State estimation in water distribution system via diffusion on the edge space

Kerimov, Bulat; Yang, Maosheng; Taormina, Riccardo; Tscheikner-Gratl, Franz

DOI
10.1016/j.watres.2024.122980

Publication date
2025

Document Version
Final published version

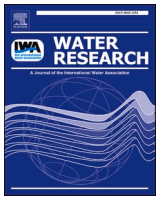
Published in
Water Research

Water Research 274 (2025) 122980

Contents lists available at ScienceDirect

Water Research

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



State estimation in water distribution system via diffusion on the edge space

Bulat Kerimov [a,*], Maosheng Yang [c], Riccardo Taormina [b,1], Franz Tscheikner-Gratl [a,1]

[a] Department of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway
[b] Department of Water Management, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, The Netherlands
[c] Department of Intelligent Systems, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands

ARTICLE INFO

Keywords:
Water Distribution Networks
Hydraulic Simulators
Parallelization GPU

ABSTRACT

The steady state of a water distribution system abides by the laws of mass and energy conservation. Hydraulic solvers, such as the one used by EPANET approach the simulation for a given topology with a Newton-Raphson algorithm. However, iterative approximation involves a matrix inversion which acts as a computational bottleneck and may significantly slow down the process. In this work, we propose to rethink the current approach for steady state estimation to leverage the recent advancements in Graphics Processing Unit (GPU) hardware. Modern GPUs enhance matrix multiplication and enable memory-efficient sparse matrix operations, allowing for massive parallelization. Such features are particularly beneficial for state estimation in infrastructure networks, which are characterized by sparse connectivity between system elements. To realize this approach and tap into the potential of GPU-enhanced parallelization, we reformulate the problem as a diffusion process on the edges of a graph. Edge-based diffusion is inherently related to conservation laws governing a water distribution system. Using a numerical approximation scheme, the diffusion leads to a state of the system that satisfies mass and energy conservation principles. Using existing benchmark water distribution systems, we show that the proposed method allows parallelizing thousands of hydraulic simulations simultaneously with very high accuracy.

1. Introduction

Hydraulic simulations are an important tool for designing, managing, and controlling of water distribution systems (WDSs). Typical challenges that are approached by applying hydraulic simulations include real-time control (Mala- Jetmarova et al., 2017), district metered area (DMA) sectorization (Burrows et al., 2000), and vulnerability assessment (Klise et al., 2017). Frequently the output of a hydraulic simulator is coupled with an optimization algorithm to obtain the best solution among a multitude of alternative scenarios (Bello et al., 2019). Scenarios are evaluated via steady state simulations (or extended period simulations of steady states), yielding pressures at nodes, and flow rates in pipes based on known hydraulic head boundary conditions and expected water demands. Producing these steady states one by one, however, can be computationally costly (Garzón et al., 2022).

The estimation of a steady state is conventionally approached using linear theory (Todini and Rossman, 2013) or the well-established Global Gradient Algorithms (GGA) based on the Newton-Rhapson method (Todini and Pilati, 1988). To be physically plausible, the solution of the simulator must satisfy both, mass and energy conservation laws. However, the computational bottleneck arises from the costly matrix inversions employed by these solvers. Some of the previous works propose to approximate the matrix inversion, for example with an artificial neural network via algorithm unrolling (Solà Roca, 2023). Other approaches use simplified models, either by replicating the response of a hydraulic simulator (Kerimov et al., 2023; Xing and Sela, 2022) or using reduced models (Shamir and Salomons, 2008). An alternative algorithm based on cellular automata showed significant speed-ups of the GGA algorithm on field-programmable gate arrays (FPGAs) (Suvizi et al., 2023). However, FPGAs are not widely adopted due to power and hardware cost.

The key advantage of computational circuits like FPGAs and GPUs stems from the ability to efficiently parallelize vectorized operations and matrix multiplications. Parallelizing computation within a single

* Corresponding author.
E-mail addresses: bulat.kerimov@ntnu.no (B. Kerimov), m.yang-2@tudelft.nl (M. Yang), r.taormina@tudelft.nl (R. Taormina), franz.tscheikner-gratl@ntnu.no (F. Tscheikner-Gratl).
[1] These authors share senior authorship.

https://doi.org/10.1016/j.watres.2024.122980
Received 16 July 2024; Received in revised form 9 December 2024; Accepted 13 December 2024
Available online 31 December 2024

simulation is often referred to as fine-grained parallelization, and more details about it can be found in Burger et al. (2016). By identifying and parallelizing the computational bottlenecks, i.e. linear solvers and exponentiation in the pipe headloss calculation, they achieved a roughly 30% more speed increase (Guidolin et al., 2012). Although GPUs showed a potential for computational gains for very large networks (Crous et al., 2012), the speed-up of a single simulation is still limited.

Instead, we propose to focus on achieving speed-ups through *coarse-grained* parallelization, which inherently involves executing multiple simulation tasks in parallel. Unlike fine-grained parallelization, which optimizes the performance of individual computations, coarse-grained parallelization addresses the simultaneous evaluation of multiple steady states. The key advantage of this approach lies in the computational savings obtained by avoiding the sequential execution of simulations. One effective strategy for coarse-grained parallelization is leveraging multiple CPU cores to perform concurrent simulations, particularly during calibration tasks in urban water systems (Mair et al., 2014). However, the scalability of CPU-based parallelization is inherently limited by the number of available cores. Tasks such as quantifying uncertainty using Monte Carlo sampling methods (Pérez et al., 2015; Steffelbauer and Fuchs-Hanusch, 2016) or evaluating the fitness of a population in optimization problems (Dragan A. Savic and Jonkergouw, 2009) often require thousands of simulations to be performed simultaneously. In such scenarios, coarse- grained parallelization can significantly reduce computational time, making it a practical and efficient solution for circumventing computational load.

Further computational gains can be achieved by leveraging a sparse topology of WDSs. As WDSs can be represented as graphs, where every junction is a node and every pipe is an edge of the graph, one can represent their connectivity with sparsely connected matrices (e.g. adjacency or Laplacian). These matrices can be represented in a condensed form, such as compressed sparse columns, coordinate lists, and others. Modern GPUs can perform a multitude of parallel sparse matrix multiplications, offering significant potential for faster computation (Yang et al., 2018; Yan et al., 2017). Accordingly, WDSs' topologies expressed in sparse matrix form via adjacency or Laplacian matrices enable GPU acceleration for Graph Signal Processing (GSP) to perform state estimation (Shi et al., 2018; Zhou et al., 2022).

GSP is a framework for processing data defined on graphs, where signals are associated with the nodes or edges of a graph. By multiplying a Laplacian matrix with a vector of node signals (for example, pressure on nodes), the signal is *diffused*, and values from neighboring nodes are mixed, similar to heat diffusion in Euclidean space (Bai and Hancock, 2004). The signal on the edges can be of similar or even greater importance. Recent applications, for example, leveraged the diffusion of the edge signal to classify graphs (Aktas and Akbas, 2021), synchronize a dynamical system (Gambuzza et al., 2021), and identify consensus on a networking structure (DeVille, 2021; Ziegler et al., 2022). Previous works clearly showed the benefits of edge-centered representation in metamodeling of the hydraulic simulator using GNNs, as such formulation inherently incorporates mass conservation (Kerimov et al., 2024).

Based on these premises, we propose a novel GPU-accelerated algorithm for state estimation in WDS based on edge-based diffusion with boundary conditions of demands and reservoir heads. We formulate the problem in the edge space as this inherently incorporates the principles of mass conservation governing WDSs. By including the loop connectivity in the formulation we similarly integrate energy conservation law. The iterative algorithm adjusts the flowrates at each step according to the errors in mass and energy conservation, until it converges. After convergence, pressures are reconstructed from headlosses by traversing the network's spanning tree from multiple nodes with fixed head (e.g., reservoirs). By leveraging efficient sparse matrix multiplication and aggregation on GPUs, the proposed approach allows the concurrent and highly accurate estimation of a multitude of steady states with different boundary conditions at a fraction of the computational costs.

The rest of the paper is organized as follows: in the method section, we first introduce the diffusion process, adapt it to the edge space of a graph, and show how to rephrase a problem of steady-state estimation as a diffusion equation. We support the findings by showcasing the convergence patterns on different available benchmarking networks in the open literature and evaluate the speed-ups due to parallelizations. Subsequently, we highlight the main findings and hint at possible directions for future work.

2. Method

The section starts with the mathematical formalization of steady state estimation in WDS derived from key physical laws (Section 2.1) and describes input parameters. It then outlines three main steps of the state estimation method.

The overview of the method is presented in Fig. 1. As we base the state estimation on the edge space, we first define the connectivity of the edges via common nodes and loops and show the relationship of the connectivity with conservation laws. To incorporate the nodal input values, such as demands and reservoir heads, we augment the WDSs with virtual edges (Section 2.2) that are incidental to real nodes. These edges are assigned with the values of corresponding real nodes and are involved in the diffusion process as *boundary conditions* (BCs). Using the edge-based representation and BCs, the second step estimates the flowrates and headlosses with a diffusion on the edge space (Section 2.3). After a description of the method, the theoretical conditions for stability are presented (Section 2.4). Finally, the pressures on the nodes are derived based on the calculated headlosses, (Section 2.5).

2.1. Problem statement

We start with an oriented graph representation of a WDS, $G = G(\mathscr{V}, \mathscr{E})$. The graph consists of a vertex set $\mathscr{V}$ with the subset $\mathscr{V}_c \subseteq \mathscr{V}$ of N_c consumption nodes and the subset $\mathscr{V}_r \subseteq \mathscr{V}$ of N_r reservoirs, and the set of edges $\mathscr{E}$ with total E edges. Within graph G we additionally highlight the set of cells $\mathscr{C}$ which are also named loops within the WDS research community. An example of a cell is depicted in Fig. 2. These are different from the self-loops in graph theory, which refer to the connections of nodes to themselves. There are in total C cells that are oriented e.g. clockwise or anticlockwise. For a given oriented cell c, we can denote the set of edges it comprises as $\mathscr{E}_c$. Thus, $\mathscr{E}_c \subseteq \mathscr{E}$, and each edge $e \in \mathscr{E}_c$ belongs to the oriented cell c. Tanks are not considered in this representation.

Next, we define the incidence matrices as portrayed in Fig. 2 The incidence between the nodes and edges of a graph can be packed in a matrix of *lower* incidence $\mathbf{B}_1 \in \mathbb{R}^{N \times E}$, where $b_{ij} = 1$ if an edge j is pointed towards the node i. If the edge is leaving the node i then $b_{ij} = -1$. Finally, if the edge is not incident to the node the b_{ij} is zero. Similarly, we can define connectivity between edges and cells with a matrix of *upper* incidence $\mathbf{B}_2 \in \mathbb{R}^{E \times C}$. Here, $b_{ij} = 1$ if the orientation of an edge i and a corresponding cell c coincide and $b_{ij} = -1$ if their orientations are opposite, and $b_{ij} = 0$ otherwise. The example is presented in Fig. 2.

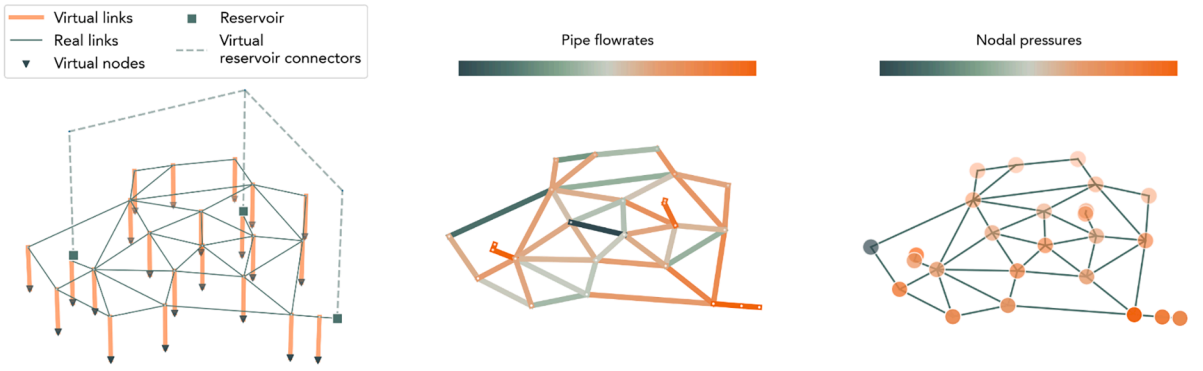
We define as nodal inputs (or nodal signal) as known demands $\mathbf{q} \in \mathbb{R}_{N_c}$ and known heads $\mathbf{p}_r \in \mathbb{R}^{N_r}$. The vector $\mathbf{p}_c \in \mathbb{R}^{N_c}$ represents the desired heads on consumption nodes. Together they comprise the vector of heads $\mathbf{p}$.

The desired output $\mathbf{f} \in \mathbb{R}^E$ represents the vector of flowrates. Negative values in the vector indicate that the true direction of the flow is opposite to the selected orientation. The orientation of edges and cells is chosen arbitrarily. We do not consider valves in this representation.

2.1.1. Mass conservation

Mass conservation on the node u simply states that the flow of incoming water equals to the sum flow of the outflowing water (i.e. the demand $\mathbf{q}(u)$). In matrix form, it is characterized as follows

$$\mathbf{B}_1\mathbf{f} = \mathbf{q} \tag{1}$$



(a) Augmentation with virtual sinks and reservoir connectors described in Section 2.2

(b) Approximation of flowrates and headlosses with an iterative scheme (Section 2.3)

(c) Reconstruction of heads by traversing the shortest paths from the reservoirs described in Section 2.5

Fig. 1. Main steps of the steady state estimation via edge diffusion.

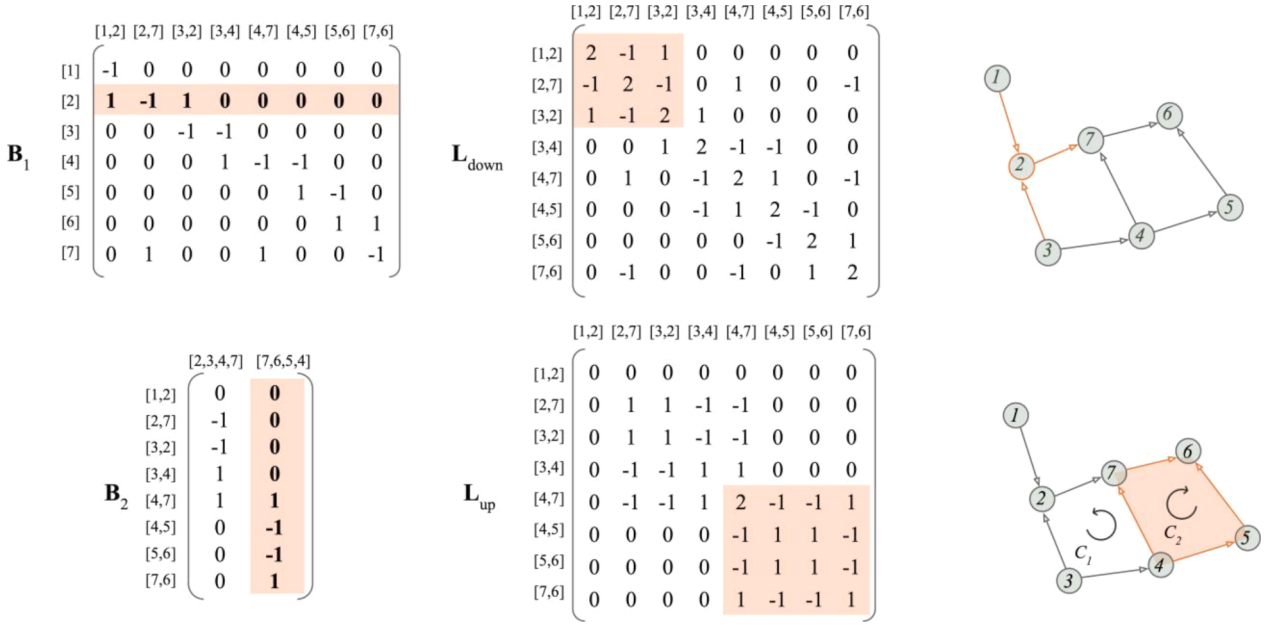


Fig. 2. An example of lower- and upper incidence matrices, Laplacian matrices, and connectivities via common nodes and common cells.

Conceptually $\mathbf{B}_1$ is a linear operator that measures the *divergence* of the edge flow on nodes. Divergence of an arbitrary edge signal on the node u is the summation of edge values on the incidental edges with their corresponding orientations (Lim, 2020) This translates the edge signal to node space

$$(\mathrm{div}\mathbf{f})(u) = \sum_{e=(v,u)\in\mathscr{E}} \mathbf{f}(e) - \sum_{e=(u,v)\in\mathscr{E}} \mathbf{f}(e) \tag{2}$$

Subsequently, $\mathbf{B}_1\mathbf{f}=0$ the edge flow is divergence-free, which happens when the inflow to the node is equal to the outflow while demands or reservoir inflows are equal to zero. By principle, divergence-free flow is mass conservative.

The divergence-free condition can be reformulated with a lower Laplacian (or down-Laplacian) $\mathbf{L}_{\text{down}}$ into

$$\mathbf{B}_1^\top\mathbf{B}_1\mathbf{f} = \mathbf{L}_{\text{down}}\mathbf{f} = 0 \tag{3}$$

The lower Laplacian is a result of a matrix multiplication of a transposed lower incidence with itself. It denotes the the connectivity of the edges via common nodes. The diagonal entries of $\mathbf{L}_{\text{down}}$ indicate the total number of nodes that an edge is incidental to which is always 2. If two edges are directed towards the same node (both "heads" converge on the node) or if their "tails" originate from the same node, the entry is $+1$, otherwise, the entry is -1. The example of $\mathbf{L}_{\text{down}}$ can be found in Figure, along with the lower incidence matrix. Lower Laplacian is widely topological data analysis and signal processing, particularly when dealing with data defined on the elements of graphs and higher-order structures (Schaub et al., 2020; Yang et al., 2022). Likewise, the matrix can be used as a measure of the divergence.

2.1.2. *Energy conservation*

The energy balance states that the difference in heads on adjacent nodes is a function of flowrates

$$\mathbf{h} = \mathbf{B}_1^\top\mathbf{p} = \mathbf{\Phi}(\mathbf{f}) \tag{4}$$

where $\mathbf{h} \in \mathbb{R}^E$ is the energy loss in the pipe, and $\mathbf{\Phi}$ is the empirical relationship between flowrates and headlosses. Within a pipe e the relationship takes the following form

$$\mathbf{h}(e) = \kappa_e \cdot |\mathbf{f}(e)|^a \cdot \text{sign}(\mathbf{f}(e)) \tag{5}$$

where $\kappa \in \mathbb{R}^E$ is the resistance coefficient, and a is a flow exponent (e.g. a equals 2 for Darcy-Weisbach and 1.852 for Hazen-Williams). The resistance coefficient of a pipe e in Hazen-Williams formulation is calculated as follows

$$\kappa_e = \frac{10.67 \cdot l_e}{r_e^{1.852} d_e^{4.870}} \tag{6}$$

Here, l, d, and r refer to pipe lengths, diameters, and roughnesses, respectively. A reverse relationship is denoted with $\mathbf{\Phi}^{-1}$ and takes the following form

$$\mathbf{f}(e) = \left((\kappa_e)^{-1} \cdot |\mathbf{h}(e)|\right)^{\frac{1}{a}} \cdot \text{sign}(\mathbf{h}(e)) \tag{7}$$

The law of energy conservation states that the sum of energy losses within a loop c (with respect to their orientation) is equal to 0. The operation of aggregation of the edge signal around the edges that forms a cell is denoted as *curl*

$$(\text{curl}\mathbf{h})(c) = \sum_{e \subset c} \mathbf{h}(e) = 0 \tag{8}$$

In fact, the upper incidence $\mathbf{B}^{\mathrm{T}}$ can act as a curl operator. Using matrix notation, the relationship takes the following form

$$\mathbf{B}_2^{\top}\mathbf{h} = 0 \tag{9}$$

Analogously to Equation 3 The energy conservation can be expressed with an *upper* Laplacian (or up-Laplacian matrix, i.e.

$$\mathbf{L}_{up}\mathbf{h} = \mathbf{B}_2\mathbf{B}_2^{\top}\mathbf{h} = 0 \tag{10}$$

The upper Laplacian denotes the connectivity between edges via common cells. The diagonal entries of $\mathbf{L}_{\text{up}}$ indicate the number of cells to which a given edge belongs. The off-diagonal entries indicate whether two edges are part of the same cell. If two edges share a cell and have aligned orientations (both contributing consistently to the boundary of the cell), the entry is $+1$. If their orientations oppose each other within the shared cell, the entry is -1.

2.2. Network augmentation

In WDS topology, the input parameters, such as heads and demands, are defined on the nodal space. These values act as BCs and drive the steady state estimation, so they must be transposed to the edge space. For that reason, we augment the network with virtual *sinks* incidental to consumption nodes and incorporate nodal demands. Similarly, we connect reservoirs with virtual *reservoir connectors* to account for differences in reservoir piezometric heads. After the augmentation, known demands and head differences can be represented as a known part of $\mathbf{f}$ and $\mathbf{h}$ correspondingly.

2.2.1. Virtual sinks

As illustrated in Equation 1, one must consider nodal water consumption for mass conservative flow. To translate them to the edge space we create a set $\mathscr{V}_v$ of N virtual nodes and couple every real node with the corresponding virtual node, as visualized in Fig. 1a. Thus the edges are augmented with a set $\mathscr{E}_s$, where $\mathscr{E}_s = \{(u, u_v) \mid u \in \mathscr{V} \text{ and } u_v \in \mathscr{V}_v\}$. The demands on the corresponding real node u are assigned to a flowrate on the edge $e = \{u, u_v\}$.

$$\mathbf{f}(e) = \mathbf{q}(u),\ e \equiv (u, u_v) \in \mathscr{E}_s\ (\text{BC}) \tag{11}$$

If we consider a demand-driven simulation, the flowrate on virtual sink values becomes a boundary condition for the state estimation on the edge space. Here we assume that all consumed water flows through those sinks.

2.2.2. Virtual reservoir connectors

In a demand-driven simulation in a network with a single reservoir, reservoir heads affect pressures but do not influence flowrates or headlosses. As our method derives pressures based on the headlosses, this step is not required for singe-reservoir networks. However, in scenarios involving multiple reservoirs, the heads of the reservoirs (specifically, the differences in heads) impact the pressures, flowrates, and headlosses altogether.

Similarly to demands, reservoir heads are nodal signals and must be translated into edge signals. We introduce a boundary condition for energy conservation (e.g. known reservoir heads) with virtual reservoir connectors for water networks with multiple reservoirs. We identify pairs of reservoirs with the shortest graph distance and connect them with a set of edges $\mathscr{E}_r$, with $N_r - 1$ virtual connectors in total. Each new edge will now be considered as a boundary condition to the energy balance. Essentially, it encodes the difference in the heads between virtually connected reservoirs, i.e.

$$\mathbf{h}(e) = \mathbf{p}(v) - \mathbf{p}(u), \mathrm{e} \equiv (\mathrm{u},\ \mathrm{v}) \in \mathscr{E}_{\mathrm{r}}(\mathrm{BC}) \tag{12}$$

The same relationship in vector format takes the form

$$\mathbf{h}_{(r)} = \left(\mathbf{B}_1^{(r)}\right)^{\top}\mathbf{p}_r \tag{13}$$

where $\left(\mathbf{B}_1^{(r)}\right)^{\top} \in \mathbb{R}^{N_r - 1 \times N_r}$ is the lower incidence matrix between reservoirs and the virtual reservoir connector. Vector $\mathbf{h}_{(r)}$ is now a part of $\mathbf{h}$ defined on virtual reservoir connectors. Schematic depiction can be found in Fig. 1a, where the connectors are denoted with a dashed line.

Since each reservoir is connected to the rest of the system via a single pipe, it is usually not incident to any cell. Thus the projection $\mathbf{B}_2^{\top}\mathbf{h}$ would not include the reservoir heads and their difference. It becomes possible, once we introduce a virtual connection. Each virtual connection introduces an additional cell to the WDS. In a scenario with a single reservoir network, this step can be omitted as the reservoir inflow can be assumed to be the sum of demands and act as a BC for mass conservation.

2.3. State estimation as a diffusion on the edge space

Following up, this section describes the second step of the method. It begins with a general description of the diffusion equation on the euclidean space and continues with the formulation of state estimation of WDS as a diffusion on the edge space.

2.3.1. Diffusion recap

Let us consider a diffusion process of a certain function $\mathbf{u}(x,t)$ on a one-dimensional Euclidean space $\mathbf{x}$. There, the temporal change of $\mathbf{u}$ within some medium is proportional to the spatial variability, defined with a Laplacian Δ.

$$\frac{\partial \mathbf{u}(x,t)}{\partial t} = -\Delta \mathbf{u}(x,t) \tag{14}$$

Eventually, this process reaches a steady state, which is characterized by

$$\frac{\partial \mathbf{u}(x,t)}{\partial t} = 0 \tag{15}$$

The solution to PDEs is often approximated by numerical procedures. The most straightforward approach is the forward Euler discretization:

$$\mathbf{u}_{t+1} \leftarrow \mathbf{u}_t - \tau \Delta \mathbf{u}_t \tag{16}$$

where τ is a selected timestamp. In such a setting the diffusion process describes a minimization of a Dirichlet energy with a *gradient descent* (for more details refer to Calder (2020)). From now on, we refer to the discretization of the diffusion process as gradient descent. That allows to accelerate convergence by augmenting Equation 16 with the momentum term (Polyak, 1964; Calder and Yezzi, 2019):

$$\mathbf{u}_{t+1} \leftarrow \mathbf{u}_t - \tau\Delta\mathbf{u}_t + \beta(\mathbf{u}_t - \mathbf{u}_{t-1}) \tag{17}$$

Here, the second term adds the difference from a previous step with a weighting coefficient β, usually taken between 0 and 1, and averages the gradients over time.

2.3.2. Flowrate and headloss reconstruction as diffusion on the edge space

Both $\mathbf{L}_{\text{down}}$ and $\mathbf{L}_{\text{up}}$ serve as discrete equivalent of the Laplacian operator. This analogy led to the construction of the diffusion process in the edge space (Schaub et al., 2020; DeVille, 2021). The steady-state condition defined in Equation 15 thus can be expressed with discrete laplacians. Moreover, they are directly associated with the conservation laws, as we show in Sections 2.1.1 and 2.1.2. The connection between the diffusion process and the conservation laws thus provides a physically-based way to estimate the steady state of a WDS.

We can now formulate the system of differential equations that describe the state estimation of a Water Distribution System (WDS).

$$\frac{\partial \mathbf{f}(e,t)}{\partial t} = -\mathbf{L}_{\text{down}}\mathbf{f}(e,t)$$

$$\frac{\partial \mathbf{h}(e,t)}{\partial t} = -\mathbf{L}_{\text{up}}\mathbf{h}(e,t)$$

$$\mathbf{h}(e,t) = \kappa_e \cdot |\mathbf{f}(e,t)|^a \cdot \text{sign}(\mathbf{f}(e,t)) \quad e \in \mathscr{E} \tag{18}$$

$$\mathbf{f}(e,t) = q_e \quad e \in \mathscr{E}_s$$

$$\mathbf{h}(e,t) = \mathbf{p}(v) - \mathbf{p}(u) \quad e \equiv (u,v) \in \mathscr{E}_r$$

These equations describe the evolution of the steady state. The vectors $\mathbf{f}(t)$ and $\mathbf{h}(t)$ are adjusted according to the errors in mass and energy conservation and gradually move to the desired state. The third equation ensures that the vectors adhere to the Hazen-Williams relationship. The desired solution can be expressed as

$$\mathbf{f}^{\infty} = \lim_{t\to\infty} \mathbf{f}(t), \qquad \mathbf{h}^{\infty} = \lim_{t\to\infty} \mathbf{h}(t)$$

Similar to the condition defined in Equation 15, the steady state of such diffusion is characterized by the conditions $\mathrm{L}_{\text{up}}\mathbf{h}^{\infty} = 0$ and $\mathrm{L}_{\text{down}}\mathrm{F}^{\infty} = 0$.

The last two equations are the BC of the diffusion. When WDS is equipped with a single source, the reservoir outflow can be calculated as a sum of demands and used as a BC. However, in multiple reservoir scenarios, the outflow is unknown and must be estimated as a part of the diffusion. The divergency-free condition on virtual nodes that are coupled with the reservoirs will thus enforce a zero flow. To generalize the approach beyond a single source, the $\mathbf{L}_{\text{down}}$ is constructed from the modified incidence $\overline{\mathbf{B}}_1$. There, the corresponding entries b_{ij} where $i \in \mathscr{V}_r$ and $j \in \mathscr{E}_s$ are equal to zero. This adjustment releases the divergence-free condition from the ends of the virtual sinks that correspond to the reservoir nodes and allows for estimating the inflows. As a result, the inflow is calculated based on the mass balance and the energy balance.

2.3.3. Iterative algorithm

Similarly to a one-dimensional case in Equation 17, the edge-based diffusion can be approximated with a numerical scheme. Thus, we perform the state estimation with two simultaneous gradient descents: the first one using the lower and the other with the upper Laplacian. We denote these iterative approximations as *lower* and *upper* gradient descents correspondingly. Each step of gradient descent includes momentum, similar to Equation 17 and enforces a BC from Equations 11 and 12. The BCs are ensured before every gradient step by substituting the values with $\mathbf{q}$ and $\mathbf{h}_{(r)}$ at virtual sinks and headlosses on reservoir connectors, denoted with $\mathbf{f}_{(v)}$ and $\mathbf{h}_{(v)}$ respectively. The gradient descents are connected with each other with frictional head loss relationship $\mathbf{\Phi}$ and its reverse $\mathbf{\Phi}^{-1}$. That is, the flowrates are transformed into headlosses after lower gradient descent. Vice-versa, the headlosses are transformed into flowrates after upper gradient descent. The algorithm is parameterized by timestep sizes (τ_{down} and τ_{up}) and momentum coefficients (β_{down} and β_{up}) for lower and upper gradient descents correspondingly, as well as the number of lower gradient steps K.

First, the vector of flowrates $\mathbf{f}^0$ is initialized with zeros. The main loop of the algorithm begins with K steps of lower gradient descent towards mass conservative flow. After that, the resulting flowrates are transformed into headlosses $\mathbf{h}^i$ via Equation 5. The algorithm follows up with a step of upper gradient descent towards energy-conservative flow. The headlosses are then transformed back to flowrates, and new flowrates will be used as an initial condition for the next iteration. The pseudocode is presented in Algorithm 1. The schematic depiction and more information can be found in Appendix A

In the iterative algorithm described above, one of the most computationally demanding operations is the repeated multiplication of the sparse Laplacian matrices ($\mathbf{L}_{\text{down}}$ and $\mathbf{L}_{\text{up}}$) by the dense vectors of flowrates $\mathbf{f}$ or headlosses $\mathbf{h}$.

These Laplacian matrices are inherently sparse, as they are derived from network connectivity domains where each node (or element) is typically connected to only a few neighbors. Such sparsity patterns significantly reduce the number of non-zero elements, allowing for more efficient memory storage and computation (see Appendix B)

2.4. Convergence and stability

The convergence and stability of the gradient descent with a constant timestep is dependent on the value of τ (Portaet al., 2017). The convergence of the discretization scheme without momentum stems from the condition that every iteration of Equation 16 magnifies $\mathbf{f}$ and by $\mathbf{h}$ less than 1, i. e.

$$|\mathbf{I} - \tau_{\text{down}}\mathbf{L}_{\text{down}}| < 1$$

$$|\mathbf{I} - \tau_{up}\mathbf{L}_{up}| < 1 \tag{19}$$

If $0 < \lambda_1 < \lambda_2 < \ldots < \lambda_{\max}$ are the eigenvalues of a Laplacian $\mathbf{L}$, it is necessary to select the timestamp s.t. $|1 - \tau\lambda_{\max}| < 1$. This condition guarantees that iterative amplification with the Laplacian matrix will not lead to instability in the numerical solution process. Momentum introduces additional complexity to the stability and the bound must adhere to the following relationship (Qian, 1999)

$$0 < \tau < \frac{2(1+\beta)}{\lambda_{\max}} \tag{20}$$

For a general gradient descent, a larger timestep means faster convergence but can lead to overshooting. The optimal timesteps, as well as momentum coefficients, can be identified with an optimization procedure.

The iterative scheme can be run infinitely, however, one can establish a stop condition based on the errors in mass and energy conservation. Ultimately, with a suitable set of parameters, it converges towards both mass and conservative solutions on the limit. Therefore, it is possible to measure convergence of the diffusion when the mean absolute errors (MAE) in mass conservation over all junctions and MAE over all loops reaches a certain threshold, i.e.

$$\epsilon_{mass} = \frac{1}{N_c}\left|\left|\mathbf{B}_1^{(c)}\mathbf{D}_0^{-1}\mathbf{f} - \mathbf{q}\right|\right| \tag{21}$$

$$\epsilon_{energy} = \frac{1}{C}\left|\left|\mathbf{B}_2^{\top}\mathbf{D}_2^{-1}\mathbf{h}\right|\right|$$

where $\mathbf{B}_1^{(c)} \in \mathbb{R}^{N_c \times E}$ denotes incidence matrix only for consumption nodes. Both mass and energy conservation laws are now normalized with diagonal matrices of node and cell degrees, $\mathbf{D}_0 \in \mathbb{R}^{N_c \times N_c}$ and $\mathbf{D}_2 \in \mathbb{R}^{C \times C}$ respectively, to obtain the mean errors. We chose thresholds for ε_{mass} as $10^{-2}\,L/s$ and for ε_{energy} as $10^{-1}\,m$ to balance between stability,

Algorithm 1
Iterative approximation

Data: Initialize vectors $\mathbf{f}^0$, $\mathbf{f}^1$, $\mathbf{h}^0$, and $\mathbf{h}^1$, with zeros
$i \leftarrow 1$;
while *not converged* **do**
  for *K iterations* **do**
    $\mathbf{f}^{i-1} \leftarrow \mathbf{f}^i$;
    /* Lower gradient descent */
    $\mathbf{f}^i_{(v)} \leftarrow \mathbf{q}$; // BC
    $\mathbf{f}^i \leftarrow \mathbf{f}^i - \tau_{\text{down}}\mathbf{L}_{\text{down}}\mathbf{f}^i + \beta_{\text{down}}\left(\mathbf{f}^i - \mathbf{f}^{i-1}\right)$; ;
  end for
  $\mathbf{h}^i \leftarrow \mathbf{\Phi}(\mathbf{f}^i)$; // Transform flowrates to headlosses
  /* Upper gradient descent */
  $\mathbf{h}^i_{(v)} \leftarrow \mathbf{h}_{(r)}$; // BC
  $\mathbf{h}^i \leftarrow \mathbf{h}^i - \tau_{\text{up}}\mathbf{L}_{\text{up}}\mathbf{h}^i + \beta_{\text{up}}\left(\mathbf{h}^i - \mathbf{h}^{i-1}\right)$;
  $\mathbf{f}^i \leftarrow \mathbf{\Phi}^{-1}(\mathbf{h}^i)$; // Transform headlosses back to flowrates
  /* Update */
  $\mathbf{f}^{i+1} \leftarrow \mathbf{f}^i$;
  $\mathbf{h}^{i+1} \leftarrow \mathbf{h}^i$;
  $i \leftarrow i + 1$;
end while

speed of the convergence, and the precision. Depending on the required accuracy, the error threshold can be adjusted.

2.5. Hydraulic head reconstruction

Finally, the obtained $\mathbf{h}^\infty$ is used to derive the heads on the nodes of the system. Once the approximated values of head losses at each pipe are known, the nodal heads are simply extracted by traversing the network from the source nodes (i.e. reservoirs) to the consumption nodes. This is analogous to the calculation of the shortest path from multiple sources on the graph via Dijkstra algorithm (Todini and Rossman, 2013; Dijkstra, 2022). The distance on each edge is defined by the value of headlosses $\mathbf{h}^\infty$. The traversal can similarly implemented within a sparse aggregation framework and requires up to D steps, where D is the topological diameter of a network. Thus, it is performed at a fraction of the time of the main algorithm.

3. Experiments

3.1. Case studies

The evaluation of the method is based on several benchmark networks of different configurations and sizes. The main information about those networks can be found in Table 1. They range from small to large benchmarks, according to the work of Wang et al. (2015). The number of pipes in the WDS range from 34 (JILIN and APULIA) to 1274 (KL) with corresponding network diameters, that is the longest shortest path between any two nodes in the WDS, ranging from 10 to 53. Three WDS feature multiple reservoirs (PES, NET-3, and MOD). The case studies employ the original demands and pipe parametrization from the *.inp* files. All *.inp* files present snapshot simulations, with the exception the L-Town network (Area C from Vrachimis et al. (2022)) which features an extended period simulation of 168 hours (10000 hydraulic timesteps). The layout of the WDSs is displayed in Appendix C.

3.2. Parameter study

As was already stated in Section 2.4, suitable timesteps and momentum values lead to faster convergence. As the theoretical identification of the optimal parameters is not straightforward, it is essential to understand the influence of the parameters on the convergence. Thus, we first introduce an optimization procedure for defining the optimal parameters. Further on, we delve deeper into the impact of timestep size. Lastly, we investigate the sensitivity of the optimal parameters to the

Table 1
Description of water networks used in the experiments ordered by the number of pipes.

Name	Network **diameter** D	Pipes	Reservoirs	Cells	Reference
JILIN	10	34	1	7	Bi and Dandy (2014)
APULIA	10	34	1	11	Hall (2021)
BAK	14	58	1	16	Lee and Lee (2001)
ASnet2	14	65	1	15	Xing and Sela (2022)
PES	22	99	3	28	Bragalli et al. (2012)
L-TOWN	22	109	1	17	Vrachimis et al. (2022)
NET-3	30	119	3	23	Rossman (2016)
ZJ	26	164	1	51	Dandy (2016)
MOD	38	317	4	46	Bragalli et al. (2012)
KL	53	1274	1	334	Kang and Lansey (2012)

change in demand inputs.

3.2.1. Parameter optimization

The optimal parameters depend on network topology. To maximize the benefit from the parallelization the most, one could determine the optimal parameters based on one or a small set of simulations. Suitable parameters can then be used for model-based applications that require evaluation of multiple simulations. Theoretically, Equation 19 determines the timestep bounds for a single, uncoupled gradient descent. As our method involves two simultaneous gradient descents, they might influence or disrupt the convergence of each other. The parameter set consists of timestep sizes (τ_{up} and τ_{down}), momentum coefficients (β_{up} and β_{down}), and the number of lower gradient steps K (1,3, or 5) within each iteration. We propose to perform a two-stage search for the optimal set of parameters. During the first stage, we separately identify the real range of τ_{down} and τ_{up} where the corresponding gradient descents do not diverge within 2000 iterations. The momentum coefficients are fixed at 0. Once the timestep size bounds are identified, the Bayesian Optimization procedure (Nguyen, 2019) defines the best set of parameters. The optimization criteria evaluate the number of iterations n until convergence.

We selected as a convergence criterion R^2 between the reconstructed flowrates and the one obtained from EPANET reaches 0.999. In reality, one would not need to run EPANET simulation and the convergence could be based on the conservation error conditions in Equation 21. However, the absolute values of errors can be specific to each network, thus we chose R^2 in flowrates as a stop criterion in favor of simplicity and universality across networks.

In the edge-based formulation, the influence of friction energy losses on the steady state is an influential factor defining the convergence and speed of the edge-based algorithms (Kerimov et al., 2024). When energy conservation doesn't affect the distribution of flowrates in the pipes, the steady state can be estimated to a large extent with lower gradient descent, with the upper gradient descent acting as a correction term. On the flip side, when the influence of energy losses is high, the flowrates require more correction. Pipes with high κ may indicate those cases. To quantify the influence, one can run the steady algorithm without the correction of upper gradient descent, i.e. selecting $\tau_{up} = 0$ The energy losses $\epsilon_{\text{energy}}|_{\tau_{up}=0}$ when $t \to \infty$ indicate the amount of influence of energy conservation on the solution.

3.2.2. Stability regions

It is important to understand how these parameters influence the convergence of the method. Thus, we visually examine the stability regions for gradient descent without momentum to study parameter space. Specifically, we evaluate the number of steps to reach convergence (n) at varying τ_{up} and τ_{down}. The analysis excludes the momentum to facilitate visualization in two dimensions. Furthermore, we visualize convergence dynamics using a selected set of sampled parameters from the stability regions plot to assess convergence dynamics in detail.

3.2.3. Robustness of optimal parameters under changing demands

In practical applications, water demands in WDS change continuously. Ensuring that the algorithm maintains consistent performance across these variations is essential for its practical utility. Several problems of interest require finding solutions across varying demand conditions to tackle uncertainty, leakage detection, and sensor placement (Steffelbauer et al., 2014). The next study evaluates the influence of varying demand patterns on the behavior of the algorithm with the optimal set of parameters. By investigating the convergence time in the context of different demand scenarios, we thus assess the robustness of identified optimal parameters. In the example of a week of an extended period simulation (EPS) of the area C of L-Town network, we identify optimal parameters using a single demand pattern at $t_H = 0$, denoted as $\mathbf{q}_0$. Following up, we measure the convergence time for each hydraulic timestep t_H with the same parameter set and visualize the results.

These dynamics are compared with the cosine similarity of the vectors of $\mathbf{q}$ at each timestep to $\mathbf{q}_0$. We then select demands that correspond to peaks of the largest deviation in terms of n. The experiment concludes with the visualization of the stability regions of the pair (τ_{down}, τ_{up}) at the peaks with fixed β_{up} and β_{down}.

3.3. Evaluation of speed-up

This part of the experiments assesses the performance of the diffusion approach by measuring the clock time T needed to solve M simulations in a parallel batch. The result is then compared with the non-parallelized EPA counterpart T_{EPA}, quantifying the speed-up S through the relationship:

$$S = \frac{T}{T_{EPA}}$$

The speedup was evaluated separately for every WDS in the case study. An additional measure of the batch size E_M, is determined by the total pipes (E) multiplied by M, directly related to sparse aggregation performance (Yang et al., 2018). Other overhead costs related to the preprocessing, identification of loops, and startup are not considered in the scope of this work. These operations can be performed in advance and don't depend on the number of simultaneous, while the constant overhead is not significant. For example, it is sufficient to find any *cycle basis* rather than enumerating all possible loops and computing a *minimum cycle basis* (Alvarruiz et al., 2015). In the example of KL and employing implementation with networkx (Hagberg et al., 2008) the identification of a cycle basis takes $2{\cdot}10^{-2}$ seconds. We conduct two evaluations of parallelization capabilities. In the first setting, we systematically increase the batch size from a single network until visual memory usage reaches 24 GB. Throughout this increment, we record corresponding speed-ups S with the accuracy of R^2 of 0.99 in flowrates and visualize them against M and E_M. The parallelization is evaluated on a computational cluster node with 24 GB virtual memory (NVIDIA RTX A5000, CUDA version 12.2). The second setting involves a standard workstation with 2 GB of visual memory (NVIDIA T500, CUDA version 11.4). For each network, we select M simulations compatible with the visual memory capacity. The algorithm runs in parallel until achieving convergence, indicated by R^2 of 0.99 and additionally by R^2 of 0.999 in flowrates. The implementation is coded in Python with PyTorch sparse matrix multiplication.

4. Results and Discussion

The section presents the results of the listed experiments and discusses the interpretations and limitations of our approach. First, we provide an overview of the optimal parameters for each WDS to illustrate their dependence on the case study. Further on, delve into the stability of the algorithm and discuss the robustness of the suitable hyperparameters to changing demands. Next, we present the results of the evaluation of speed-ups. Lastly, we evaluate the potential computational gains obtained from parallelization with the edge diffusion.

4.1. Parameter study

4.1.1. Parameter optimization

Table 2 showcases the obtained optimal parameters alongside the number of iterations n required for convergence.

The column $\lambda^{\max}_{\text{down}}$ in Table 1 displays that the largest eigenvalues of $\mathbf{L}_{\text{down}}$ are similar. At the same time, $\lambda^{\max}_{\text{up}}$ shows a larger variety of values. The optimal timesteps are largely defined by eigenvalues of the Laplacian. Naturally, one can observe that τup varies to a more extent than τdown. Traditionally, the convergence of a linear system of a form $\mathbf{Ax} = \mathbf{b}$, such as GGA, can be gauged by the condition number. A high condition number indicates sensitivity to numerical errors and potential

Table 2
Optimal timesteps, momenta, largest eigenvalues, number of lower gradient descents K, and the total number of iterations n required for convergence obtained from the Bayesian optimization.

Name	τ_{up}	τ_{down}	β_{up}	β_{down}	λ_{up}^{max}	λ_{down}^{max}	K	$\epsilon_{energy}\|_{\tau_{up}=0}$	n
APULIA	$5.6 \cdot 10^{-3}$	0.14	0.53	0.29	$2.5 \cdot 10^{1}$	7.2	5	$9.0 \cdot 10^{0}$	30
JILIN	$3.0 \cdot 10^{-2}$	0.20	0.44	0.77	$7.6 \cdot 10^{0}$	7.3	5	$8.1 \cdot 10^{0}$	10
BAK	$5.4 \cdot 10^{-4}$	0.04	0.18	0.10	$5.1 \cdot 10^{1}$	7.7	3	$5.2 \cdot 10^{2}$	160
ASnet2	$2.0 \cdot 10^{-2}$	0.27	0.62	0.48	$8.7 \cdot 10^{1}$	7.5	3	$5.1 \cdot 10^{-3}$	20
PES	$1.7 \cdot 10^{-3}$	0.27	0.69	0.24	$9.3 \cdot 10^{1}$	7.8	5	$2.6 \cdot 10^{1}$	110
L-TOWN	$9.1 \cdot 10^{-3}$	0.44	0.44	0.74	$1.4 \cdot 10^{2}$	7.2	3	$1.0 \cdot 10^{-3}$	20
NET-3	$1.5 \cdot 10^{-3}$	0.20	0.47	0.51	$4.3 \cdot 10^{1}$	7.3	5	$5.2 \cdot 10^{-2}$	300
ZJ	$1.8 \cdot 10^{-2}$	0.28	0.87	0.81	$8.5 \cdot 10^{1}$	8.0	5	$8.0 \cdot 10^{-2}$	15
MOD	$7.9 \cdot 10^{-4}$	0.24	0.40	0.10	$1.1 \cdot 10^{3}$	7.7	5	$1.0 \cdot 10^{0}$	430
KL	$5.9 \cdot 10^{-4}$	0.28	0.79	0.69	$1.7 \cdot 10^{3}$	7.7	5	$1.1 \cdot 10^{-1}$	200

difficulties in achieving fast convergence (Burger et al., 2016). However, for the Laplacian matrices used here— being singular and thus having at least one zero eigenvalue — the condition number is effectively infinite, rendering it an unsuitable stability measure. Instead, we focus on the distribution of eigenvalues. The spacing and magnitude of these eigenvalues, particularly the largest ones, fundamentally influence how quickly gradient-based methods converge. Larger eigenvalues often necessitate smaller timesteps to maintain numerical stability, which in turn affects the number of iterations required for convergence (Qian, 1999).

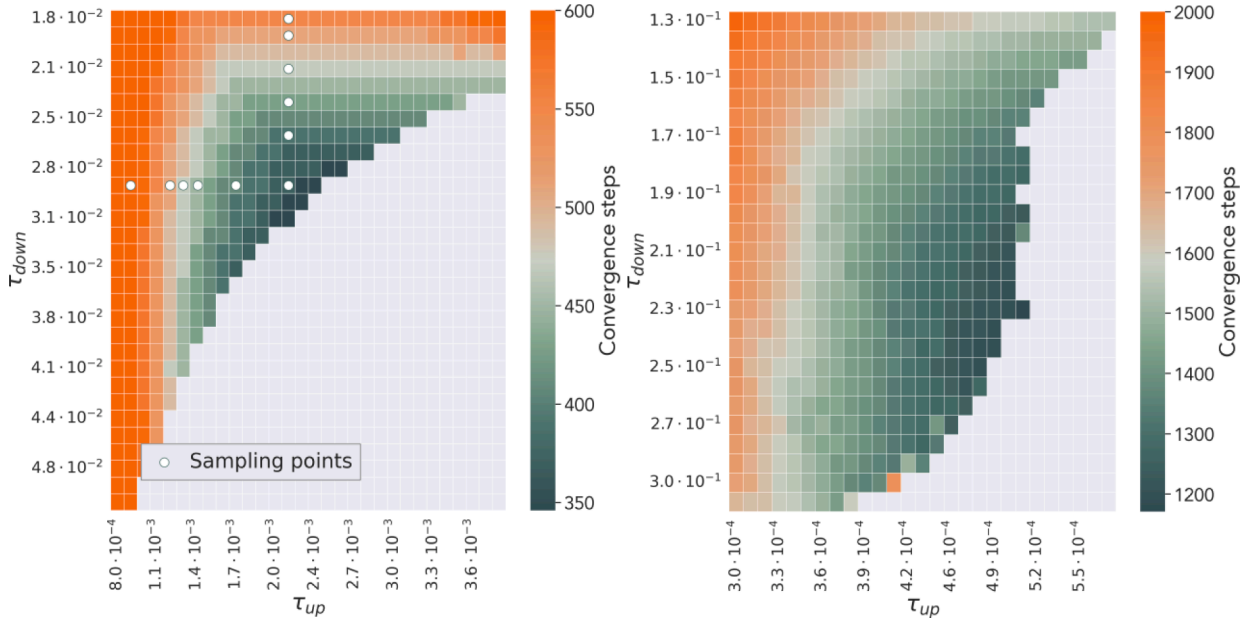
For all networks, the momentum term seems to benefit the convergence, which agrees with the theoretical findings of Polyak (1964). Further comparison of minimum n without momentum is attached to Appendix E.1. In general, a higher K is preferable and tends to decrease the total number of iterations n. More examples of the influence of K are included in Appendix E.2.

Expectedly, smaller networks converge faster. Due to the local nature of matrix multiplication, the solution propagates from edge to edge, thus smaller networks inherently require fewer iterations to achieve convergence. This is consistent with the analogy of a heat diffusion process and its numerical approximation. Within this perspective, the size of a network can be perceived as the granularity of discretization of a non-euclidean space. As such, a grid with finer resolution inherently requires a higher number of iterations. This hints at the equivalences between different water networks and steady states from the perspective of topology (Levie et al., 2019).

Although smaller networks typically take fewer steps for convergence, the same doesn't hold for BAK. It stands as an outlier and requires significantly more steps. This network is equipped with a pipe with a very small diameter (and, correspondingly, high κ). The influence of the friction to the convergence, measured with $\epsilon_{energy}|_{\tau_{up}=0}$, could explain the slower convergence. Notably, the previous work on surrogate modeling with edge-based graph neural networks similarly showed lower accuracy on BAK network (Kerimov et al., 2024).

4.1.2. *Stability regions*

Following up, Figs. 3a and 3b illustrate the stability regions for timestep values in the example of BAK and KL respectively with zero momenta. BAK network presents an outlier in terms of convergence rates, while KL is the largest network of the case study. The area of divergence is denoted with grey. For a single, uncoupled gradient descent, a larger timestep size typically results in faster convergence, given the condition outlined in Section 2.4 is satisfied. However, in a case with two coupled gradient descent, the process becomes susceptible to the combination of τ_{down} and τ_{up}. As a result, we observe a complex picture of stability regions in both figures. This sensitivity emerges because the steps of lower and upper gradient descents can be strong enough to disrupt the convergence of each other. The border between the convergence and divergence shows a non-linear shape. In the example of BAK, the border appears to be concave towards higher τ_{up}.



(a) BAK without momentum and K = 3. (b) KL without momentum and K = 5

Fig. 3. Stability regions. Convergence is denoted with MAE in mass conservation of 10^{-2} L/s and MAE in energy conservation of 10^{-3} *m*.

The same border in the example of KL is predominantly convex, with the exception of the upper right corner. The parameter space holds a "sweet spot" with the fastest convergence near the divergence areas. Notably, the shape, size, and curvature of the area of the fastest convergence varies across WDSs. In Appendix D we include a heuristic procedure to identify the area of the fastest convergence without running Bayesian optimization.

For a better understanding of the algorithm and the influence of τ, we visualized the convergence dynamics on the example of BAK in terms of errors of mass and energy conservation. Based on the stability regions, we sampled the several pairs of timestep sizes by firstly varying τ_{down} and fixing τ_{up}. The visualization in Figs 4a and 4b shows that changes in the τ_{down} noticeably affect the dynamics of both gradient descents. Vice versa, we illustrated the change in the dynamics when varying only τ_{up} and fixing τ_{down} in Figs. 4c and 4d. The figures show that increasing the upper timestep size mainly impacts the dynamics of the upper descent, while mass conservation dynamics are barely affected. This suggests that the lower gradient descent plays a larger role in the convergence of the overall algorithm.

In the first 150 iterations, both conservation errors show relatively volatile behavior. Notably, all figures show an exponential decrease in conservation errors after stabilizing, indicating effective convergence. However, as the convergence progresses, a diminishing return on the accuracy becomes more visible. Achieving very small errors in mass and energy conservation may incur higher computational costs and vice versa. Thus, the proposed algorithm provides a flexible framework to control the trade-off between accuracy and speed.

4.1.3. Robustness of optimal parameters under changing demands

The previous experiments demonstrated the effect of the selected parameters on the convergence of the algorithm based on fixed and snapshot simulations. However, real applications often consider alternative demand patterns evaluated in parallel. The varying demand can affect the optimal set of parameters and the number of steps until convergence n. This section explores the stability of optimal parameters amid these changes. We selected the demand patterns from EPS of area C of L-Town as an example and illustrated the convergence dynamics with fixed selected parameters. By examining n one-by-one, we are able to study the robustness in detail. The optimal parameter set was identified based on the demand pattern at the beginning of hydraulic time $\mathbf{q}_0$. Consecutive $\mathbf{q}$ are compared with $\mathbf{q}_0$ with a cosine similarity:

$$COS\ \text{similarity}(\mathbf{q}, \mathbf{q}_0) = \frac{\mathbf{q} \cdot \mathbf{q}_0}{||\mathbf{q}||\ |\mathbf{q}_0|} \tag{22}$$

Despite minor deviations, primarily occurring during periods of low cosine similarity with the $\mathbf{q}_0$ (Fig. 5a), the n remains within the bounds of 23 iterations. Fig. 5b shows additional visualizations of stability regions that correspond to hydraulic timestamps with the largest deviation in convergence time n. The stability regions are visualized similarly to the setting in Section 4.1.2 within bounds $\tau_{down} = [0.25, 0.60]$ and $\tau_{up} = [0.005, 0.013]$, while the momentum coefficients are the same as identified at $\mathbf{q}_0$. The figures illustrate that the stability regions change shape with changing demands, along with the position of respective optimal pairs (τ_{down}, τ_{up}). The momentum seems to add further complexity to the shapes and border of the stability regions. The visualization of the stability regions without momentum can be found in Appendix E.2. These findings suggest, that the optimal parameters may depend on the demand patterns. One may consider this influence during the optimization procedure, for example, by employing several data points for finding balanced optimum. The data points can be selected based on the least cosine similarity. The number of recommended points depends on the application. For example, during the uncertainty estimation, the optimal points can be identified on a single simulation. If the task assumes large deviations in demand patterns, or the steady state is influenced by pipe friction, it is recommended to optimize based on a larger number of simulations. Based on the example in Fig. 5a, one may select simulations with the least similar demand patterns. The optimization can be likewise performed in parallel.

4.2. Evaluation of speed-up

Figs. 6a and 6b illustrate computational gains against the number of simultaneously evaluated simulations in a batch (M) and the total number of pipes in a batch (E_M), respectively. These experiments were carried out on a computational cluster node with 24 GB visual memory. Until a certain point number of pipes in a batch, the depicted relationship is largely linear and the clock time T for running thousands of parallel simulations remains constant.

Therefore the algorithm provides a linear speedup, however, this behavior changes as E_M reaches 10^5 (see Fig. 6a) for every case study. Beyond this threshold, the average time per additional simulation equals that of EPANET, indicating a diminishing return with an increasing number of E_M. The core limitation is attributed to the performance of sparse aggregation, as detailed in Yang et al. (2018). The break-even point between EPANET and the diffusion method is determined where the speed-up intersects the horizontal line of speed-up $S = 1$. For ZJ, L-TOWN, ASNET2, KL, and APULIA, our method achieves convergence at a similar or faster pace than EPANET. Benefits for the remaining cases

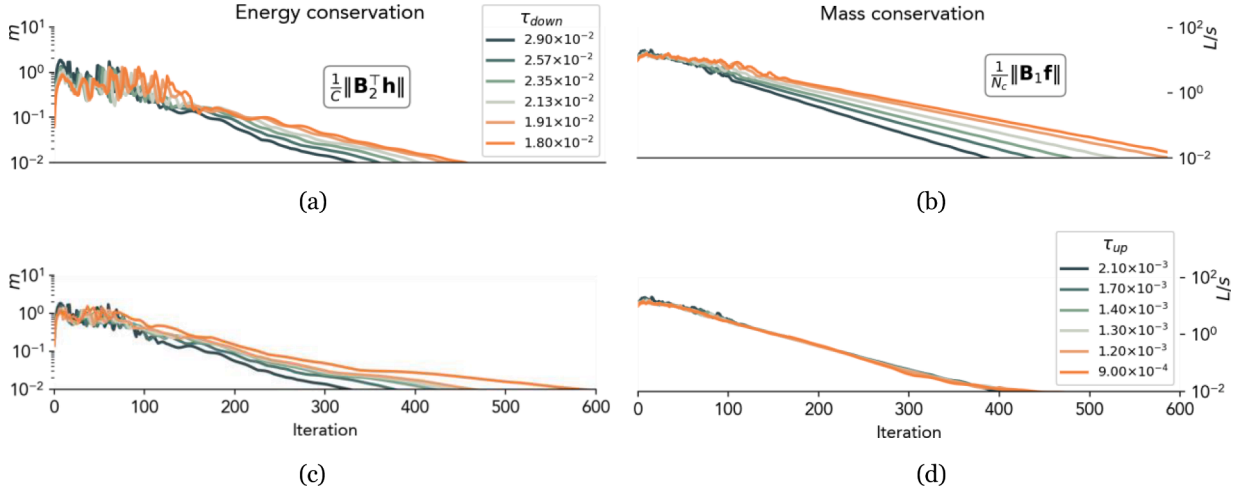


Fig. 4. Convergence dynamics at sampled timesteps without momentum on the example of BAK. Top figure shows dynamics at increasing τdown with fixed τup = $2.90 \cdot 10^{-2}$. Bottom figure shows dynamics at increasing τup with fixed τdown = $2.13 \cdot 10^{-3}$.

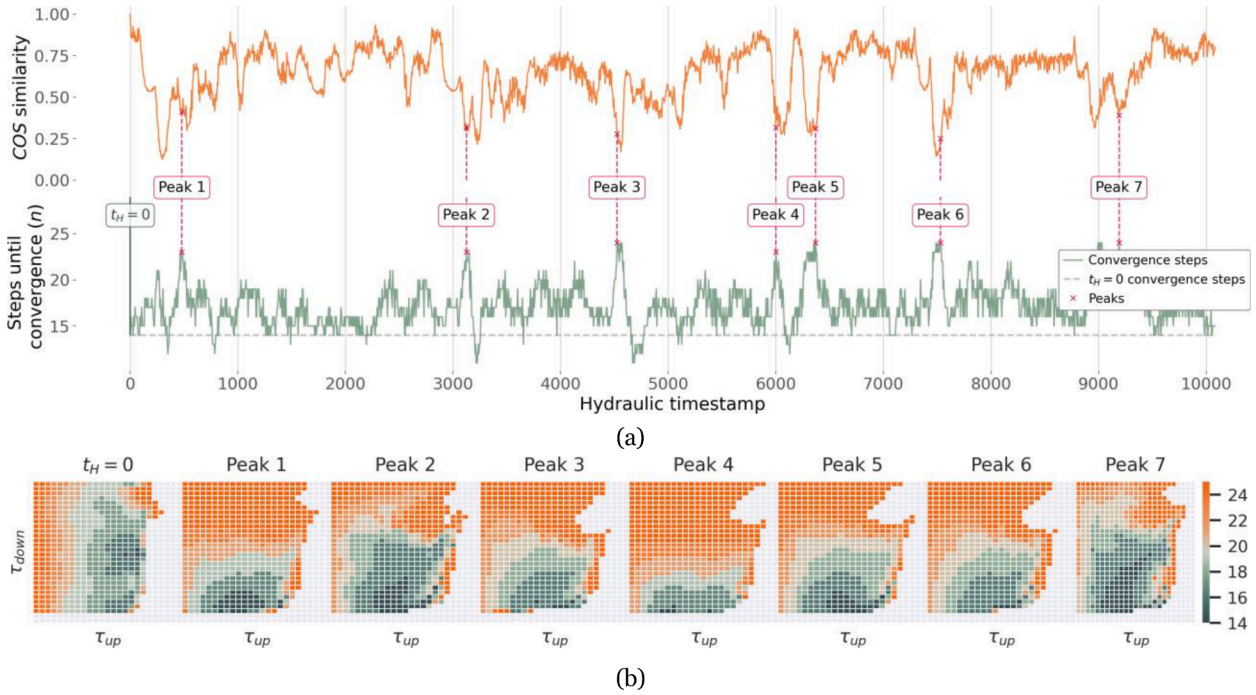


Fig. 5. Robustness of optimal parameters under changing demand patterns. Fig. 5a compares the number of steps until convergence (t) with the cosine similarity between $\mathbf{q}$ and $\mathbf{q}_0$. Figure 5b depicts stability regions in terms of n (capped at 25) and within bounds $\tau_{\text{down}} = [0.25, 0.60]$ and $\tau_{\text{up}} = [0.005, 0.013]$. The momentum values are equal to the ones identified at $\mathbf{q}_0$.

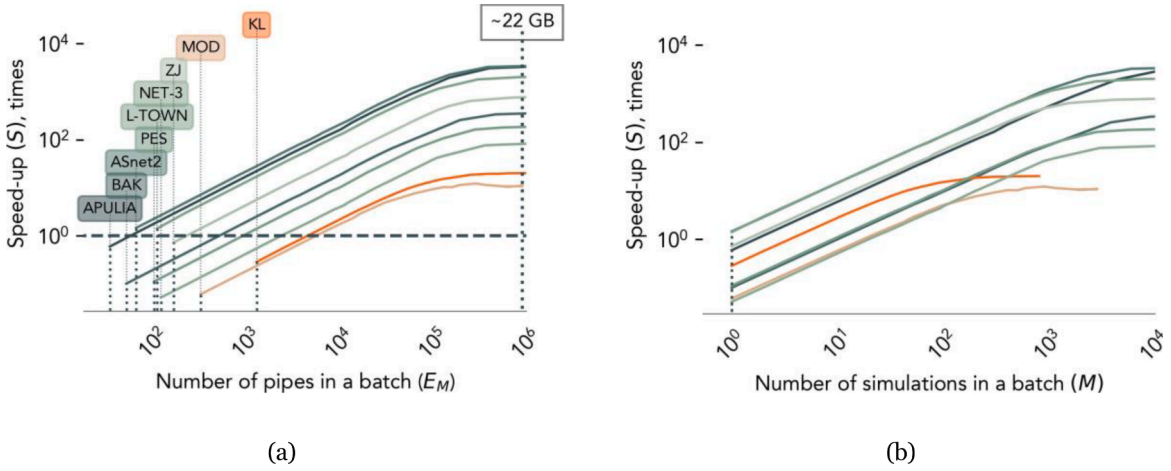


Fig. 6. Analysis of speed-ups with respect to batch size in terms of E_M (left) and M (right). The evaluation is performed on the computational cluster node with 24 GB GPU memory.

become evident in problems that need at least 10-20 parallel steady-state evaluations.

In the second set of findings, we visualize the speed-ups achievable on a typical PC with 2 GB of GPU memory for single and multiple reservoir networks. The relationship between pipe count and speed-up $S = 1$ is further analyzed in Fig. 7. We witness a strong correlation between pipe count and S in single-reservoir networks. Smaller networks exhibit the most significant gains, achieving speeds several thousand times faster. In contrast, the parallelization on multi-core CPUs tends to yield more substantial speedups as network size increases, at least in the case of artificial networks (Burger et al., 2016). This commonly occurs because fine-grained parallelization on CPUs amortizes the overhead of synchronizing numerous small work units more effectively when those units are part of a larger, more computationally intensive network. Unlike the works on CPU parallelization, our approach focuses on coarse-grained parallelization. The benefits of our approach are correlated with the number of simulations that can be evaluated simultaneously in the batch of simulations.

Notably, attaining a marginally higher accuracy of 0.999 in R^2 for flowrates takes at least twice as long as achieving an accuracy of 0.99. Reaching the same accuracy in terms of headlosses requires comparable time for most networks. However, in the case of KL, the accuracy of headlosses is limited to R^2 of 0.995, possibly due to the larger size of the network. Multiple-reservoir networks experience less parallelization benefit, as the reservoir outflows are not considered known and are not used as boundary conditions (see Section 2.2.1). It is also important to acknowledge that EPANET iterates using GGA until very tight tolerances in mass and energy errors are met, ensuring highly accurate solutions but increasing computation time. If one is willing to accept slightly larger errors (e.g., by halting at an R^2 of about 0.999 instead of an even

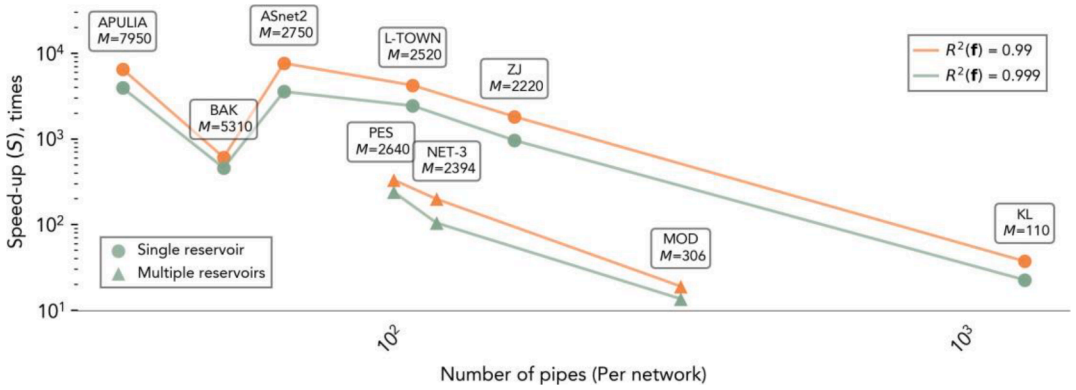


Fig. 7. Speed-up obtained with 2 GB of graphical memory.

more stringent value), this can reduce the number of GGA iterations needed. In our tests, such a reduction in required iterations generally resulted in about a twofold increase in speed for most networks. For the ZJ network, speedups of around 30% were observed. Nonetheless, edge-based diffusion still delivers substantial parallelization capabilities, resulting in a minimum tenfold acceleration of calculations.

4.3. Limitations

In comparison to GGA, our method presents a marginally lower accuracy, as GGA converges towards an energy- conservative solution while preserving mass conservation at each iteration. Thus, the GGA provides a more precise solution. Additionally, it must be noted that the speed-ups present a best-case scenario. The optimal parameters were selected based on the demands from *.inp* file. However, the model-based application may require state estimation with adjusted demands. As visible from the experiment on robustness, the change in demands can slightly affect the minimal n and, therefore, the convergence speed. Still, parallelization via diffusion allows the evaluation of thousands of simulations in parallel. Our algorithm presents a further potential for faster estimation by using a faster sparse aggregation frameworks, e.g. JAX (Bradbury et al., 2018).

In this work, the method did not include additional hydraulic components, e.g. tanks, valves and pumps. Since the flowrates are estimated at each hydraulic timestep, the representation of tanks can be included similarly to the implementation in EPANET. While flowrate is a key characteristic in determining the operating efficiency of the pumps, edge-based diffusion may then be more accurate in identifying the operational point of a pump. Mechanical energy, contributed into the system via the pump can be incorporated into the energy balance.. It is additionally possible to incorporate some of the flow-dependent valves e.g. check-valves or flow-control valves the same way as in EPANET. In our model, they will operate as a boundary condition, similar to the demand values. Implementing pressure-dependent valves (e.g. PRV, PSV, and others) can be done with additional virtual connections or intermediate pressure reconstruction.

The first option is to build additional virtual connectors between PRV or PSV and the node with a known head (e.g. reservoir), similar to how virtual reservoir connectors link 2 known heads. The pressure at the valves can be calculated based on the difference between the known head and the predicted $\mathbf{h}$. (Alvarruiz et al., 2015)

The second option assumes intermediate pressure reconstruction in the whole network (or at nodes of interest), by traversing a path of the shortest distance or minimal spanning tree (Todini and Rossman, 2013) with predicted $\mathbf{h}$. A similar strategy was employed in the work on surrogate models for WDSs (Ashraf et al., 2024). The intermediate calculation of pressures on every junction will allow for modeling pressure-driven analysis, pressure-dependent valves, and emitters.

5. Conclusion

The work posed the steady state estimation as a diffusion process on the edge space of a graph. Approximation of the diffusion process with an iterative scheme with momentum leads to both mass and energy-conservative steady state, presenting a novel approach to hydraulic solving. Parallelizable on GPU, the method drastically decreases the required time for evaluation, as was shown in a multiple case studies. Thousands of hydraulic simulations can be evaluated with nearly constant time. With the continuous development of graphical hardware and corresponding computational frameworks (e.g. JAX), we anticipate that our method will achieve further computational gains. The study additionally offers an optimization procedure for determining the optimal set of parameters that lead to the fastest convergence. The set is mostly defined by the topology and pipe parameterization, to a much lesser extent, by the demand values.

Current applications include evaluation of the model response to the uncertainty of demands, model-based leakage detection, sensor placement, and the analysis of sensitivity to demands. Ensuring the robustness of the optimal parameters to the changes in pipe parameterization will enable the majority of model-based design, optimization, and calibration problems.

Reformulating state estimation as a diffusion process presents several opportunities. Firstly, it allows studying the problem from the perspective of convex optimization and differential equations. As such, it opens the way for applying multi-step methods (e.g. Runge-Kutta) and implicit iterative methods to provide an additional speed-up. These iterative schemes are known to be more stable than the forward Euler scheme and require fewer iterations to reach convergence, benefiting cases like BAK. Moreover, a study on different initial conditions and loop identification methods is a promising direction for further improvement. Secondly, our framework can aid in studying the analogy between the size of WDS and the discretization of theoretical topological space that underlies the water networks and be used in skeletonization. A skeletonized version requires fewer steps for convergence because of the smaller network size. One may leverage the solution for a skeletonized version and link it to the coarser version as an initial condition to achieve further speed-ups.

Future avenues can address the mentioned issues with accuracy and study the stability regions and their robustness to the change in demands in more depth. Exploring the interplay between the lower and upper steps could be a promising direction for future research. Lastly, this work posed the steady state estimation with boundary conditions of demands. Real-life applications often assume pressure-driven dynamics. Formulating the boundary condition as a function of pressures (Wagner et al., 1988) could be a step towards modeling the pressure-driven dynamics.

CRediT authorship contribution statement

Bulat Kerimov: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Data curation, Conceptualization. **Maosheng Yang:** Writing – review & editing, Methodology, Conceptualization. **Riccardo Taormina:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Franz Tscheikner-Gratl:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

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

Data availability statement

The code and data are available in an online repository at the following link https://github.com/erytheis/edge-based-diffusion-for-wds.

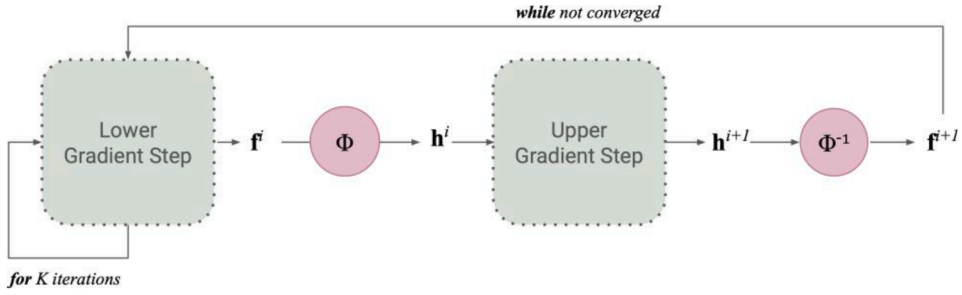
Acknowledgements

The authors acknowledge the financial support through the NTNU Green2050 Centre for Green Shift in the Built Environment and the TU Delft AI Labs programme. The authors would also like to thank Dr Elvin Isufi, and other fellow colleagues in AIdroLab and NTNU for their insightful comments.

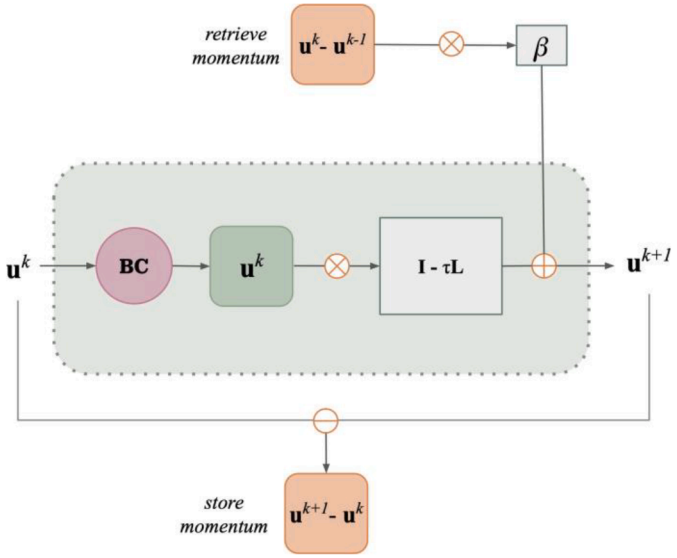
Appendix A. Schematic depiction of the iterative algorithm

This section provides a schematic overview of the iterative algorithm described in Section 2.3.3. Fig. 8a illustrates the main loop of the procedure, showing how the algorithm alternates between mass-conserving and energy- conserving steps.

The algorithm begins by initializing the flow rate vector $\mathbf{f}^{(0)}$ to zero. It then enters the main loop, which consists of two phases: K iterations of a lower gradient descent and an upper gradient descent. Fig. 8b provides a general depiction of the gradient step. In this figure, any vector (**f** or **h**) can replace the generic vector **u**. The associated matrices and parameters (such as the upper- or lower-Laplacians, τ_{up} or τ_{down}, and β_{up} or β_{down}) are substituted as needed, depending on whether a mass-conserving or energy-conserving step is being performed. Each gradient descent step involves a BC, multiplication with the Laplacian matrix, and addition of the momentum term. The momentum term is stored to be retrieved in the next iteration for that vector.



(a) A depiction of the main loop



(b) General depiction of the gradient step for some vector **u**, Laplacian matrix **L** and parameters τ and β.

Fig. 8. Schematic depiction of the main iterative algorithm

Appendix B. Parallelization with sparse aggregation

Sparsity is the key property of both lower- and upper-Laplacians. This means that most of the entries in those matrices are zeros, which allows for representing those matrices in compressed format (Buluç et al., 2009). This enables the creation of a very large graph with multiple connected components.

Specifically, when multiple simulations are considered, their Laplacian matrices are stacked as a block diagonal matrix. A combined matrix $\mathbf{L}_{\text{comb}}$ represents the edge connectivity of a large graph network with M connected components, where M is the batch size. Flowrates are concatenated in the first dimension, i.e.

$$\mathbf{L}^{comb} = \begin{bmatrix} \mathbf{L}^{(1)} & & \\ & \ddots & \\ & & \mathbf{L}^{(M)} \end{bmatrix} \mathbf{f}^{comb} = \begin{bmatrix} \mathbf{f}^{(1)} \\ \vdots \\ \mathbf{f}^{(M)} \end{bmatrix}$$

As a result, a large matrix **L**comb is sparse. Sparse aggregation techniques allow the construction of a very large combined matrix (Yang et al., 2018). In the context of simultaneous simulation, parallel multiplication of thousands of WDSs becomes possible and is used as a basis for edge-based diffusion.

B.1. Visualization of Sparse Matrix Representation

Sparse matrices can be stored using various formats, such as COO (Coordinate Format). This format is straight- forward and stores the row indices, column indices, and values of the non-zero entries in the matrix.

For example, consider the sparse matrix:

$$\mathbf{A} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 \end{bmatrix}$$

The COO representation of this matrix is as follows:

Row Index	Column Index	Value
0	0	10
0	6	12
1	4	7
3	3	5
5	2	3
6	5	4

This format requires storing only 18 entries (i.e. 8 values, their columns, and rows) instead of 49. In a block-diagonal structure, the number of zero entries drastically increases.

B.2. Sparse matrix multiplication

Sparse matrix multiplication is a computationally intensive operation but can be significantly accelerated using parallelization techniques, particularly on GPUs. In GPUs, matrix operations are divided into multiple threads, each responsible for computing a subset of the output matrix. For sparse matrices, each thread can independently compute a single row of the output vector by iterating over the non-zero elements of that row.

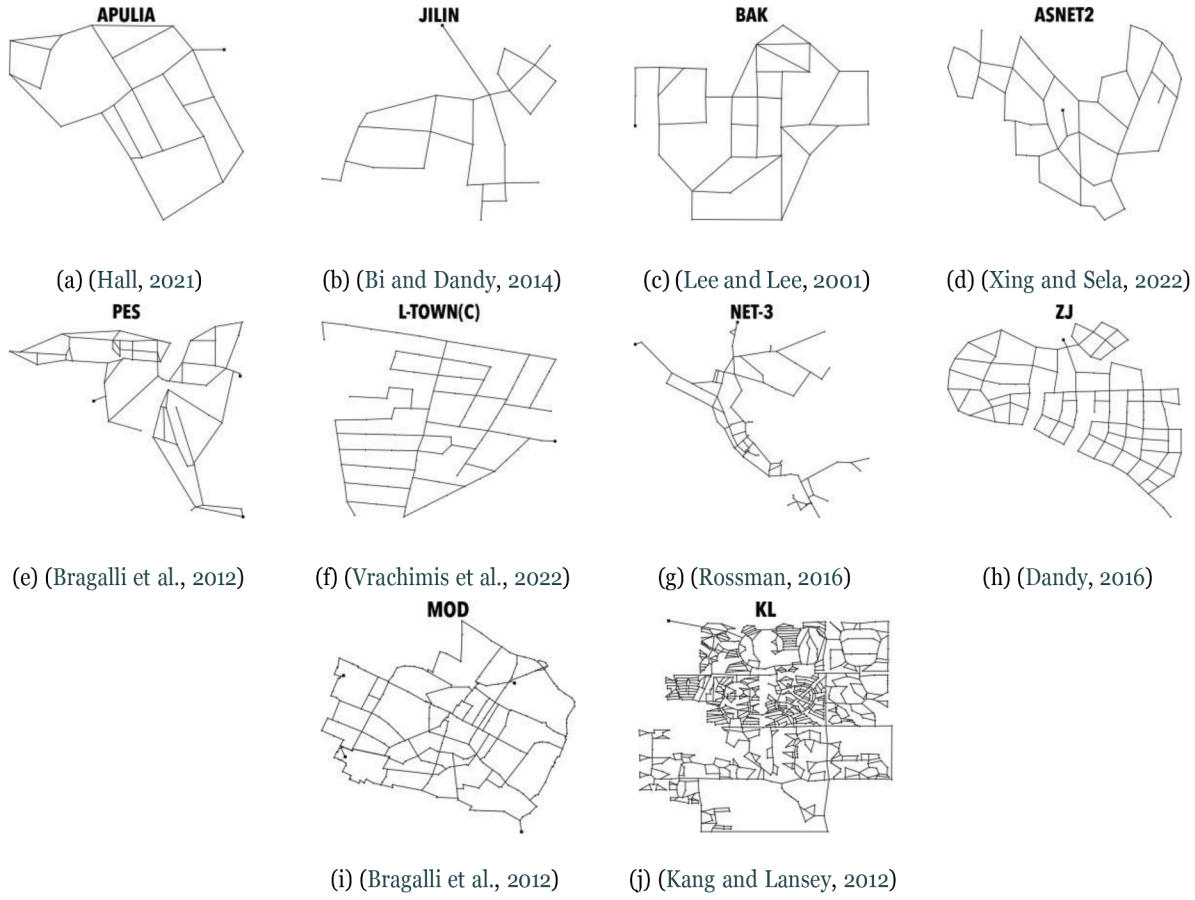
Consider a sparse matrix **A** stored in CSR format:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 \\ 4 & 0 & 0 & 5 \end{bmatrix} \mathbf{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \\ \boldsymbol{x}_4 \end{bmatrix}$$

The computation of $\mathbf{y} = \mathbf{A}\ \mathbf{x}$ proceeds as follows:

- Thread 0 computes $\mathbf{y}\ [0] = x_1 + 2x_4$.
- Thread 1 computes $\mathbf{y}\ [1] = 3x_3$
- Thread 2 computes $\boldsymbol{y}\ [2] = 4x_1 + 5x_4$.

and so on. Each thread operates independently, storing the result in the corresponding entry of **y**.



(a) (Hall, 2021) (b) (Bi and Dandy, 2014) (c) (Lee and Lee, 2001) (d) (Xing and Sela, 2022)

(e) (Bragalli et al., 2012) (f) (Vrachimis et al., 2022) (g) (Rossman, 2016) (h) (Dandy, 2016)

(i) (Bragalli et al., 2012) (j) (Kang and Lansey, 2012)

Fig. 9. Water distribution networks used in the experiments.

Appendix C Layout of the case studies

The case studies are visualized in Fig. 9.

Appendix D. Heuristic for identification of optimal parameters

The common pattern in the stability regions is the region with the fastest convergence which is usually located near the border between the divergence and convergence. As a rule of thumb, the engineers could employ the following procedure: Select 0.8 for both β, and iteratively increase τ_{down} with a large step, e.g. 0.1, until the algorithm starts to diverge. Once the lower border is identified, iteratively increase τ_{up} with some selected small step size, until the algorithm diverges. Decrease τ_{down} until the algorithm converges again, now with a smaller stop size, e.g. 0.3. Next increase τ_{up} again with a smaller step size, until the algorithm diverges. An example of the search is presented Fig. 10.

Appendix E Additional experiments

E.1 Influence of momenta

In this study, we conducted an ablation by removing the momentum component from our model. We compared the number of steps required for convergence n without momentum to those observed in Experiment 3.2.1, where momentum was present. This comparison helps us understand the significance of the momentum component in accelerating convergence. Similarly to the baseline experiment, we identified the optimal parameters timestep sizes, while momentum coefficients were set to zero. The number of lower iterations K is selected according to Table 2.

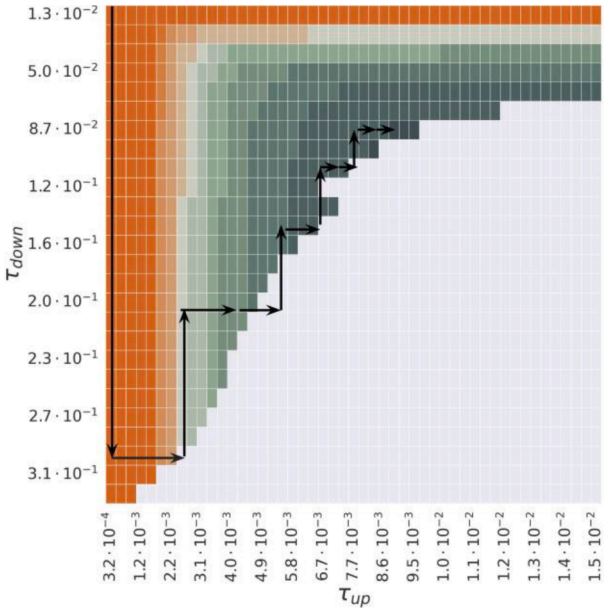


Fig. 10. Example of the heuristic search of optima

The comparison is visualized in Fig. 11. For each case, the inclusion of momentum resulted in significantly faster convergence. Specifically, for the PES, ZJ, and KL models, the presence of momentum led to roughly five times fewer steps needed for convergence.

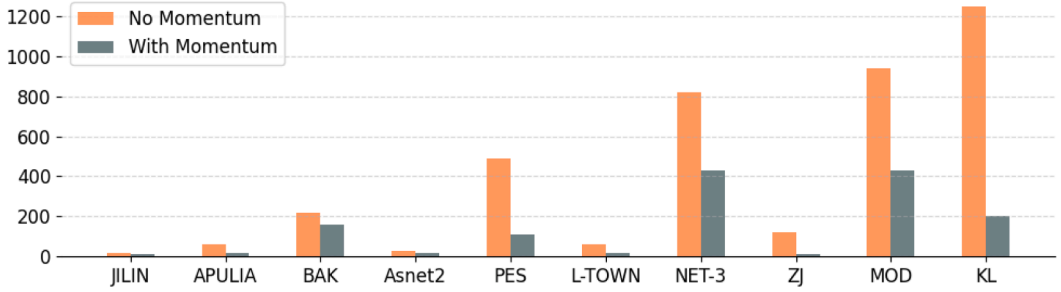


Fig. 11. Comparison of identified minimum n with and without momenta

E.2 Influence of number of lower gradients steps K

This section presents analysis of the influence of Kon stability regions in the example of APULIA, PES, and L-TOWN networks. The momentums are selected as 0.

The Fig. 12 displays the results. As K increases, the border of divergence seems to gradually move towards smaller values of τ_{down}. This behavior is expected, as the algorithm becomes more sensitive to the lower timestep size. We additionally observe that the area of fastest convergence is larger for K greater than 1. It becomes especially visible in the example of L-TOWN in Fig. 12c. For $k = 9$, however, the area seems to decrease in size. Furthermore, larger K will induce additional computational costs. Thus, selected K must take into consideration both the number of iterations until convergence and the final clocktime of the algorithm. In our experiments, the optimal K lies in the range between 3 and 5 iterations.

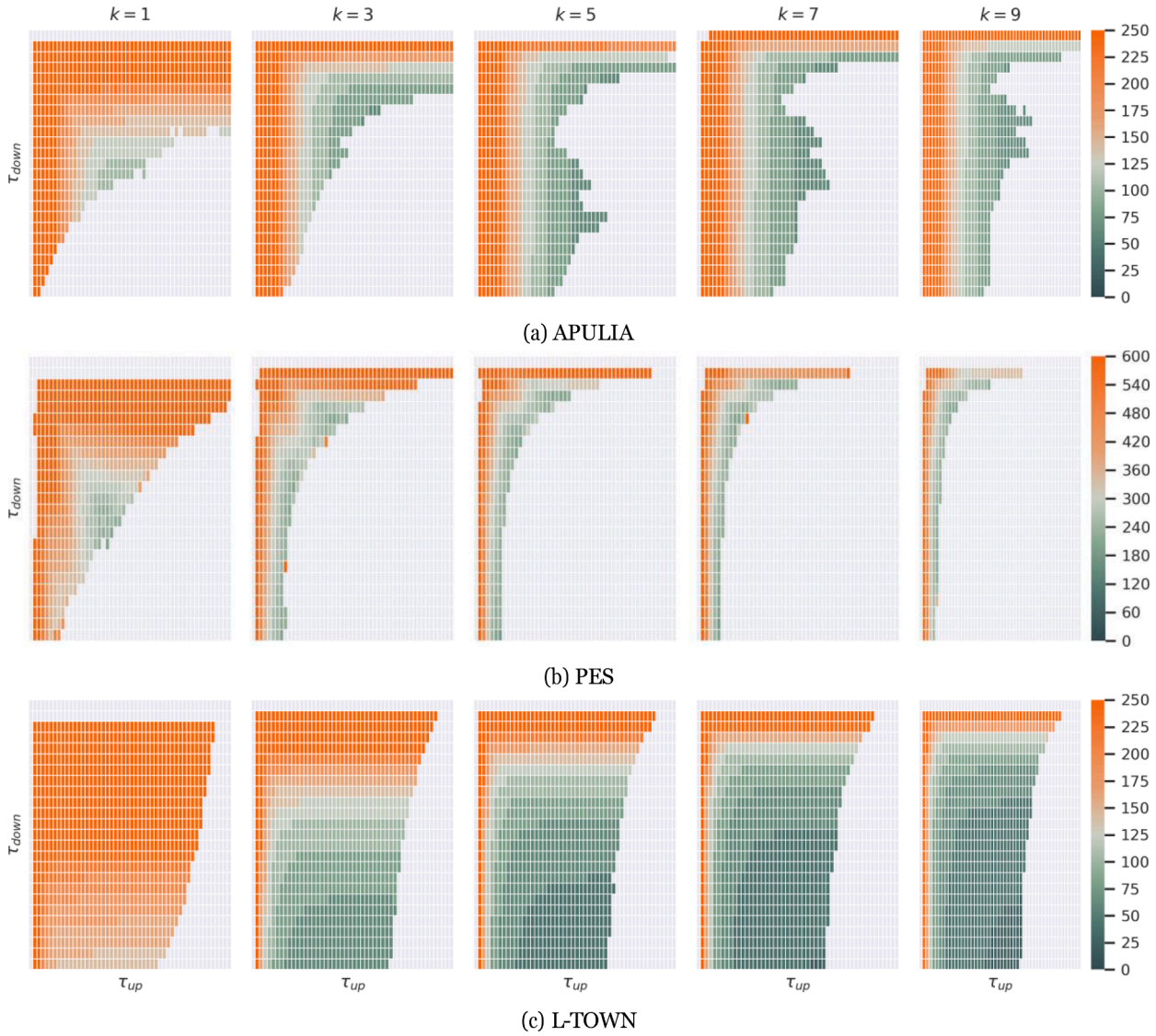
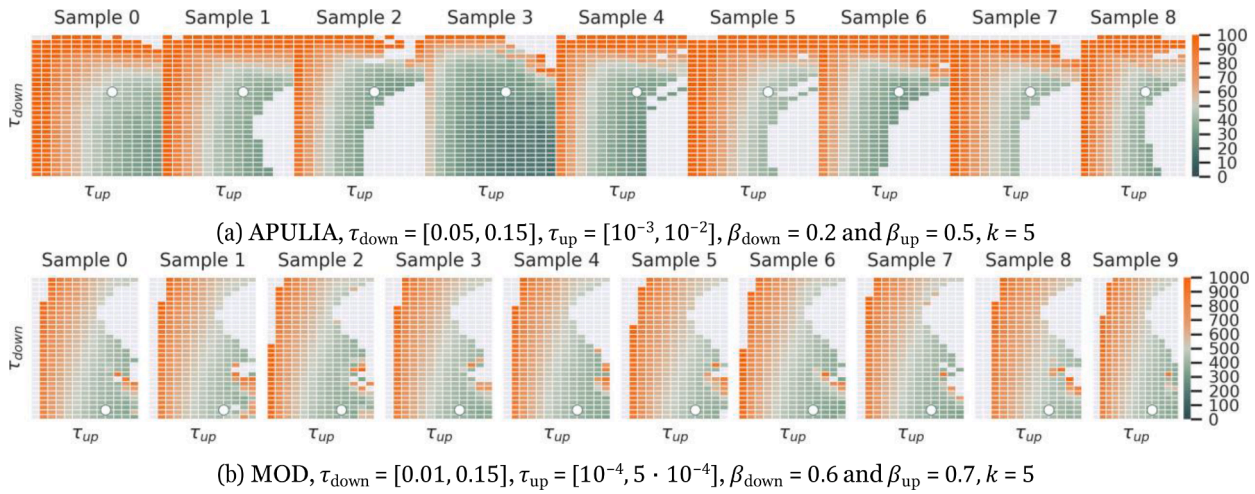


Fig. 12. Stability regions with the increasing number of steps of lower gradient descent K.

E.3 Robustness of optimal parameters under changing demands

This section includes additional experiments on visualization of the robustness of optimal parameters to changing demands. We selected 2 case studies: MOD as an example of a large network with multiple reservoirs, and APULIA as an example of a network with high κ. A higher κ usually leads to a higher influence of lower and upper gradient steps on the convergence of each other. It is visible through Hazen-Williams relationship in Equation 5 and Lines 12 of the main algorithm.

As the original *.inp* file contains a single set of demand patterns, we generated 10 additional simulations by uniformly sampling demands on every node as $\mathbf{q} \sim Uniform(0,\ 1)\ L/s$. The stability regions are visualized similarly to the ones in Fig. 5b. Fig. 13 displays that the regions of highest convergence remain relatively stable to the change in demands. Thus, an optimal parameter can be selected based on multiple simulations at once. For some simulations, the minimum n is higher than the value achieved based on the demands from the original *.inp,* see Table 2. This confirms the finding that the minimum n depends on the demands.



(a) APULIA, $\tau_{\text{down}} = [0.05, 0.15]$, $\tau_{\text{up}} = [10^{-3}, 10^{-2}]$, $\beta_{\text{down}} = 0.2$ and $\beta_{\text{up}} = 0.5$, $k = 5$

(b) MOD, $\tau_{\text{down}} = [0.01, 0.15]$, $\tau_{\text{up}} = [10^{-4}, 5 \cdot 10^{-4}]$, $\beta_{\text{down}} = 0.6$ and $\beta_{\text{up}} = 0.7$, $k = 5$

Fig. 13. Robustness of optimal parameters to sampled demand patterns. The figure depicts stability regions in terms of n. The white dots are an optima defined based on these simulations.

References

Aktas, M.E., Akbas, E., 2021. Graph classification via heat diffusion on simplicial complexes. IEEE Access 9, 12291–12300.

Alvarruiz, F., Martínez-Alzamora, F., Vidal, A.M., 2015. Improving the efficiency of the loop method for the simulation of water distribution systems. Journal of Water Resources Planning and Management 141, 04015019. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000539. URL:

Ashraf, I., Strotherm, J., Hermes, L., Hammer, B., 2024. Physics-informed graph neural networks for water distribution systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, 38, pp. 21905–21913. https://doi.org/10.1609/aaai.v38i20.30192. URL: https://ojs.aaai.org/index.php/AAAI/article/view/30192.

Bai, X., Hancock, E.R., 2004. Heat kernels, manifolds and graph embedding. In: Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, August 18-20, 2004. Proceedings. Springer, pp. 198–206.

Bello, O., Abu-Mahfouz, A.M., Hamam, Y., Page, P.R., Adedeji, K.B., Piller, O., 2019. Solving management problems in water distribution networks: A survey of approaches and mathematical models. Water 11. https://doi.org/10.3390/w11030562. URL: https://www.mdpi.com/2073-4441/11/3/562.

Bi, W., Dandy, G., 2014. Optimization of water distribution systems using online retrained metamodels. Journal of Water Resources Planning and Management 140, 04014032. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000419.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q., 2018. JAX: composable transformations of Python+NumPy programs. URL: http://github.com/google/jax.

Bragalli, C., D'Ambrosio, C., Lee, J., Lodi, A., Toth, P., 2012. On the optimal design of water distribution networks: A practical MINLP approach. Optimization and Engineering 13, 219–246. https://doi.org/10.1007/s11081-011-9141-7.

Buluç, A., Fineman, J.T., Frigo, M., Gilbert, J.R., Leiserson, C.E., 2009. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In: Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures, pp. 233–244.

Burger, G., Sitzenfrei, R., Kleidorfer, M., Rauch, W., 2016. Quest for a new solver for epanet 2. Journal of Water Resources Planning and Management 142, 04015065. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000596. URL:

Burrows, R., Crowder, G., Zhang, J., 2000. Utilisation of network modelling in the operational management of water distribution systems. Urban Water 2, 83–95. https://doi.org/10.1016/S1462-0758(00)00046-7. URL: https://www.sciencedirect.com/science/article/pii/S1462075800000467. developments in water distribution systems.

Calder, J., 2020. The calculus of variations. URL: http://www-users.math.umn.edu/~jwcalder/8385F19/CalculusOfVariations.pdf.

Calder, J., Yezzi, A., 2019. Pde acceleration: a convergence rate analysis and applications to obstacle problems. Res Math Sci 6, 35. https://doi.org/10.1007/s40687-019-0197-x. URL:

Crous, P.A., van Zyl, J.E., Roodt, Y., 2012. The potential of graphical processing units to solve hydraulic network equations. Journal of Hydroinformatics 14, 603–612.

Dandy, G., 2016. 06 Zhi Jiang. International Systems 6. URL: https://uknowledge.uky.edu/wdst_international/.

DeVille, L., 2021. Consensus on simplicial complexes: Results on stability and synchronization. Chaos: An Interdisciplinary Journal of Nonlinear Science 31, 023137. https://doi.org/10.1063/5.0037433. URL:

Dijkstra, E.W., 2022. A note on two problems in connexion with graphs. Edsger Wybe Dijkstra: His Life, Work, and Legacy, pp. 287–290.

Dragan, A., Savic, Z.S.K., Jonkergouw, P.M., 2009. Quo vadis water distribution model calibration? Urban Wa-ter Journal 6, 3–22. https://doi.org/10.1080/15730620802613380. URL:

Gambuzza, L.V., Di Patti, F., Gallo, L., Lepri, S., Romance, M., Criado, R., Frasca, M., Latora, V., Boccaletti, S., 2021. Stability of synchronization in simplicial complexes. Nature communications 12, 1255.

Garzón, A., Kapelan, Z., Langeveld, J., Taormina, R., 2022. Machine learning-based surrogate modeling for urban water networks: Review and future research directions. Water Resources Research 58. https://doi.org/10.1029/2021WR031808.

Guidolin, M., Kapelan, Z., Savić, D., 2012. Using high performance techniques to accelerate demand-driven hydraulic solvers. Journal of Hydroinformatics 15, 38–54. https://doi.org/10.2166/hydro.2012.198. URL:arXiv: https://iwaponline.com/jh/article-pdf/15/1/38/386915/38.pdf.

Hagberg, A., Swart, P., S Chult, D., 2008. Exploring network structure, dynamics, and function using NetworkX. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Hall, A., 2021. 01 Apulia. URL: https://uknowledge.uky.edu/wdst_international/1.internationalSystems, 1.

Kang, D., Lansey, K., 2012. Revisiting optimal water-distribution system design: Issues and a heuristic hierarchical approach. Journal of Water Resources Planning and Management 138, 208–217. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000165.

Kerimov, B., Bentivoglio, R., Garzón, A., Isufi, E., Tscheikner-Gratl, F., Steffelbauer, D.B., Taormina, R., 2023. Assessing the performances and transferability of graph neural network metamodels for water distribution systems. Journal of Hydroinformatics. https://doi.org/10.2166/hydro.2023.031.

Kerimov, B., Taormina, R., Tscheikner-Gratl, F., 2024. Towards transferable metamodels for water distribution systems with edge-based graph neural networks. Water Research 261, 121933. https://doi.org/10.1016/j.watres.2024.121933. URL: https://www.sciencedirect.com/science/article/pii/S0043135424008340.

Klise, K.A., Bynum, M., Moriarty, D., Murray, R., 2017. A software framework for assessing the resilience of drinking water systems to disasters with an example earthquake case study. Environmental Modelling Software 95, 420–431. https://doi.org/10.1016/j.envsoft.2017.06.022. URL: https://www.sciencedirect.com/science/article/pii/S1364815216309501.

Lee, S.C., Lee, S.I., 2001. Genetic algorithms for optimal augmentation of water distribution networks. Journal of Korea Water Resources Association 34, 567–575.

Levie, R., Isufi, E., Kutyniok, G., 2019. On the transferability of spectral graph filters. In: 2019 13th International conference on Sampling Theory and Applications (SampTA), 1–5, 59413913. URL: https://api.semanticscholar.org/CorpusID.

Lim, L.H., 2020. Hodge laplacians on graphs. SIAM Review 62, 685–715. https://doi.org/10.1137/18M1223101. URL:

Mair, M., Sitzenfrei, R., Kleidorfer, M., Rauch, W., 2014. Performance improvement with parallel numerical model simulations in the field of urban water management. Journal of Hydroinformatics 16, 477–486. https://doi.org/10.2166/hydro.2013.287.

Mala-Jetmarova, H., Sultanova, N., Savic, D., 2017. Lost in optimisation of water distribution systems? a literature review of system operation. En- vironmental Modelling Software 93, 209–254. https://doi.org/10.1016/j.envsoft.2017.02.009. URL: https://www.sciencedirect.com/science/article/pii/S1364815216307769.

Nguyen, V., 2019. Bayesian optimization for accelerating hyper-parameter tuning. In: Proceedings - IEEE 2nd International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2019, pp. 302–305. https://doi.org/10.1109/AIKE.2019.00060.

Pérez, R., Sanz, G., Cugueró, M.À., Blesa, J., Cugueró, J., 2015. Parameter uncertainty modelling in water distribution network models. Procedia Engineering 119, 583–592.

Polyak, B., 1964. Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics 4, 1–17. https://doi.org/10.1016/0041-5553(64)90137-5. URL: https://www.sciencedirect.com/science/article/pii/0041555364901375.

Porta, F., Cornelio, A., Ruggiero, V., 2017. Runge–kutta-like scaling techniques for first-order methods in convex optimization. Applied Numerical Mathematics 116, 256–272. https://doi.org/10.1016/j.apnum.2016.08.011. URL: https://www.sciencedirect.com/science/article/pii/S0168927416301611. newTrends in Numerical Analysis: Theory, Methods, Algorithms and Applications (NETNA 2015).

Qian, N., 1999. On the momentum term in gradient descent learning algorithms. Neural networks 12, 145–151.

Rossman, L.A., 2016. 06 EPANET Net 3. Software Manual Examples. https://uknowledge.uky.edu/wdst_manuals/2.

Schaub, M.T., Benson, A.R., Horn, P., Lippner, G., Jadbabaie, A., 2020. Random walks on simplicial complexes and the normalized hodge 1- laplacian. SIAM Review 62, 353–391. https://doi.org/10.1137/18m1201019.

Shamir, U., Salomons, E., 2008. Optimal real-time operation of urban water distribution systems using reduced models. Journal of Water Resources Planning and Management 134, 181–185. https://doi.org/10.1061/(ASCE)0733-9496(2008)134:2(181).

Shi, X., Zheng, Z., Zhou, Y., Jin, H., He, L., Liu, B., Hua, Q.S., 2018. Graph processing on gpus: A survey. ACM Computing Surveys (CSUR) 50, 1–35.

Solà Roca, A., 2023. Gganet: Algorithm unrolling for water distribution networks metamodelling.

Steffelbauer, D., Fuchs-Hanusch, D., 2016. Efficient sensor placement for leak localization considering uncertainties. Water Resources Management 30, 1–17. https://doi.org/10.1007/s11269-016-1504-6.

Steffelbauer, D., Neumayer, M., Günther, M., Fuchs-Hanusch, D., 2014. Sensor placement and leakage localization considering demand uncer- tainties. Procedia Engineering 89, 1160–1167. https://doi.org/10.1016/j.proeng.2014.11.242. URL: https://www.sciencedirect.com/science/article/pii/S1877705814023571, 16th Water Distribution System Analysis ConferenceWDSA2014.

Suvizi, A., Farghadan, A., Saheb Zamani, M., 2023. A parallel computing architecture based on cellular automata for hydraulic analysis of water distribution networks. Journal of Parallel and Distributed Computing 178, 11–28. https://doi.org/10.1016/j.jpdc.2023.03.009. URL: https://www.sciencedirect.com/science/article/pii/S0743731523000527.

Todini, E., Pilati, S., 1988. A gradient algorithm for the analysis of pipe networks. Computer Applications in Water Supply 1, 1–20.

Todini, E., Rossman, L.A., 2013. Unified framework for deriving simultaneous equation algorithms for water distribution networks. Journal of Hydraulic Engineering 139, 511–526. https://doi.org/10.1061/(ASCE)HY.1943-7900.0000703.

Vrachimis, S.G., Eliades, D.G., Taormina, R., Kapelan, Z., Ostfeld, A., Liu, S., Kyriakou, M., Pavlou, P., Qiu, M., Polycarpou, M.M., 2022. Battle of the leakage detection and isolation methods. Journal of Water Resources Planning and Management 148. https://doi.org/10.1061/(ASCE)WR.1943-5452. 0001601.

Wagner, J.M., Shamir, U., Marks, D.H., 1988. Water distribution reliability: Analytical methods. Journal of Water Resources Planning and Management 114, 276–294. https://doi.org/10.1061/(ASCE)0733-9496(1988)114:3(276). URL:cited by: 363; All Open Access, Green Open Access.

Wang, Q., Guidolin, M., Savic, D., Kapelan, Z., 2015. Two-objective design of benchmark problems of a water distribution system via moeas: Towards the best-known approximation of the true pareto front. Journal of Water Resources Planning and Management 141, 04014060.

Xing, L., Sela, L., 2022. Graph neural networks for state estimation in water distribution systems: Application of supervised and semisupervised learning. Journal of Water Resources Planning and Management 148. https://doi.org/10.1061/(ASCE)WR.1943-5452.0001550.

Yan, D., Wu, T., Liu, Y., Gao, Y., 2017. An efficient sparse-dense matrix multiplication on a multicore system. In: 2017 IEEE 17th International Conference on Communication Technology (ICCT), pp. 1880–1883. https://doi.org/10.1109/ICCT.2017.8359956.

Yang, C., Buluç, A., Owens, J.D., 2018. Design principles for sparse matrix multiplication on the gpu. In: Euro-Par 2018: Parallel Processing: 24th International Conference on Parallel and Distributed Computing, Turin, Italy, August 27 - 31, 2018, Proceedings. Springer-Verlag, Berlin, Heidelberg, pp. 672–687. https://doi.org/10.1007/978-3-319-96983-1_48. URL:

Yang, M., Isufi, E., Leus, G., 2022. Simplicial convolutional neural networks. In: Proceedings of the ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, United States, pp. 8847–8851. https://doi.org/10.1109/ICASSP43922.2022. 9746017.

Zhou, X., Liu, S., Xu, W., Xin, K., Wu, Y., Meng, F., 2022. Bridging hydraulics and graph signal processing: A new perspective to estimate water distribution network pressures. Water Research 217, 118416. https://doi.org/10.1016/j.watres.2022.118416. URL: https://www.sciencedirect.com/science/article/pii/S0043135422003724.

Ziegler, C., Skardal, P.S., Dutta, H., Taylor, D., 2022. Balanced hodge laplacians optimize consensus dynamics over simplicial complexes. Chaos: An Interdisciplinary Journal of Nonlinear Science 32. https://doi.org/10.1063/5.0080370. URL: