed networks, where pressure and flow at any point are determined by the balance of the entire network. Most existing approaches are not designed to capture such global relationships. Additionally, they commonly exhibit other limitations: they are often topology-dependent and must be redesigned when the layout changes; some provide only partial predictions, omitting pipe-level or element-wise outputs; and most scale poorly to large networks. GNN-based surrogates are particularly promising, as they naturally address several of these limitations. However, as they rely on localized information flow, the fundamental challenge of modeling network-wide hydraulic dependencies persists.

This work addresses this gap through the following contributions:

- A surrogate modeling formulation for meshed DHN hydraulics based on cycle flows rather than individual pipe flows, reducing the number of unknowns, enforcing mass conservation by construction, and making loop interdependencies explicit.
- A condensed representation of the DHN graph designed to shorten the topological distance between hydraulically coupled elements and as such allowing GNNs to capture long-range dependencies with only few layers.
- A two-stage GNN-based surrogate model mimicking the correction structure of classical iterative solvers while retaining fixed inference cost and differentiability.
- A hybrid simulation framework that couples the proposed hydraulic surrogate with a GPU-optimized Lagrangian thermal model, matching the throughput of surrogate models while being fully physics-based.
- A comprehensive validation of the approach on both synthetic and real-world monitoring data.

While this work focuses on DHNs, the approach can be adapted to cooling networks.

1.3. Organization of the paper

The remainder of this paper is structured as follows. Section 2 formalizes the hydraulic simulation problem using a cycle-based approach. Section 3 details the proposed GNN-based surrogate hydraulic model, outlining its input–output structure, architecture, and loss function. The physics-based thermal model used to complete the simulation pipeline is described in Section 4. In Section 5, we present the case study, data generation process, model configurations, and evaluation protocol. The performance of the model on synthetic data is presented in Section 6, followed by an ablation study on the loss function in Section 7. Section 8 demonstrates the model's effectiveness on real-world monitoring data. Finally, we discuss the implications of our findings in Section 9 and conclude the paper in Section 10.

2. Problem statement

2.1. Problem setup

The hydraulic state of a DHN is only partially observable, with sensors typically at producers and consumers, but not within the pipe network. The simulation task is thus to infer the complete network state from this limited set of measurements. To do so, we assume that we are given two categories of information:

- **Static Network Structure**: physical layout and pipe properties (length and diameter) are known and fixed.
- **Boundary Conditions**: observations at producers and consumers define the hydraulic problem at each time step, fully constraining the system. These are: (i) mass flow rates at consumers and secondary producers, (ii) pressure difference across the main producer.

2.2. Modeling assumptions and limitations

The proposed surrogate model relies on the following assumptions, which define the scope of its applicability:

1. **Physical Model:** the system is modeled under steady-state conditions, neglecting transient pressure waves. The fluid is assumed incompressible with constant density and viscosity. The network is treated as a closed-loop system with no leakage.
2. **Topological Constraints:** the model targets meshed, third-generation DHNs where the supply and return lines are topologically distinct. The physical layout and pipe properties (roughness, diameter, length) are assumed known and fixed.
3. **System Observability:** the hydraulic problem is assumed well-posed, meaning boundary conditions are fully known at each time step.
4. **Operational Bounds:** as a data-driven approach, the surrogate model is valid for interpolation within the operational limits defined by the training data. Performance is not verified for boundary conditions that require extrapolation outside this distribution.

These assumptions align with standard practice for third-generation DHN simulation. The framework is best suited for applications requiring numerous forward evaluations, such as long-horizon simulations, design optimization, or scenario analysis, where the cumulative cost of iterative solvers becomes prohibitive. In particular, when demand profiles are predetermined, the decoupling of hydraulic and thermal simulations allows the entire hydraulic horizon to be solved in parallel, maximizing computational gains. For applications involving tight thermal–hydraulic control loops, the advantage is instead limited to parallel execution of multiple independent scenarios rather than acceleration of a single simulation. For networks with fundamentally different operating principles (e.g., fourth-generation low-temperature systems with bidirectional flows), the framework would require adaptation.

2.3. Hydraulic problem formulation and physics-based modeling

Following the formalization in [31], depicted in Fig. 1, the DHN is modeled as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}|$ vertices (also called nodes), representing junctions, and $|\mathcal{E}|$ edges, representing pipes, substations and heat sources. We construct the graph such that the subset of edges associated with pipes forms two connected subgraphs, one for the supply and one for the return line, and that the vertices can only have a total degree of 2 or 3 (degree 2 if connected to a non-pipe edge). Note that this representation leads to a graph with several cycles, paths of unique edges starting and ending at the same vertex (Fig. 1, right, and Fig. 2, left). These cycles represent physical loops within the DHN, including large-scale circuits where fluid circulates between producers and consumers, and internal loops existing entirely within either the supply or the return line, typical of meshed networks. The latter are the main challenge for hydraulic simulations as they introduce multiple paths for water to flow. While frequently used interchangeably, we will use the term *loop* for the physical structure and *cycle* for its abstract graph representation. Pipe properties are stacked into vectors $\boldsymbol{d} \in \mathbb{R}^{|\mathcal{E}|}$ and $\boldsymbol{\ell} \in \mathbb{R}^{|\mathcal{E}|}$ for diameter and length respectively, with zero values for non-pipe elements. Observed mass flow rates, which constitute hydraulic boundary conditions at secondary heat sources or substations, are similarly collected in the vector $\dot{\boldsymbol{m}}_{\text{set}} \in \mathbb{R}^{|\mathcal{E}|}$, with zero entries in correspondence of pipe and main producer edges.

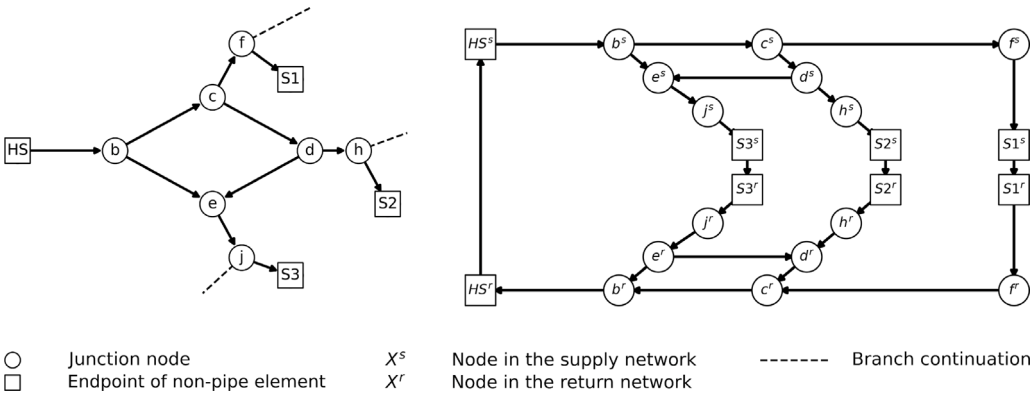


Fig. 1. Example of a DHN and its graph representation. **Left:** GIS-style DHN layout, where supply and return lines are represented as a single connection. The heating plant is labeled HS, substations as S, and junctions with lowercase letters. **Right:** Graph representation used in this work, where supply and return lines are explicitly separated.

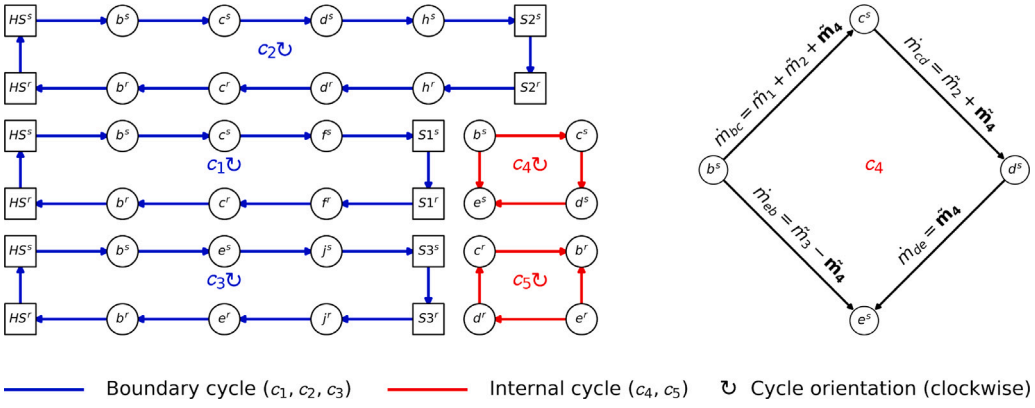


Fig. 2. Cycle-based decomposition of mass flows. **Left:** Cycles of the graph in Fig. 1. Boundary cycle flows are fully determined by the mass flow given at their unique substation. **Right:** Superposition principle for cycle c_4. Edge mass flows are obtained by summing signed contributions from the cycle flows $\tilde{m}_i$. The contribution of $\tilde{m}_4$ is shown in bold, signs reflect the cycle-edge incidence matrix.

Given $\mathcal{G}$, $\boldsymbol{d}$, $\boldsymbol{\ell}$ and $\dot{\boldsymbol{m}}_{\text{set}}$, the aim of the simulation is to estimate the vectors of edge mass flows $\dot{\boldsymbol{m}} \in \mathbb{R}^{|\mathcal{E}|}$ ($\mathrm{kg\,s^{-1}}$) and edge pressure differences $\boldsymbol{\Delta p} \in \mathbb{R}^{|\mathcal{E}|}$ (Pa) from known boundary conditions. Since in general the pressure difference is a function of the mass flow, i.e.,

$$\boldsymbol{\Delta p} = \boldsymbol{\phi}(\dot{\boldsymbol{m}}), \tag{1}$$

it is sufficient to determine the set of valid mass flows in the pipe elements.

Under the hypotheses of incompressible fluid, constant fluid properties and assuming a closed-loop system (i.e., no water injection or leak), the hydraulic simulation is governed by the two Kirchhoff's laws [32]:

$$\boldsymbol{A}\dot{\boldsymbol{m}} = \boldsymbol{0}, \tag{2}$$

$$\boldsymbol{B}\boldsymbol{\Delta p} = \boldsymbol{0}. \tag{3}$$

Here, $\boldsymbol{A} \in \{-1, 0, 1\}^{|\mathcal{V}|\times|\mathcal{E}|}$ denotes the *vertex-edge incidence matrix*, and $\boldsymbol{B} \in \{-1, 0, 1\}^{|\mathcal{C}|\times|\mathcal{E}|}$ denotes the *cycle-edge incidence matrix* associated with the set of cycles $\mathcal{C}$. In essence, Eq. (2) enforces mass conservation at each vertex, ensuring that the net mass flow entering and leaving any junction is zero. Eq. (3) states that the algebraic sum of pressure gains and losses around any closed cycle must be zero. It is sufficient to enforce this condition on a *cycle basis*,[1] since any other cycle in the graph can be formed as a linear combination of the basis cycles, so that in practice, the size of $\mathcal{C}$ can be reduced to $|\mathcal{C}| = |\mathcal{E}|-|\mathcal{V}|+1$. Rather than solving for the mass flow in every pipe individually, we reformulate the problem in terms of cycle flows to reduce the number of unknowns. Each cycle in the chosen basis $\mathcal{C}$ is assigned a scalar cycle flow, forming the vector of unknowns $\tilde{\boldsymbol{m}}$. The mass flow in any given edge is then the algebraic sum of the flows from all cycles that contain that edge [32]:

$$\dot{\boldsymbol{m}} = \boldsymbol{B}^{\top}\tilde{\boldsymbol{m}}. \tag{4}$$

Then, substituting Eq. (4) into the Kirchhoff equations and using the relation in Eq. (1), we obtain:

$$\boldsymbol{A}(\boldsymbol{B}^{\top}\tilde{\boldsymbol{m}}) = \boldsymbol{0}, \tag{5}$$

$$\boldsymbol{B}\boldsymbol{\phi}(\boldsymbol{B}^{\top}\tilde{\boldsymbol{m}}) = \boldsymbol{0}. \tag{6}$$

Furthermore, since a cycle is by definition a closed path, the incidence and cycle basis matrices are orthogonal [32]:

$$\boldsymbol{A}\boldsymbol{B}^{\top} = \boldsymbol{0}. \tag{7}$$

As a consequence, any flow vector constructed from cycle flows, as in Eq. (4), automatically satisfies mass conservation (Eq. (5)). The problem thus reduces to finding a cycle flow vector $\tilde{\boldsymbol{m}}$ that satisfies both the pressure balance in Eq. (6) and the given boundary conditions on mass flow.

The problem can be further simplified by using a structured cycle partitioning that exploits the fact that boundary conditions uniquely determine the mass flow in branches of the network outside internal

[1] A cycle basis is a minimal set of cycles in a graph such that any other cycle can be expressed as a combination of them. A graph can have multiple valid cycle bases.

loops. In particular, if the cycle basis C is chosen such that each basis cycle either:

1. Forms internal loops within the pipe network and does not contain any non-pipe element (subset C_l), or
2. Contains exactly two non-pipe elements, where one element is common to all cycles in this subset and the other unique to each individual cycle (subset C_b)

then the cycle basis can be partitioned into a subset of boundary cycles (subscript b) and a subset of internal loops (subscript l). Analogously, the cycle-edge incidence matrix $\boldsymbol{B}$ can be partitioned into a *boundary cycle matrix* $\boldsymbol{B}_\mathrm{b}$ and an *internal loop cycle matrix* $\boldsymbol{B}_\mathrm{l}$, corresponding to the subsets C_b and C_l, respectively. Under this partitioning, the system in Eq. (6) becomes:

$$\boldsymbol{B}\boldsymbol{\phi}(\boldsymbol{B}_\mathrm{b}^\top \tilde{\boldsymbol{m}}_\mathrm{b} + \boldsymbol{B}_\mathrm{l}^\top \tilde{\boldsymbol{m}}_\mathrm{l}) = \boldsymbol{0}\,. \tag{8}$$

Because each boundary cycle contains exactly one unique non-pipe element whose mass flow is a known boundary condition, the boundary cycle flow vector $\tilde{\boldsymbol{m}}_\mathrm{b}$ is uniquely determined as $\tilde{\boldsymbol{m}}_\mathrm{b} = \boldsymbol{B}_\mathrm{b}\dot{\boldsymbol{m}}_\mathrm{set}$, where the product reduces to a selection since each row of $\boldsymbol{B}_\mathrm{b}$ has exactly one non-zero entry at a position where $\dot{\boldsymbol{m}}_\mathrm{set}$ is also non-zero. This has two important consequences:

1. The total mass flow through the main producer (shared across all boundary cycles) is now fixed as the sum of all boundary cycle flows, accounting for their orientation. Its pressure lift is then immediately given by the function ϕ. In this work, it is instead imposed directly as a boundary condition.
2. Once the internal cycle flows $\tilde{\boldsymbol{m}}_\mathrm{l}$ are determined, the pressure differences across all other non-pipe elements (each uniquely associated with a boundary cycle) follow directly from Kirchhoff's second law, which is thus automatically satisfied in the boundary cycles.

Consequently, the only remaining unknown in the system is the vector of internal cycle flows $\tilde{\boldsymbol{m}}_\mathrm{l}$, which is found by enforcing Kirchhoff's second law on internal cycles. Therefore, the system in Eq. (8) reduces to:

$$\boldsymbol{B}_\mathrm{l}\boldsymbol{\phi}(\boldsymbol{B}_\mathrm{b}^\top \tilde{\boldsymbol{m}}_\mathrm{b} + \boldsymbol{B}_\mathrm{l}^\top \tilde{\boldsymbol{m}}_\mathrm{l}) = \boldsymbol{0}\,. \tag{9}$$

This reduces the number of equations to be solved iteratively to $|C_l|$. An illustration of this cycle-based decomposition is given in Fig. 2.

Since the unknown part of Eq. (9) only involves internal cycles, and these are composed exclusively of pipe elements, it is sufficient to define the pressure loss function ϕ for pipes. Neglecting hydrostatic pressure, this is a nonlinear function defined as:

$$\boldsymbol{\Delta p}_\mathrm{l} = \frac{1}{4\pi^2\rho}\left(\boldsymbol{\ell}_\mathrm{l} \odot \boldsymbol{f}_\mathrm{D,l} \odot (\boldsymbol{d}_\mathrm{l}/2)^{-5} \odot |\dot{\boldsymbol{m}}_\mathrm{l}| \odot \dot{\boldsymbol{m}}_\mathrm{l}\right)\,. \tag{10}$$

Here, $\boldsymbol{\ell}_\mathrm{l}$ and $\boldsymbol{d}_\mathrm{l}$ are the vectors of lengths and diameters of the internal loop pipes, $\dot{\boldsymbol{m}}_\mathrm{l}$ is the corresponding mass flow vector, and $\boldsymbol{f}_\mathrm{D,l}$ contains the Darcy–Weisbach friction factors. The density ρ is assumed constant throughout the network. The Hadamard product $\odot$ is applied elementwise. Friction factors are computed in this work using the same relationships as in [31]. With the pressure loss model and reduced system of equations defined, we can solve for the internal cycle flows $\tilde{\boldsymbol{m}}_\mathrm{l}$ using the Newton–Raphson method. However, this iterative procedure is the main computational bottleneck, motivating the use of a learned surrogate model to approximate the solution efficiently, as described in the next section.

3. Surrogate modeling

This section details the GNN-based surrogate model developed to solve the hydraulic problem introduced in Section 2 without costly Newton–Raphson iterations. The overall procedure is summarized in Fig. 3. For a given set of boundary conditions, the process is as follows:

1. Derive the boundary cycle flows $\tilde{\boldsymbol{m}}_\mathrm{b} = \boldsymbol{B}_\mathrm{b}\dot{\boldsymbol{m}}_\mathrm{set}$, and compute the corresponding boundary edge flows: $\dot{\boldsymbol{m}}_\mathrm{b} = \boldsymbol{B}_\mathrm{b}^\top \tilde{\boldsymbol{m}}_\mathrm{b}$ (Fig. 4, center).
2. Construct a condensed graph $\mathcal{G}'$, whose structure is described by the incidence matrix $\boldsymbol{A}'$ along with vectors of edge features $\boldsymbol{e}$, and node features $\boldsymbol{\ell}'$ and $\boldsymbol{d}'$. In $\mathcal{G}'$, each of the N_V nodes corresponds either to a pipe or to an internal cycle, making the cycle–pipe interactions within internal loops explicit (Fig. 4, right).
3. Prepare training/inference samples $S = (\boldsymbol{A}', \boldsymbol{X}, \boldsymbol{e})$, with $\boldsymbol{X} = \left[\boldsymbol{x}_u^\top\right]_{u=1}^{N_\mathrm{V}} \in \mathbb{R}^{N_\mathrm{V}\times 3}$, where each node feature vector is defined as $\boldsymbol{x}_u = [\dot{m}'_{b,u}, d'_u, \ell'_u]^\top$. The vector $\dot{\boldsymbol{m}}'_\mathrm{b}$ is adapted from $\dot{\boldsymbol{m}}_\mathrm{b}$ to match the cycle-based representation of the condensed graph $\mathcal{G}'$.
4. For training and evaluation, compute the ground-truth internal cycle flows $\tilde{\boldsymbol{m}}_\mathrm{l}$ through the fully physics-based simulation described in Section 2. These are not available during deployment.
5. Use the surrogate model to estimate the internal cycle flows $\tilde{\boldsymbol{m}}_\mathrm{l}^\mathrm{pred}$ from a tuple S.
6. Reconstruct the edge mass flow vector for the whole network: $\dot{\boldsymbol{m}}^\mathrm{pred} = \dot{\boldsymbol{m}}_\mathrm{b} + \boldsymbol{B}_\mathrm{l}^\top \tilde{\boldsymbol{m}}_\mathrm{l}^\mathrm{pred}$.

The following subsections detail the condensed graph representation (Section 3.1), the input–output formulation of the problem (Section 3.2), the GNN framework (Section 3.3), the surrogate model architecture (Section 3.4), and the loss function used for training (Section 3.5).

3.1. Condensed graph representation

As discussed in Section 1.1, the main challenge for GNN-based hydraulic surrogates lies in their limited receptive field: pressure and flow distributions in meshed networks depend on global balance, but GNNs operate locally on the physical network topology. Adding more layers to expand the receptive field reduces throughput and, as discussed in Section 3.3, causes oversmoothing, where node representations become progressively indistinguishable.

To address this, we construct a condensed graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ that reduces the topological distance between hydraulically coupled elements while retaining only the structure required to solve the nonlinear system in Eq. (9). Specifically, we design $\mathcal{G}'$ to mirror the incidence structure encoded by $\boldsymbol{B}_\mathrm{l}$. Nodes represent: (i) each pipe belonging to at least one internal loop (N_p pipe nodes), and (ii) each internal cycle (N_c cycle nodes). Edges connect pipe nodes to the cycle nodes in which they participate, yielding a bipartite graph with $N_\mathrm{V} = N_\mathrm{p} + N_\mathrm{c}$ vertices. Elements not participating in internal loops (producers, consumers, and branch pipes) are excluded, and pipe–pipe connections are omitted as they introduce no additional degrees of freedom in the cycle flow formulation. This structure ensures that any two pipes sharing a cycle are separated by only two hops, regardless of their physical distance in the network.

The incidence matrix $\boldsymbol{A}'$ encodes these connections but discards the sign information present in $\boldsymbol{B}_\mathrm{l}$. We recover it through an edge feature vector $\boldsymbol{e} \in \{-1, +1\}^{|\mathcal{E}'|}$, where each entry indicates whether a cycle's positive direction contributes positively or negatively to the corresponding pipe's mass flow.

Finally, to retain the physical information necessary for hydraulic calculations, we map the static pipe properties from the edge space of the original graph ($\mathbb{R}^{|\mathcal{E}|}$) to the node space of the condensed graph ($\mathbb{R}^{|\mathcal{V}'|}$). We adopt the prime notation ($'$) to distinguish these new node attribute vectors from the original edge attribute vectors. Specifically, for each pipe node u in $\mathcal{G}'$ corresponding to a physical pipe edge (v, w) in $\mathcal{G}$, we assign the length $\ell'_u = \ell_{vw}$ and diameter $d'_u = d_{vw}$ (m). For the virtual cycle nodes, which have no physical dimensions, these properties are explicitly set to zero.

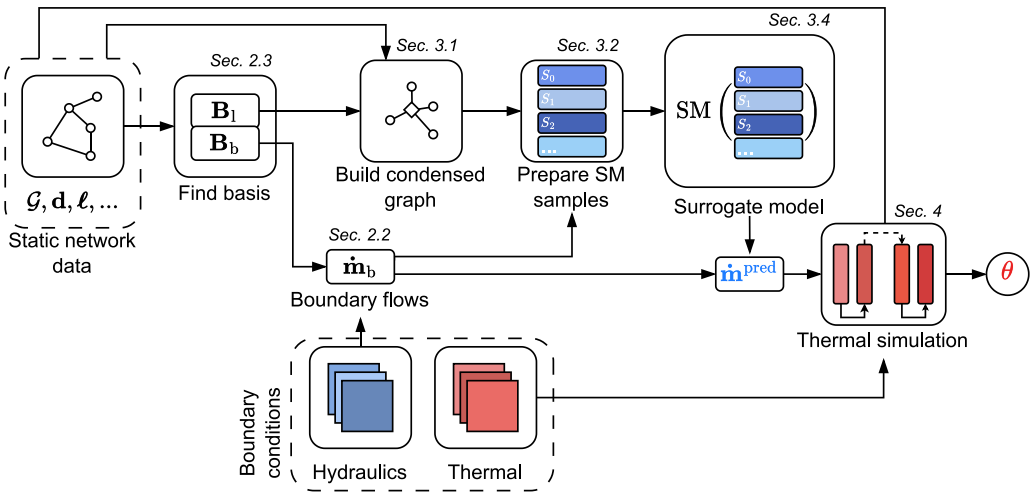


Fig. 3. Overview of the simulation pipeline and relevant sections. The network topology is used to extract a cycle basis and define the structure and edge features of the condensed graph, while static pipe properties provide its node attributes. These, together with the boundary flows computed at each time step from hydraulic boundary conditions, form the input to the Surrogate Model (SM in the figure). The model predicts internal cycle flows, used to reconstruct full pipe mass flows $\dot{\boldsymbol{m}}^{\text{pred}}$, which are then passed, along with thermal boundary conditions, to a physics-based thermal model.

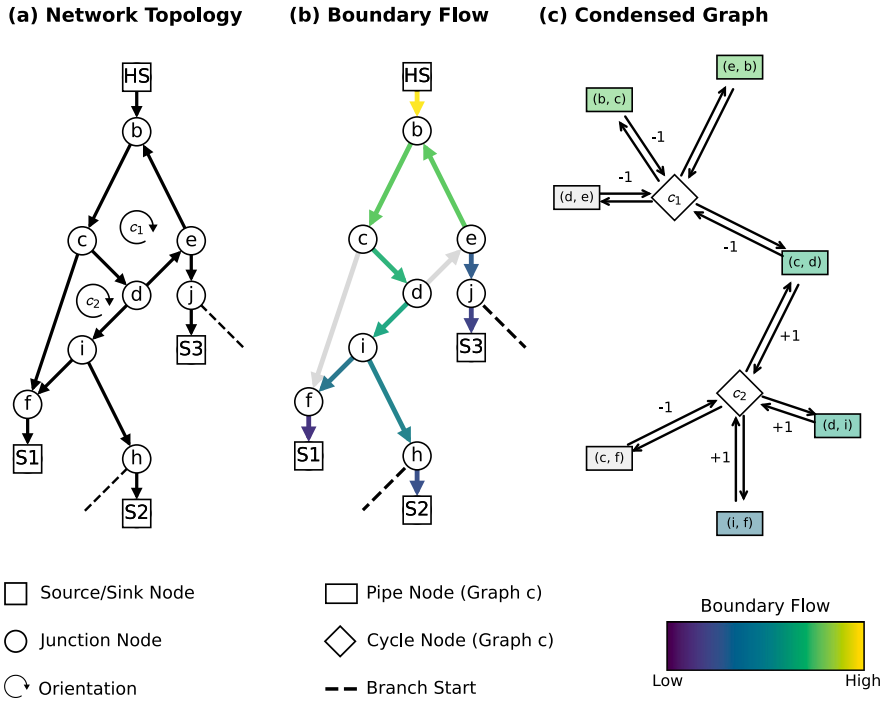


Fig. 4. Construction of the condensed graph. **Left:** Original network graph showing a producer (HS), substations (S), and junctions (lowercase letters). **Center:** Boundary mass flows are computed based on boundary conditions, with color intensity qualitatively representing flow magnitude (gray indicating the edge is not part of the tree). **Right:** In the condensed graph, each cycle is a virtual node connected to its constituent pipe nodes, which retain boundary flow magnitude as a feature. Edges are assigned an orientation attribute.

3.2. Input–output structure of the surrogate model

Given the condensed graph, its static properties, and a set of boundary conditions, we generate training samples consisting of model inputs and corresponding ground-truth targets. Note that targets are only available during training and evaluation; they are not accessible at inference time.

First, we define the dynamic input component. We compute the boundary edge flows $\dot{\boldsymbol{m}}_{\text{b}}$ on the original graph as described in Section 2 and map them to the node space of $\mathcal{G}'$ to form the vector $\dot{\boldsymbol{m}}'_{\text{b}}$ ($\text{kg}\,\text{s}^{-1}$), following the same approach used to build $\boldsymbol{d}'$ and $\boldsymbol{\ell}'$ in Section 3.1. The node feature matrix $\boldsymbol{X}$ is then constructed by stacking a feature vector $\boldsymbol{x}_u = [\dot{m}'_{\text{b},u}, d'_u, \ell'_u]^\top$ for each node u.

The learning target is then the vector of internal cycle flows $\tilde{\boldsymbol{m}}_{\text{l}} \in \mathbb{R}^{N_{\text{c}}}$, defined over the cycle nodes. Each training sample thus consists of the tuple $S = (\boldsymbol{A}', \boldsymbol{X}, \boldsymbol{e}, \tilde{\boldsymbol{m}}_{\text{l}})$, where $\boldsymbol{e}$ encodes edge orientations as described in the previous section. Note however that each training sample only differs in the values of $\dot{m}'_{\text{b},u}$ in the input, and $\tilde{\boldsymbol{m}}_{\text{l}}$ at the output, all other variables being defined according to the DHN and graph structure.

Because the supply and return lines do not share any pipe, $\mathcal{G}'$ is naturally disconnected, forming two independent subgraphs. While physically identical, we compute their cycle bases independently, which makes their corresponding portions of the global incidence matrix ($\boldsymbol{A}'$) and edge-feature vector ($\boldsymbol{e}$) different. We treat these structurally distinct subgraphs as separate samples to augment our training data. For notational simplicity, however, we will refer to these inputs with the unified symbols $\boldsymbol{A}'$ and $\boldsymbol{e}$ in the following sections.

3.3. Graph neural networks

GNNs are deep learning models for graph-structured data. They are broadly divided into *spectral* methods, which rely on the graph Laplacian and operate in a frequency-like domain, and *spatial* methods, which define operations directly on nodes and their neighborhoods. In particular, the latter use a *message-passing* mechanism, where nodes aggregate information from their neighbors to update their own representations.

In this work we propose to use spatial GNNs for our task. The main motivation is that in our condensed graph, each cycle vertex has to aggregate information from its connected pipes, which can naturally and conceptually be captured by message-passing operations. In addition, spatial methods are scalable, flexible, and well suited to *multi-graph settings*, where the model is trained across different network topologies, allowing future investigations into model transferability and generalization.

The inference process in spatial GNNs is done by iterating message-passing operations acting on some hidden representation, which is achieved by stacking several message-passing layers in the neural network. Such iterations allow the model to capture broader dependencies between data measures and identify more expressive patterns. More formally, each iteration or layer is working as follows. We denote by $\boldsymbol{h}_u^t$ the hidden representation associated with node u at layer t, where $\boldsymbol{h}_u^0$ denotes the initial embedding commonly obtained by applying a learnable transformation (e.g., a linear layer) to the raw input features $\boldsymbol{x}_u$. The general message-passing framework Gilmer et al. [33] defines the update from layer t to $t+1$ as:

$$\boldsymbol{h}_u^{t+1} = U_{\boldsymbol{\Theta}}\left(\boldsymbol{h}_u^t, \bigoplus_{v\in\mathcal{N}(u)} M_{\boldsymbol{\Theta}}\left(\boldsymbol{h}_u^t, \boldsymbol{h}_v^t, \boldsymbol{e}_{v,u}\right)\right). \tag{11}$$

Here, $\mathcal{N}(u)$ is the set of neighbors of node u. The message function $M_{\boldsymbol{\Theta}}$ computes messages passed from neighboring nodes v to node u, optionally incorporating edge features $\boldsymbol{e}_{v,u}$. The aggregation operator $\bigoplus$ combines these messages, usually with a permutation-invariant function (e.g., summation), and the update function $U_{\boldsymbol{\Theta}}$ integrates the result with the node's previous representation to produce $\boldsymbol{h}_u^{t+1}$. Message-passing GNNs have a limited receptive field: each message-passing layer allows a node to incorporate information from its immediate neighbors; stacking t layers extends the receptive field to nodes up to t hops away. While increasing the receptive field is desirable in many applications, stacking multiple message-passing layers quickly leads to *oversmoothing*, a phenomenon in which node representations become increasingly similar across the graph, degrading performance in tasks requiring fine-grained distinctions [34].

To mitigate oversmoothing, we adopt the approach proposed by Li et al. [35], which combines improved message aggregation with architectural features designed for deep GNNs. Specifically, we use the DyResGEN block, based on the GENeralized Graph Convolution (GENConv), which defines the message-passing update as:

$$\boldsymbol{h}_u^{t+1} = \mathrm{MLP}\left(\boldsymbol{h}_u^t + \bigoplus_{v\in\mathcal{N}(u)} \left(\mathrm{ReLU}\left(\boldsymbol{h}_v^t + \boldsymbol{\Theta}_e \boldsymbol{e}_{v,u}\right) + \epsilon\right)\right). \tag{12}$$

In the above equation, ReLU is the rectified linear activation function [36], $\boldsymbol{\Theta}_e \in \mathbb{R}^{dim(\boldsymbol{h}_v^t)\times dim(\boldsymbol{e}_{v,u})}$ is a learnable weight matrix used to project edge features into the same dimensionality as node embeddings,

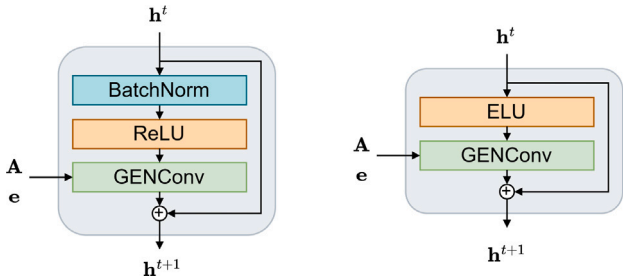


Fig. 5. Left: Original DyResGEN block from [35]. **Right:** Our implementation with ELU. The plus sign $\bigoplus$ indicates element-wise summation.

$\epsilon = 10^{-7}$ is a small positive constant for numerical stability, and the update function, MLP, is a multi-layer perceptron (see [35] for architectural details). The aggregation operation $\bigoplus$ in GENConv is a generalized softmax-weighted summation with a learnable temperature parameter. The DyResGEN block further extends this GENConv formulation by adding residual connections and dynamic normalization. The full block consists of the following sequence: BatchNorm [37] → ReLU → GENConv → residual addition with the input (see Fig. 5, left).

3.4. Surrogate model architecture

The surrogate model builds on the GNN framework introduced above and predicts internal cycle mass flows, $\tilde{\boldsymbol{m}}_\mathrm{l}$, using the condensed graph representation $\mathcal{G}'$ (Section 3.1). Inputs include the graph structure ($\boldsymbol{A}'$), edge features ($\boldsymbol{e}$), and a node feature matrix ($\boldsymbol{X}$). $\boldsymbol{X}$ combines static pipe properties (diameter, length) with dynamic boundary mass flows ($\dot{\boldsymbol{m}}'_\mathrm{b}$). Features for cycle nodes are initialized to zero. For stability and faster convergence, $\boldsymbol{X}$ is column-wise scaled to $\boldsymbol{X}_\mathrm{scaled}$ by the maximum absolute value over the training set (max-abs scaling), such that each feature dimension lies within $[-1, 1]$. Specifically, for each column k of $\boldsymbol{X}$, we compute $X_{\mathrm{scaled},uk} = X_{uk} / \max_i |X_{ik}|$, where the maximum is taken over all training samples. This normalization ensures that boundary mass flows, diameters, and lengths, which span different numerical ranges, contribute comparably to the initial node embeddings.

The graph structure $\boldsymbol{A}'$ and directions $\boldsymbol{e}$ are fixed for each line (e.g., supply), while only the boundary mass flows vary across samples.

The architecture (Fig. 6) consists of two encoder–decoder stages trained jointly end-to-end, where the second stage functions as a residual correction mechanism, refining the prediction of the first stage. This design is motivated by an analogy to iterative numerical solvers: physics-based methods for meshed network hydraulics typically require multiple Newton–Raphson iterations to converge, suggesting that a single feedforward evaluation may be insufficient. Instead of increasing depth within a single stage, the proposed architecture introduces an explicit intermediate prediction that forms a low-dimensional, physically meaningful summary of the network state, discarding latent features that may become diffuse or oversmoothed in deep message-passing networks. Conditioning the second stage on this constrained intermediate solution effectively resets the message-passing process, allowing it to focus on corrective updates. Empirical validation of this design choice is provided in Section 6, where the two-stage architecture significantly outperforms single-stage alternatives of comparable depth. Further investigations on the implications of this choice are given in Appendix.

In Stage 1, the encoder projects $\boldsymbol{X}_\mathrm{scaled}$ into a latent space via a linear layer and processes it with a Series of seven Message-Passing blocks (MPS). Each block is based on DyResGEN but modified by replacing the original BatchNorm–ReLU pre-activation with Exponential Linear Units (ELUs) (Fig. 5, right), following evidence of improved performance [38] confirmed in our preliminary tests. Message direction

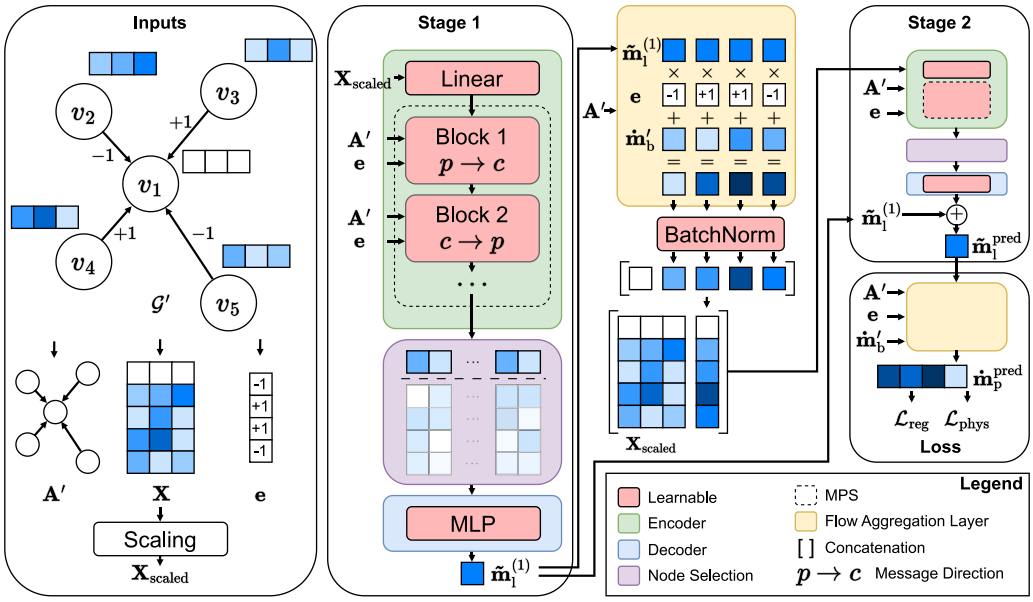


Fig. 6. Overview of the proposed GNN architecture. Inputs are processed to create scaled node features ($\boldsymbol{X}_{\text{scaled}}$), an incidence matrix ($\boldsymbol{A}'$), and edge features ($\boldsymbol{e}$). In Stage 1, an encoder–decoder model produces an initial cycle flow prediction ($\tilde{\boldsymbol{m}}_{\text{l}}^{(1)}$). This prediction is then processed by a Flow Aggregation Layer and combined with the original inputs to form a new feature matrix for Stage 2, which outputs a final, refined prediction ($\tilde{\boldsymbol{m}}_{\text{l}}^{\text{pred}}$).

alternates between blocks: odd blocks propagate from pipes to cycles, even blocks from cycles to pipes, with separate parameters for each block. The edge attribute $\boldsymbol{e}$ is incorporated into these updates to encode the relative orientation of pipes, allowing the model to correctly factor the sign of pressure losses relative to the cycle direction. Since overlapping cycles exchange information only through common pipes, two layers are required for one cycle-to-cycle update; in our case study, five layers are needed for the most distant cycles, so we use seven to ensure full propagation without excessive depth or oversmoothing. The encoder produces context-aware node embeddings, which the decoder (a two-layer MLP with ELU activation) maps to an initial prediction of cycle flows, $\tilde{\boldsymbol{m}}_{\text{l}}^{(1)}$.

This intermediate prediction is then processed by a Flow Aggregation Layer, which projects the cycle flows to pipe flows (analogous to Eq. (4)) and sums them with the corresponding boundary flows. The resulting vector is then batch-normalized, zero-filled at cycle nodes, and concatenated with $\boldsymbol{X}_{\text{scaled}}$. This new feature matrix forms the input to Stage 2, which shares the same structure as Stage 1 but with independent parameters. Its decoder outputs a correction term which is summed with the Stage 1 prediction to yield the final estimate $\tilde{\boldsymbol{m}}_{\text{l}}^{\text{pred}}$.

3.5. Loss function

The model is trained by minimizing a composite loss function that combines a supervised regression term ($\mathcal{L}_{\text{reg}}$) with a physics-informed penalty ($\mathcal{L}_{\text{phys}}$) in a weighted sum:

$$\mathcal{L} = w_1(n_e) \cdot \mathcal{L}_{\text{reg}} + w_2(n_e) \cdot \mathcal{L}_{\text{phys}}\,, \tag{13}$$

where n_e is the current training epoch, and $w_1(n_e)$ and $w_2(n_e)$ are dynamic weights that balance the two objectives. Both loss components operate on the predicted mass flows of the subset of pipe nodes in the condensed graph, denoted with the subscript p, $\dot{\boldsymbol{m}}_{\text{p}}^{\text{pred}}$. These are reconstructed from the model's cycle-flow output, $\tilde{\boldsymbol{m}}_{\text{l}}^{\text{pred}}$ as $\dot{\boldsymbol{m}}_{\text{p}}^{\text{pred}} = [\dot{\boldsymbol{m}}_{\text{b}}]_{\text{p}} + \boldsymbol{B}_{\text{p}}^{\top}\tilde{\boldsymbol{m}}_{\text{l}}^{\text{pred}}$. Here, $\boldsymbol{B}_{\text{p}} \in \mathbb{R}^{N_c \times N_p}$ is a sub-matrix of $\boldsymbol{B}_{\text{l}}$ containing only the columns corresponding to the N_p internal loop pipes.

The regression loss is then defined as the mean squared error between predicted and ground-truth pipe-level mass flows:

$$\mathcal{L}_{\text{reg}} = \frac{1}{N_{\text{s}} N_{\text{p}}} \sum_{i=1}^{N_{\text{s}}} \sum_{j=1}^{N_{\text{p}}} \left(\dot{m}_{p,ij}^{\text{pred}} - \dot{m}_{p,ij} \right)^2 . \tag{14}$$

Here, N_{s} is the number of samples in the batch. The physics loss term reflects Kirchhoff's second law (Eq. (9)) and it is defined as a log-squared penalty on the cycle-level residuals:

$$\mathcal{L}_{\text{phys}} = \frac{1}{N_{\text{s}} N_{\text{c}}} \sum_{i=1}^{N_{\text{s}}} \sum_{j=1}^{N_{\text{c}}} \left(\ln\left(1 + \frac{|r_{ij}|}{k}\right)\right)^2 . \tag{15}$$

Here, r_{ij} denotes the pressure residual of internal cycle j in sample i, computed from the predicted pipe flows. Specifically,

$$r_{ij} = \left[\boldsymbol{B}_{\text{p}}\, \boldsymbol{\phi}(\dot{\boldsymbol{m}}_{\text{p},i}^{\text{pred}}, \boldsymbol{d}'_{\text{p}}, \boldsymbol{\mathscr{e}}'_{\text{p}}) \right]_j . \tag{16}$$

The normalization constant $k = \delta/(e-1)$ is chosen such that residuals smaller than the threshold $\delta = 100$ Pa (matching the tolerance used in physics solvers such as [39]) maps to below 1 before squaring. The goal is to introduce an inflection point in the loss function: residuals below δ contribute less aggressively, while larger violations remain strongly penalized. As pressure residuals can span several orders of magnitude, we use the natural logarithm to ensure training stability by dampening the effect of large squared errors.

To facilitate convergence, we give more importance to the regression loss in early training, and progressively shift toward enforcing physical consistency. This is achieved through a sigmoid-based schedule centered at the midpoint of training:

$$w_2(n_e) = \sigma\left(20 \cdot \frac{n_e - \frac{N_{\text{e}}}{2}}{N_{\text{e}}}\right), \quad w_1(n_e) = 1 - w_2(n_e)\,, \tag{17}$$

where N_{e} is the total number of epochs and $\sigma(x) = \frac{1}{1+\exp(-x)}$. This schedule ensures a smooth transition from data-driven learning to physics-guided refinement.

4. Thermal simulation

To evaluate the proposed hydraulic model in a practical setting, we couple it with a fully physics-based thermal model that computes the temperature evolution of the network. We retain a physics-based approach for this stage because, unlike the hydraulic problem which involves solving a computationally expensive non-linear system, the thermal problem is linear [12] and can be solved efficiently using direct methods. Consequently, simulation speed depends primarily on

implementation. By optimizing our solver for GPU execution, we match the throughput of surrogate models while preserving strict physical guarantees.

The thermal model relies on the original DHN graph introduced in Section 2, using the predicted mass flows $\dot{\boldsymbol{m}}^{\text{pred}}$ (Section 3) as input. Due to the decoupling of network hydraulics and heat transfer, the thermal simulation is run *after* solving the hydraulics over the entire simulation horizon.

Our implementation builds on the Lagrangian thermal model in [31], which we adapted for GPU execution and improved computational performance. In this model, only pipes are discretized. Each pipe contains a set of water volumes that are tracked in a Lagrangian fashion. The internal pipe walls are also discretized into segments matching the discretization of water volumes. All water volumes and pipe wall segments are initialized at a temperature of 50°C. To prevent unbounded memory growth over the simulation horizon, we impose a limit of 150 discrete volumes per pipe. When this limit is reached, adjacent pairs of fluid volumes are merged, and their temperatures recomputed as a volume-weighted average. While a complete derivation of the model is beyond the scope of this paper, we outline below the key operations involved in each simulation time step. To simplify calculations, edges with negative mass flow are temporarily reversed at the beginning of each time step so that all mass flows are treated as positive in the direction of edge traversal. After all computations, the original edge orientations and flow signs are restored to maintain consistency with the original network topology.

1. **Water and pipe wall temperature update.** The temperature of each fluid volume and wall segment within a pipe is updated using the heat loss formulation from Dénarié et al. [40].
2. **Advection of water volumes.** Fluid volumes in pipes are displaced along the direction of flow according to the mass flow rates. Newly incoming volumes are temporarily assigned a placeholder temperature, to be resolved after solving the nodal energy balance.
3. **Nodal energy conservation.** A linear system is built to enforce energy balance at each non-slack node[2]:

$$\sum_{j\in\mathcal{I}_i} \dot{m}_j \cdot \psi_j(\theta_{t(j)}) = \sum_{k\in\mathcal{O}_i} \dot{m}_k \cdot \theta_i, \quad \forall i \in \mathcal{V} \setminus \mathcal{V}_{\text{slack}}\,, \tag{18}$$

   where $\mathcal{I}_i$ and $\mathcal{O}_i$ are the incoming and outgoing edges at node i, $\theta_{t(j)}$ is the temperature at the starting node (tail) of edge j, $\psi_j(\cdot)$ is the temperature transfer function of edge j, and $\mathcal{V}_{\text{slack}}$ is the set of slack nodes (outlet nodes of heat plants). For slack nodes, the temperature is directly imposed:

$$\theta_i = \theta_{\text{set},i}, \quad \forall i \in \mathcal{V}_{\text{slack}}\,. \tag{19}$$

   Since all transfer functions $\psi_j(\cdot)$ are linear, the system is efficiently solved using direct methods.
4. **Zero-flow nodes.** Nodes that are only connected to edges with zero mass flow cannot be resolved through the energy balance system, as their contribution leads to singularities (i.e., rows of zeros in the system matrix). These nodes are therefore excluded from the system and assigned a temperature equal to the average temperature of the adjacent fluid volumes.
5. **Update of incoming pipe temperatures.** Once all nodal temperatures are computed, the placeholder values assigned to newly incoming water volumes in pipes are overwritten with the temperature of the corresponding inlet node.

To ensure that the initial temperature values do not affect the simulation outcome, a pre-heating phase must be run before recording any results. During this phase, the simulation is carried out for a number of time steps using the first available boundary conditions, allowing thermal transients to stabilize. The required number of pre-heating steps depends on the size and thermal inertia of the network.

Regarding the component-wise transfer functions $\psi_j(\cdot)$ used in the nodal energy balance, we adopt a modified linear formulation of the pipe model originally used in [31]. To account for the contribution of the incoming water temperature when the inflow volume exceeds the pipe capacity, we introduce a blending factor based on the ratio between the incoming volume V_{in} and the pipe's maximum volume V_{max}:

$$\theta_{\text{out}} = \alpha \cdot \theta_{\text{in}} + (1-\alpha)\cdot \bar{\theta}_{\text{out}}, \quad \alpha = \max\left(0, \frac{V_{\text{in}} - V_{\text{max}}}{V_{\text{in}}}\right). \tag{20}$$

Here, $\bar{\theta}_{\text{out}}$ denotes the volume-weighted average temperature of the portion of water volumes displaced outside the pipe. For substations, we retain a fixed temperature drop model, i.e., $\psi_j(\theta_{\text{in}}) = \theta_{\text{in}} - \Delta\theta_{\text{set}}$.

5. Data, model configuration, and evaluation protocol

This section details the experimental framework used to evaluate the proposed surrogate model for internal loop flow prediction in a real-world DHN. We first describe the case study network topology and available monitoring data (Sections 5.1 and 5.2). We then explain the procedure used to generate synthetic training, validation, and testing datasets to have a broad coverage of operational conditions (Section 5.3). Subsequently, we outline the model configurations adopted for benchmarking and describe the training protocol and hyperparameters (Section 5.4). We then define the evaluation metrics: a core set used for model selection, and additional set to assess robustness and constraint satisfaction of the best performing model (Section 5.5). Finally, we provide relevant implementation details (Section 5.6).

5.1. Case study

The case study considered in this work is the DHN of Verbier, Switzerland, described in [31]. The network serves 165 substations through 3 heat plants, one of which is always active (MAIN) while the remaining two (SEC1 and SEC2) are activated when MAIN cannot satisfy the demand. The supply and return lines of the network have the same meshed topology with six overlapping internal loops each, and occasionally aerial pipes. Since no reliable data on the elevation of the network is available, we disregard any hydrostatic pressure differences assuming equal pipe heights. Fig. 7 shows the layout of the network highlighting the central section that contains the internal loops.

5.2. Basic data collection

Monitoring data from the network's sensors was collected between January 10 and January 22 2022, with several missing measurements starting from January 16. This includes (i) incoming and outgoing measures of temperature and volumetric flow for all substations and heat plants, and (ii) inlet and outlet pressure for the heat plants and 2 substations highlighted in Fig. 7. In addition, the peak power of substations and heat plants was collected from the DHN operator. The data is processed and aggregated into 15-minute intervals according to the process described in [31].

5.3. Experimental data generation

Our goal is to estimate internal loop flows under varying boundary conditions, specifically in the central meshed section of the network (Fig. 7). The available monitoring data only covers a short winter period (see Section 5.2) and does not capture the full range of operating conditions experienced throughout the year. Moreover, measurements are only available at substations and heat plants, while mass flows and

[2] Slack nodes have fixed temperatures that do not depend on the network and act as boundary conditions in the thermal simulation.

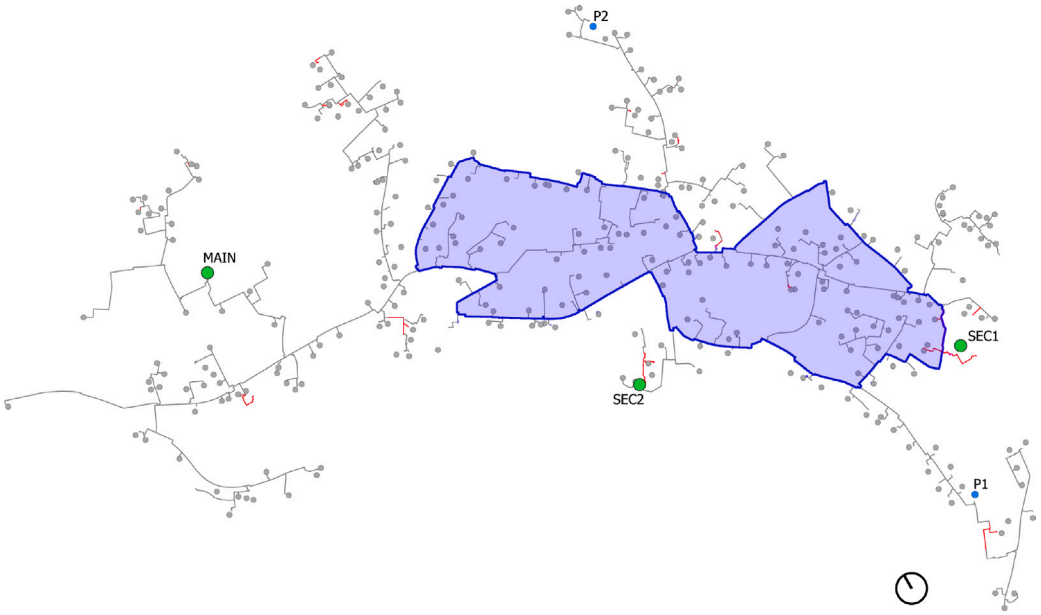


Fig. 7. The DHN of Verbier. Heat plants are shown in green, while the substations with pressure measurements are in blue and aerial pipes are colored in red. The central section of the network containing the internal loops is highlighted in blue.

pressures in the network pipes are not directly measured. For these reasons, building on real operational data to define realistic constraints, we generate training, validation, and testing datasets through systematic sampling of boundary conditions, and physics-based simulations.

Since the supply and return lines share identical topology and properties, corresponding pipes in the two lines will have the same mass flow under a given set of boundary conditions. Therefore, it is sufficient to explicitly generate boundary conditions that ensure broad coverage on the supply side, implicitly covering the same problem space on the return. We begin by calculating the minimum and maximum mass flow rates for each of the 37 branches connected to the internal loops, excluding the branch containing the MAIN heat plant. The maximum mass flow rate is estimated based on the branch's cumulative peak power (as provided by the DHN operator), assuming a fixed temperature difference of 30 K at substations, which is a typical design value in DHNs. For a minority of substations where a higher mass flow than the one estimated with the above method was measured, we set the peak mass flow rate as the maximum measured value plus a 5% margin. The minimum mass flow rate for each branch is then set as 0, minus potentially the maximum mass flow output of a secondary heat plant if present in the branch, computed from the peak power and a temperature difference of 30 K. We then use Latin Hypercube Sampling (LHS) to draw synthetic scenarios between these bounds, providing a uniform coverage of the high-dimensional input space that spans the full range of plausible operating conditions. Mass flow in the remaining branch containing the MAIN heat plant is adjusted to ensure mass balance for each scenario.

Finally, we evenly distribute each branch mass flow among its non-pipe elements as boundary conditions and simulate each scenario using PyDHN [31], with a convergence threshold of 1 Pa. This process is repeated to create separate sets of values for training (150 000 cases), validation (50 000 cases), and testing (50 000 cases). Each case is split into two condensed graphs, as described in Section 3.2. The subsets of chosen internal cycles for the supply and return line are shown in Fig. 8. With this specific cycle configuration, the farthest cycles are separated by a distance of 3 (i.e., three hops are required to traverse from one cycle to another; for example, cycles 3 and 4). Considering the model architecture described in Section 3.4, which alternates message passing between pipe and cycle vertices across layers, at least 5 layers are necessary to ensure information exchange between all cycles.

Table 1
Training dataset input feature statistics for pipe nodes. The normalized range is computed by dividing each value by the absolute maximum of the corresponding feature within the training dataset. Static features (d', ℓ') represent fixed pipe properties, while dynamic features ($\dot{m}'_b$) vary across operating scenarios.

Feature	Range	Mean ± Std	Norm.	Static
d' (m)	[0.0320, 0.2000][a]	0.0825 ± 0.0312	[0.16, 1.00]	✓
ℓ' (m)	[2.500, 185.0]	45.32 ± 38.71	[0.01, 1.00]	✓
$\dot{m}'_b$ (kg s^{-1})	[−12.50, 48.30]	2.156 ± 4.823	[−0.26, 1.00]	

[a] Discrete values based on commercial diameters.

Table 1 summarizes the distribution of input features generated for training.

5.4. Model configurations

The proposed reference (*ELU-2-7l*) architecture, described in Section 3.4, is a two-stage GNN composed of two stacked sub-models with 7 message-passing layers each. To better understand the contribution of different architectural choices, we compare the proposed model with a set of variants, denoted as *[activation]-[#stages]-[#layers]*. We vary:

- **Activation function:** To isolate the benefit of our proposed ELU activation, we test variants that use the ReLU and BatchNorm combination, in line with the original DyResGEN formulation.
- **Number of stages:** To study the effect of staging on oversmoothing, we compare our two-stage architecture against single-stage variants.
- **Model depth (single-stage only):** To evaluate the impact of depth, we test single-stage models with 7 and 15 layers. The 15-layer model has a comparable number of parameters and message-passing operations to our two-stage model, providing a direct comparison.

All models use a hidden dimensionality of 256 for node embeddings and are trained using the AdamW optimizer [41] for 100 epochs. The learning rate is initialized at 0.0001 and decayed by a factor of 10 every 10 epochs. We use a batch size of 32 and a weight decay factor of 0.01. To account for variability due to initialization, each configuration is run 10 times with different random seeds.

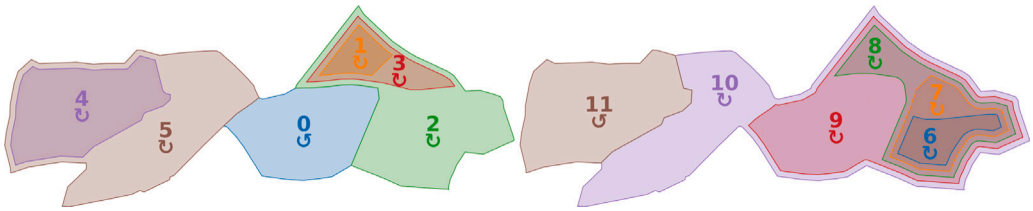


Fig. 8. Selected internal cycles for the supply (left) and return (right) lines with their respective IDs and directions. The section of the network with these cycles is highlighted in Fig. 7.

Table 2
Validation set performance over 10 runs for each tested architecture. Values reported as mean ± std (best) over the different seeds. Best values for each KPI are highlighted in bold.

Model	MAE_m ($kg\,s^{-1}$) ↓	Res_c (Pa) ↓	Acc (%) ↑
ELU-2-7l	**0.0040** ± 0.0006 (**0.0037**)	**23.72** ± 3.99 (**17.95**)	86.17 ± 9.92 (**96.94**)
ELU-1-7l	0.0165 ± 0.0004 (0.0162)	86.52 ± 6.39 (79.13)	12.67 ± 2.61 (16.46)
ELU-1-15l	0.0126 ± 0.0009 (0.0120)	74.66 ± 10.31 (65.38)	20.50 ± 5.26 (27.07)
ReLU-2-7l	0.0060 ± 0.0004 (0.0056)	26.41 ± 1.94 (24.09)	**88.90** ± 2.06 (91.46)
ReLU-1-7l	0.0232 ± 0.0020 (0.0214)	190.92 ± 17.22 (166.08)	1.55 ± 0.73 (3.11)
ReLU-1-15l	0.0272 ± 0.0043 (0.0196)	232.66 ± 54.43 (127.51)	1.48 ± 1.38 (5.29)

5.5. Evaluation metrics

We evaluate model performance using two groups of metrics: one for model selection, evaluated on the validation set, and a second set for further analysis of the best-performing model, evaluated on the test set. The following metrics are used for model selection:

1. **Accuracy (Acc)**: the percentage of test samples for which all cycle residuals in Eq. (9) are below 100 Pa.
2. **Mean Absolute Error on pipe-level mass flow (MAE_m)**: the average prediction error in mass flow across all pipes and samples.
3. **Average cycle residual (Res_c)**: the average absolute residual (Pa) per cycle, averaged over all cycles and samples.

For each model configuration, we retain the parameters corresponding to the epoch with the highest *Acc*, and report the average and best performance over the 10 runs. After identifying the best-performing architecture based on *Acc*, we further characterize its robustness using:

4. **Sample-level maximum residual (Res_s)**: the average of the sample-wise maximum absolute cycle residuals, taken over the entire test set.
5. **Global maximum residual (Res_m)**: the single highest absolute cycle residual observed across all test samples.

5.6. Implementation details

All experiments were conducted on a system equipped with an NVIDIA GeForce RTX 3090 and 16 GB of RAM. The models were implemented in PyTorch 2.1.2 [42] and PyTorch Geometric 2.5.0 [43].

6. Results

We compare variants of the proposed GNN architecture along two dimensions: (i) single-stage (7 and 15 layers) vs. two-stage design, and (ii) ELU-based vs. ReLU-based blocks. Table 2 summarizes the results for the six configurations described in Section 5.4 according to the metrics detailed in Section 5.5.

The *ELU-2-7l* variant outperformed, on average, all other tested architectures, with the best model estimating the hydraulic state of the network with a maximum residual of 100 Pa on 97% of the validation cases. However, compared to its *ReLU* counterpart, accuracy exhibited a wider variation, ranging from 67.11% to 96.94%, whereas with *ReLU-2-7l* it ranged from 83.83% to 91.46%. Despite this, the estimation of mass flow was closer to that of physics-based simulations and the average residuals were lower. Interestingly, architectures that employ a two-stage refinement process demonstrated significantly higher accuracy compared to single-stage architectures, even when using the same total number of message-passing layers. In fact, simply increasing the number of layers in the single-stage model only improved performance by a modest margin, and even degraded average performance in the *ReLU* variant. This is consistent with our earlier analysis (see Section 3.4) showing that at least five layers are needed for communication between the most distant cycles. Adding more layers without architectural separation may lead to oversmoothing rather than increased expressiveness.

Table 3
Performance of the best run (model *ELU-2-7l*) on the test set: results are shown for the supply and return lines separately.

Subset	Acc ↑ (%)	MAE_m ↓ ($kg\,s^{-1}$)	Res_c ↓ (Pa)	Res_s ↓ (Pa)	Res_m ↓ (Pa)
Supply	97.34	0.0034 ± 0.0017	16.82 ± 18.31	42.00 ± 23.75	686.26
Return	96.42	0.0040 ± 0.0024	19.03 ± 19.26	45.43 ± 25.17	1099.30

An overview of the performance of the best training run on the test set is given in Table 3. The model achieves comparable performance on the test set as it did on the validation set, with an overall accuracy of 96.89%. This accuracy value is averaged over the supply and return lines, which, although physically identical, are represented using different cycle bases. As a result, the corresponding condensed graphs differ in structure, providing distinct inputs to the GNN despite encoding the same underlying physics (see Section 5.3). In practical applications, however, where the supply and return networks are symmetric, the model could be trained and evaluated on just one side, with flows for the other inferred directly. At test time, this also means that the model's effective accuracy could be approximated by selecting the best-performing cycle decomposition, rather than averaging across both. We observed slightly different model performance across the two lines: while in both cases the accuracy was above 96%, the model exhibits slightly better values for all metrics on the supply line. While the consistent high accuracy across these differing incidence structures serves as a promising indicator of the model's robustness to the cycle basis selection (provided that the maximum distance between cycle nodes is within the model's receptive field), further investigation on physically distinct networks is required to generalize this finding.

A closer look at the model accuracy under different acceptance thresholds on physics residuals is given in Fig. 9. We observe that

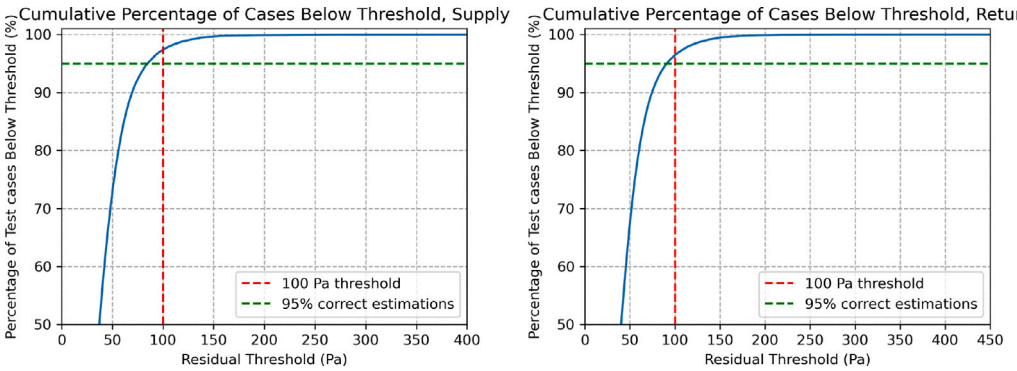


Fig. 9. Cumulative percentage of test samples whose maximum cycle residual falls below a given threshold, shown separately for the supply (left) and return (right) lines. Dashed lines mark the 100 Pa convergence threshold and the 95% accuracy level.

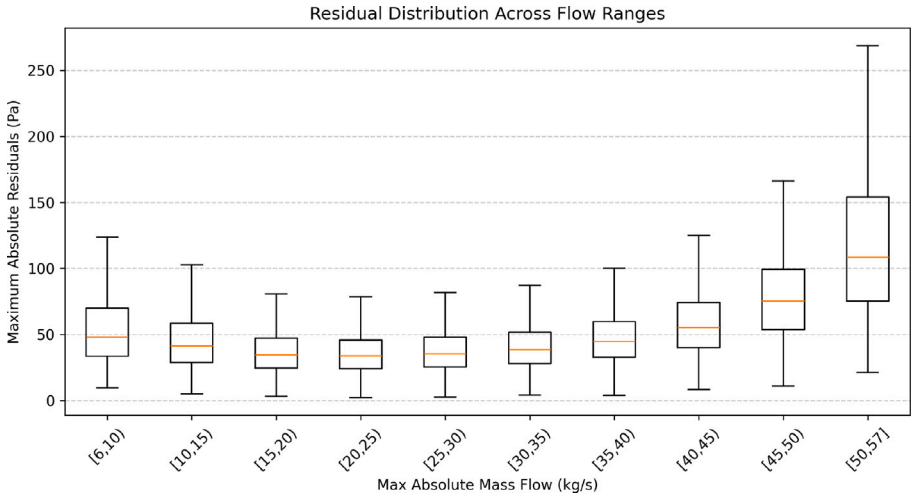


Fig. 10. Box plot of sample-level maximum absolute residuals on the test set, grouped by the maximum circulating mass flow ($\mathrm{kg\,s^{-1}}$) in each sample.

most cases have a maximum residual below 50 Pa, including more than 70% of supply line samples. In general, cycles with ID 4 in the supply line and ID 7 in the return line most frequently exhibited the highest pressure residuals among all cycles, though no clear pattern or explanation was identified.

To better understand when the surrogate model tends to perform worse, we analyze how the magnitude of residuals varies across different flow conditions. This helps us identify whether performance degradation correlates with extreme operating regimes, such as high or low circulating mass flow. Examining the mass flow distribution in the network (Fig. 10), we find that the residuals are, on average, higher when the mass flow is close to its lower or upper limit, and particularly for values above 40 $\mathrm{kg\,s^{-1}}$. This is likely because, although boundary conditions for each branch were sampled uniformly (LHS), extreme total flows require the rare configuration of many branches operating simultaneously at their limits. Additionally, since pressure losses scale roughly with the square of the mass flow, modulated by a flow-dependent friction factor, even small relative errors in predicted mass flows can lead to large absolute residuals under high-flow conditions.

As a further investigation, we isolated the test samples where the model failed to meet the convergence threshold. We then solved these cases using the Newton–Raphson method, comparing a standard cold initialization (zero flow) against a warm initialization using the surrogate's non-converged prediction. With the first approach the solver required 7.62 ± 1.04 iterations to converge, whereas the warm start reduced this to 1.27 ± 0.44 iterations.

6.1. Training cost analysis

Generating the 250 000 training, validation, and test samples via physics-based simulation with PyDHN required approximately 2 h 50 min on a single CPU core. Training the GNN surrogate for 100 epochs on the full training set (150 000 samples), including validation monitoring and gradient computation, required approximately 9 h 55 min on a single GPU. Once trained, the model can be deployed for inference with negligible additional computational cost.

For applications involving large numbers of operating scenarios, this one-time training cost is amortized across subsequent inference calls. In particular, the proposed surrogate achieves an 85× speedup over the physics-based solver. For example, simulating 1×10^6 hydraulic steps would require approximately 11 h with PyDHN, compared to about 8 min using the trained GNN.

7. Ablation study: Loss function variants

To assess the impact of the proposed loss design, we conduct an ablation study evaluating two simplified alternatives:

1. A model trained using only the regression loss $\mathcal{L}_{\mathrm{reg}}$, without any physical residual supervision.
2. A model trained with fixed, equal weights ($w_1, w_2 = 0.5$) for the regression and physics loss components throughout the full training.

Both variants use the same architecture as the reference model (*ELU-2-7l*), trained under identical conditions over 10 random seeds.

Table 4
Ablation study on the loss function. Comparison between regression-only loss (*Regr.*), fixed-weight loss (*Fixed*), and adaptive weighting (*Ours*). Validation set performance over 10 runs reported as mean ± std (best). Best values in bold.

Variant	MAE_m ($kg\,s^{-1}$) ↓	Res_c (Pa) ↓	Acc (%) ↑
Regr.	**0.0029** ± 0.0001 (**0.0027**)	23.09 ± 1.11 (19.90)	83.08 ± 2.97 (90.24)
Fixed	0.0079 ± 0.0004 (0.0072)	26.66 ± 0.97 (23.78)	**90.75** ± 0.78 (92.83)
Ours	0.0038 ± 0.0001 (0.0037)	**22.79** ± 1.30 (**17.95**)	89.23 ± 3.05 (**96.94**)

We omit a physics-only variant, since it failed to produce meaningful results. Table 4 reports the average and best performance across seeds on the validation set. To reduce the impact of outliers, the averages are computed after discarding the two best and worst runs for each variant.

The regression-only model achieves the lowest average and best MAE_m, but has a noticeably lower physical accuracy. While the regression loss minimizes absolute flow differences uniformly across the network, this does not translate directly into improved physical consistency. Pressure losses depend nonlinearly on pipe diameter, and linearly on length, meaning that identical flow errors can result in significantly different residuals across cycles. The physics-based loss accounts for these effects, making it a more targeted objective for improving physical fidelity, even if the overall MAE_m is slightly higher. At the same time, gradually shifting training focus from data consistency to physical consistency appears beneficial for overall performance, allowing the model to first learn the general structure of the outputs before enforcing domain constraints more strictly. In fact, while the average accuracy is slightly lower (by approximately 1.5%) when using the weight scheduler, the average cycle residuals are consistently reduced, indicating an overall improvement on physical consistency.

8. Example application

To demonstrate the practical applicability of our surrogate model, this section replicates the experiment from [31], comparing our model's predictions against both real operational data from the Verbier DHN and the results of the original work.

The hydraulic simulation is run with a 15-minute time step, to match the resolution of the operational data, assuming constant flows within each interval. The thermal simulation uses a finer 60-second step size to capture faster heat dynamics. As the supply and return lines share layout and characteristics, we use the surrogate model to estimate mass flows only for the supply line and apply the same output to the return line. The full 12-day period is simulated, but results are only evaluated over the first 6 days with reliable data, totaling 1247 hydraulic steps, 18 705 thermal steps, and 1440 pre-heating steps (24 h) for the thermal case. Using the proposed method, the hydraulic simulation completes in 0.57 s (2187 steps/s) versus 50 s with PyDHN (solver time only), with a 97.75% convergence rate. The full thermal simulation, including pre-heating, completes in 3 min and 47 s (89 steps/s).

Table 5 summarizes model performance against monitoring data, focusing on inlet temperatures of MAIN, SEC1, P1, and P2. Following the recommendations of Johra et al. [44], we report multiple metrics to capture both accuracy and variability, even though reference measurements are available. In addition to the Mean Absolute Error (MAE), we compute the Normalized Mean Bias Error (NMBE) to quantify systematic bias, and three forms of the Coefficient of Variation of the Root Mean Square Error (CVRMSE): standard, hourly-averaged (HCVRMSE), and daily amplitude (DACVRMSE).

The two approaches show similar performance, with error metrics falling within the expected range of modeling and measurement uncertainty. Both models generally have small NMBE, with the proposed approach showing slightly higher values than PyDHN. This may suggest a minor systematic bias, possibly due to overfitting or limitations in generalizing across different conditions.

Table 5
Comparison of the proposed surrogate-based pipeline (ML) and the physics-based reference (PyDHN) against monitoring data for inlet temperatures at selected measurement points. NMBE, CVRMSE, HCVRMSE, DACVRMSE, and MAE are reported.

	Method	NMBE	RMSE			MAE
		(≈0)	CV (↓)	HCV (↓)	DACV (↓)	(↓)
MAIN	PyDHN	0.20	1.41	1.09	34.49	0.45
	ML	0.36	1.38	1.07	35.71	0.43
SEC1	PyDHN	0.94	2.14	1.59	11.46	0.71
	ML	0.94	2.21	1.61	13.69	0.74
P1	PyDHN	1.79	2.55	2.17	86.91	1.59
	ML	1.85	2.59	2.20	88.85	1.63
P2	PyDHN	0.08	1.88	1.41	35.98	0.82
	ML	0.17	1.91	1.45	36.2	0.83

9. Discussion

Results show that the proposed surrogate achieves over 97% accuracy on synthetic test data with a throughput of 2187 hydraulic steps per second, approximately 85 times faster than iterative solvers in our setup. This represents a significant improvement to the speed-accuracy tradeoff typical of physics-based tools, confirming the suitability of the framework for the applications outlined in Section 2.2: long-horizon simulations, design optimization, and scenario analysis where hydraulic and thermal dynamics are decoupled or where multiple scenarios can be evaluated in parallel. To this end, an additional benefit for optimization is the model's differentiability: the GNN provides smooth gradients that are beneficial for direct integration into gradient-based pipelines. Non-converged predictions remain within reasonable physical bounds and, where strict accuracy is required, provide effective warm starts that reduce Newton–Raphson iterations from 7.6 to 1.3 on average.

In its current form, the model is suited for applications involving a single DHN topology, addressing the scalability and meshed network limitations identified in Section 1.1. The computational burden shifts from inference to training; however, this one-time investment (less than 13 h including data generation) is quickly absorbed after 10^5–10^6 forward evaluations, a threshold easily reached in optimization or uncertainty quantification workflows. Fine-tuning from pre-trained weights could substantially reduce this initial investment, although this remains to be validated experimentally. Being based on message passing, the proposed architecture also has the potential to be used to train models that generalize across networks, but this too requires further validation.

A practical consideration for deployment is that the surrogate learns the hydraulic behavior encoded in the training data, which relies on nominal pipe properties. Real networks may deviate from design specifications due to aging, fouling, or undocumented modifications, potentially requiring calibration of the underlying physical model before generating training samples.

The choice of coupling the learned hydraulic model with a physics-based thermal simulation achieves throughput comparable to fully surrogate approaches in the literature (e.g., [27]), while retaining strict physical guarantees on thermal dynamics. The comparable accuracy between the hybrid pipeline and the fully physics-based tool PyDHN on real monitoring data (Table 5) confirms that the learned hydraulic

approximation does not propagate into significant thermal prediction errors.

Finally, it is worth noting that the model is only suitable for third-generation DHNs. This is a reasonable limitation given the widespread adoption of these systems and the utility of the model towards operational optimization, rather than design of new networks. Extending the approach to later-generation DHNs with bidirectional flows or compressible working fluids would require methodological adaptations beyond the scope of this work.

10. Conclusion

This work presented a graph neural network-based surrogate model for the hydraulic simulation of meshed district heating networks. The key contribution is addressing the receptive field limitation that prevents state-of-the-art surrogates based on graph neural network from accurately solving the hydraulic problem in meshed networks, where pressure and flow distributions depend on global balance across internal loops. By reformulating the problem in terms of cycle flows and introducing a condensed graph representation, the proposed architecture captures global network dynamics within a shallow message-passing framework, while retaining the scalability and architectural transferability of existing approaches. We couple this surrogate with a fully physics-based Lagrangian thermal model optimized for parallel computation, preserving physical guarantees on thermal dynamics while reaching the throughput of data-driven models.

The surrogate model matches the performance of physics-based solvers in over 97% of synthetic test cases while providing a computational speedup of approximately 85 times. When coupled with the physics-based thermal model and validated against real-world monitoring data from an operational district heating network, predictions are comparable to high-fidelity simulation tools, with negligible differences in error metrics.

By shifting computational effort from repeated iterative solves to a one-time training investment, the framework is well-suited for practical applications requiring extensive forward evaluations, such as real-time control optimization, scenario analysis, and long-horizon simulations.

Future work will focus on investigating the transferability of the model to different case studies, potentially with few labeled examples, and on improving the model's generalization to unseen network layouts and operating conditions.

CRediT authorship contribution statement

Roberto Boghetti: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Jean-Marc Odobez:** Writing – review & editing, Supervision, Conceptualization. **Jérôme H. Kämpf:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used Google Gemini 2.5 Pro in order to improve the clarity and readability of sentences. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

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

R.B. acknowledges additional support from the EPFLGlobaLeaders program, which has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 945363. The authors would like to thank ALTIS Groupe SA for providing the monitoring data of the Verbier network. The weather data was provided by MeteoSwiss, the Swiss Federal Office of Meteorology and Climatology.

Appendix. Analysis of the two-stage architecture

This appendix investigates the mechanism behind the performance gains of the two-stage architecture. We analyze the pressure residuals at three points: the initial boundary flows ($\dot{m}_b$), the intermediate output (Stage 1), and the final prediction (Stage 2). We conduct the analysis on the test dataset using the best model realization of the *ELU-2-7l* variant and considering only the supply line results.

Table A.6 presents the residual statistics of the model at the different points. The two stages contribute asymmetrically to the total error reduction. In terms of absolute improvement in Res_s:

- **Initial → Stage 1:** Mean reduction of 3.58×10^6 Pa (99.7% of total improvement)
- **Stage 1 → Stage 2:** Mean reduction of 10 589 Pa (0.3% of total improvement)

Despite contributing only 0.3% of the absolute improvement, Stage 2 is essential to raise the accuracy above 0. These results suggest that the two stages might solve fundamentally different sub-problems.

To investigate this hypothesis, we examine how residual magnitude relates to the underlying cycle flow. Fig. A.11 shows that Stage 1 residuals exhibit a power-law relationship with ground-truth cycle flow magnitude ($|\text{residual}| \propto |\tilde{m}|^{\alpha}$, with $\alpha \approx 0.71$), while Stage 2 substantially reduces this dependency ($\alpha \approx 0.29$). The distinct scaling behaviors are consistent with the two stages implicitly specializing into complementary computational roles, analogous to a *coarse-to-fine* strategy in numerical solvers.

These findings offer potential insights on the superior performance of the two-stage architecture compared to a single-stage variant of equivalent depth (as reported in Section 6). While establishing a causal link is a challenging endeavor beyond the scope of this study, we suggest one possible explanation. The performance gap may stem from the difficulty deep GNNs face in preserving high-frequency local information against *oversmoothing*: as message-passing layers accumulate, representations of local physical features (e.g., pipe diameters) become diluted by global topological information, precisely when such features become most critical for a finer resolution of cycle interactions. The two-stage architecture circumvents this by design. Discarding the intermediate latent representation and re-injecting original features at Stage 2 provides undiluted local information after a coarse solution has been provided.

Table A.6
Pressure residual statistics across inference stages. The tolerance threshold for operational accuracy is 100 Pa.

Metric	Initial	Stage 1	Stage 2
Res_c (Pa)	$8.95 \times 10^5 \pm 1.65 \times 10^6$	3139 ± 4157	16.8 ± 18.3
Res_s (Pa)	$3.59 \times 10^6 \pm 2.51 \times 10^6$	$10\,631 \pm 3119$	42.00 ± 23.7
Res_m (Pa)	1.09×10^7	23 660	686
Acc (%)	0.0	0.0	97.3

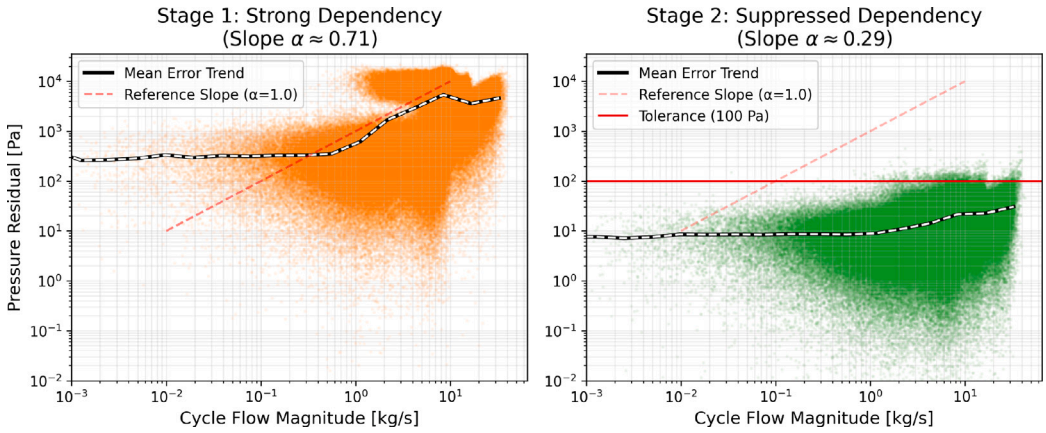


Fig. A.11. Pressure residual magnitude versus ground-truth cycle flow for Stage 1 (left) and Stage 2 (right). The black line shows the binned mean trend. Stage 1 exhibits strong flow-dependent error ($\alpha \approx 0.71$), while Stage 2 suppresses this dependency ($\alpha \approx 0.29$), achieving residuals predominantly below the 100 Pa tolerance (red line) across all flow magnitudes.

Data availability

The authors do not have permission to share the data. The models used in this study are available from the corresponding author RB on request.

References

[1] IEA. World Energy Outlook 2023. Technical Report, IEA; 2023.

[2] Zuberi MJS, Narula K, Klinke S, Chambers J, Streicher KN, Patel MK. Potential and costs of decentralized heat pumps and thermal networks in Swiss residential areas. Int J Energy Res 2021;45(10):15245–64. http://dx.doi.org/10.1002/er.6801.

[3] Sporleder M, Rath M, Ragwitz M. Design optimization of district heating systems: A review. Front Energy Res 2022;10. http://dx.doi.org/10.3389/fenrg.2022.971912.

[4] Buffa S, Fouladfar MH, Franchini G, Lozano Gabarre I, Andrés Chicote M. Advanced control and fault detection strategies for district heating and cooling systems—A review. Appl Sci 2021;11(1):455. http://dx.doi.org/10.3390/app11010455.

[5] Tosatto A, Dahash A, Ochs F. Simulation-based performance evaluation of large-scale thermal energy storage coupled with heat pump in district heating systems. J Energy Storage 2023;61:106721. http://dx.doi.org/10.1016/j.est.2023.106721.

[6] Guelpa E, Verda V. Demand response and other demand side management techniques for district heating: A review. Energy 2021;219:119440. http://dx.doi.org/10.1016/j.energy.2020.119440.

[7] Hari SKK, Zlotnik A, Srinivasan S, Sundar K, Ewers M. Optimization of district heating network parameters in steady-state operation. 2024, http://dx.doi.org/10.48550/arXiv.2404.18868, arXiv:2404.18868.

[8] Morvaj B, Evins R, Carmeliet J. Optimising urban energy systems: Simultaneous system sizing, operation and district heating network layout. Energy 2016;116:619–36. http://dx.doi.org/10.1016/j.energy.2016.09.139.

[9] Del Hoyo Arce I, Herrero López S, López Perez S, Rämä M, Klobut K, Febres JA. Models for fast modelling of district heating and cooling networks. Renew Sustain Energy Rev 2018;82:1863–73. http://dx.doi.org/10.1016/j.rser.2017.06.109, [Accessed 2024-10-21].

[10] Guelpa E. Impact of network modelling in the analysis of district heating systems. Energy 2020;213:118393. http://dx.doi.org/10.1016/j.energy.2020.118393.

[11] Kuntuarova S, Lickleder T, Huynh T, Zinsmeister D, Hamacher T, Perić V. Design and simulation of district heating networks: A review of modeling approaches and tools. Energy 2024;305:132189. http://dx.doi.org/10.1016/j.energy.2024.132189.

[12] Guelpa E, Verda V. Compact physical model for simulation of thermal networks. Energy 2019;175:998–1008. http://dx.doi.org/10.1016/j.energy.2019.03.064.

[13] Zhang Z, Wang P, Jiang P, Liu Z, Fu L. Energy management of ultra-short-term optimal scheduling of integrated energy system considering the characteristics of heating network. Energy 2022;240:122790. http://dx.doi.org/10.1016/j.energy.2021.122790.

[14] Rodrigue D, Mabrouk MT, Pasdeloup B, Meyer P, Lacarrière B. Topology reduction through machine learning to accelerate dynamic simulation of district heating. Energy AI 2024;17:100393. http://dx.doi.org/10.1016/j.egyai.2024.100393.

[15] Trias A. The holomorphic embedding load flow method. In: 2012 IEEE power and energy society general meeting. 2012, p. 1–8. http://dx.doi.org/10.1109/PESGM.2012.6344759.

[16] Guo J, Li X, Jiang T, Zhang R, Wang C. Energy flow analysis of electricity and heat integrated energy systems based on holomorphic embedding method. In: 2024 IEEE power & energy society general meeting. PESGM, 2024, p. 1–5. http://dx.doi.org/10.1109/PESGM51994.2024.10688482.

[17] Chen K, Sun K, Lin L, Chen S, Wang M, Li Z, Zheng J. Steady-state flow analysis of district heating system using the holomorphic embedding. In: 2022 power system and green energy conference. PSGEC, 2022, p. 699–704. http://dx.doi.org/10.1109/PSGEC54663.2022.9881196.

[18] Sun Y, Ding T, Xue Y, Jia W, Chang X, Shahidehpour M. Multi-dimensional holomorphic embedding method for probabilistic energy flow calculation in integrated distribution power and heating system. IEEE Trans Sustain Energy 2024;15(3):1677–89. http://dx.doi.org/10.1109/TSTE.2024.3366311.

[19] Villano F, Mauro GM, Pedace A. A review on machine/deep learning techniques applied to building energy simulation, optimization and management. Thermo 2024;4(1):100–39. http://dx.doi.org/10.3390/thermo4010008.

[20] Potočnik P, Škerl P, Govekar E. Machine-learning-based multi-step heat demand forecasting in a district heating system. Energy Build 2021;233:110673. http://dx.doi.org/10.1016/j.enbuild.2020.110673.

[21] Li M, Deng W, Xiahou K, Ji T, Wu Q. A data-driven method for fault detection and isolation of the integrated energy-based district heating system. IEEE Access 2020;8:23787–801. http://dx.doi.org/10.1109/ACCESS.2020.2970273.

[22] Pinto G, Deltetto D, Capozzoli A. Data-driven district energy management with surrogate models and deep reinforcement learning. Appl Energy 2021;304:117642. http://dx.doi.org/10.1016/j.apenergy.2021.117642.

[23] Rodrigue D, Mabrouk MT, Pasdeloup B, Meyer P, Lacarrière B. Leveraging artificial neural networks to accelerate dynamic simulations of district heating networks through topology reduction. 2024, http://dx.doi.org/10.2139/ssrn.4799722, arXiv:4799722.

[24] Boussaid T, Rousset F, Scuturici V-M, Clausse M. Evaluation of graph neural networks as surrogate model for district heating networks simulation. In: 36th international conference on efficiency, cost, optimization, simulation and environmental impact of energy systems. ECOS 2023, Las Palmas De Gran Canaria, Spain: ECOS 2023; 2023, p. 3182–93. http://dx.doi.org/10.52202/069564-0286.

[25] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Trans Neural Netw 2009;20(1):61–80. http://dx.doi.org/10.1109/TNN.2008.2005605.

[26] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. 2018, http://dx.doi.org/10.48550/arXiv.1710.10903, arXiv:1710.10903.

[27] Boussaid T, Rousset F, Scuturici V-M, Clausse M. Enabling fast prediction of district heating networks transients via a physics-guided graph neural network. Appl Energy 2024;370:123634. http://dx.doi.org/10.1016/j.apenergy.2024.123634.

[28] Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. 2014, http://dx.doi.org/10.48550/arXiv.1406.1078, arXiv:1406.1078.

[29] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. 2017, http://dx.doi.org/10.48550/arXiv.1609.02907, arXiv:1609.02907.

[30] de Giuli LB, La Bella A, Scattolini R. Physics-informed neural network modeling and predictive control of district heating systems. IEEE Trans Control Syst Technol 2024;32(4):1182–95. http://dx.doi.org/10.1109/TCST.2024.3355476.

[31] Boghetti R, Kämpf JH. Verification of an open-source Python library for the simulation of district heating networks with complex topologies. Energy 2024;290:130169. http://dx.doi.org/10.1016/j.energy.2023.130169.

[32] Osiadacz AJ, Pienkosz K. Methods of steady-state simulation for gas networks. Int J Syst Sci 1988;19(7):1311–21. http://dx.doi.org/10.1080/00207728808547163.

[33] Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural message passing for quantum chemistry. In: Proceedings of the 34th international conference on machine learning. PMLR; 2017, p. 1263–72.

[34] Rusch TK, Bronstein MM, Mishra S. A survey on oversmoothing in graph neural networks. 2023, http://dx.doi.org/10.48550/arXiv.2303.10993, arXiv:2303.10993.

[35] Li G, Xiong C, Thabet A, Ghanem B. DeeperGCN: All you need to train deeper GCNs. 2020, http://dx.doi.org/10.48550/arXiv.2006.07739, arXiv:2006.07739.

[36] Fukushima K. Visual feature extraction by a multilayered network of analog threshold elements. IEEE Trans Syst Sci Cybern 1969;5(4):322–33. http://dx.doi.org/10.1109/TSSC.1969.300225.

[37] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015, http://dx.doi.org/10.48550/arXiv.1502.03167, arXiv:1502.03167.

[38] Clevert D-A, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUs). 2016, http://dx.doi.org/10.48550/arXiv.1511.07289, arXiv:1511.07289.

[39] Robinson D, Haldi F, Kämpf JH, Leroux P, Perez D, Rasheed A, Wilke U. CitySim: Comprehensive micro-simulation of resource flows for sustainable urban planning. In: Proceedings of building simulation 2009: 11th conference of IBPSA. Building simulation, vol. 11, Glasgow, Scotland: IBPSA; 2009, p. 1083–90. http://dx.doi.org/10.26868/25222708.2009.1083-1090.

[40] Dénarié A, Aprile M, Motta M. Heat transmission over long pipes: New model for fast and accurate district heating simulations. Energy 2019;166:267–76. http://dx.doi.org/10.1016/j.energy.2018.09.186.

[41] Loshchilov I, Hutter F. Decoupled weight decay regularization. 2019, http://dx.doi.org/10.48550/arXiv.1711.05101, arXiv:1711.05101.

[42] Ansel J, Yang E, He H, Gimelshein N, Jain A, Voznesensky M, Bao B, Bell P, Berard D, Burovski E, Chauhan G, Chourdia A, Constable W, Desmaison A, DeVito Z, Ellison E, Feng W, Gong J, Gschwind M, Hirsh B, Huang S, Kalambarkar K, Kirsch L, Lazos M, Lezcano M, Liang Y, Liang J, Lu Y, Luk CK, Maher B, Pan Y, Puhrsch C, Reso M, Saroufim M, Siraichi MY, Suk H, Zhang S, Suo M, Tillet P, Zhao X, Wang E, Zhou K, Zou R, Wang X, Mathews A, Wen W, Chanan G, Wu P, Chintala S. PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, volume 2. ASPLOS '24, vol. 2, New York, NY, USA: Association for Computing Machinery; 2024, p. 929–47. http://dx.doi.org/10.1145/3620665.3640366.

[43] Fey M, Lenssen JE. Fast graph representation learning with pytorch geometric. 2019, http://dx.doi.org/10.48550/arXiv.1903.02428, arXiv:1903.02428.

[44] Johra H, Mans M, Filonenko K, De Jaeger I, Saelens D, Tvedebrink T. Evaluating different metrics for inter-model comparison of urban-scale building energy simulation time series. In: 2021 Building Simulation Conference. 2021, http://dx.doi.org/10.26868/25222708.2021.30410.