GGANet

# Algorithm Unrolling for Water Distribution Networks Metamodelling

Albert Solà Roca


TU Delft

# GGANet

## Algorithm Unrolling for Water Distribution Networks Metamodelling

by


## Albert Solà Roca


to obtain the degree of Master in Computer Science, Data Science and Technology track,
at the Delft University of Technology,
to be defended publicly on November $20^{th}$ 2023 at 16:00.


Student number: 5458676



Project duration: November 23, 2022 – November 20, 2023


*Thesis committee:*

| | |
|---|---|
| Chair and Advisor: Dr. Elvin Isufi, | Faculty EEMCS, TU Delft |
| Advisor: Dr. Riccardo Taormina, | Faculty CEG, TU Delft |
| Committe member: Dr. Stjepan Picek, | Faculty EEMCS, TU Delft |


An electronic version of this thesis is available at http://repository.tudelft.nl/.


TU Delft

# Acknowledgements

I would like to express my heartfelt gratitude to the individuals who have been instrumental in the successful completion of my Master's thesis. This research journey has been a challenging but immensely rewarding experience, and I could not have achieved it without the support, guidance, and inspiration of those who helped me throughout this process.

First and foremost, I am deeply grateful to my supervisors Elvin Isufi and Riccardo Taormina. Both of you have greatly helped me in both my academic and personal growth. Your trust, support, and feedback have been crucial to the completion of this thesis. I am grateful for the hours you dedicated to our discussions, your insightful feedback, and your continuous motivation. Finally, I want to once again express my heartfelt appreciation for providing me with the opportunity to present my thesis at CCWI, it was an incredible experience. Thanks as well to Dr. Stjepan Picek for taking the time to read, evaluate, and attend the defence of this thesis.

Next, I want to thank Alexander Garzón for his weekly support throughout the thesis. All the Wednesday afternoon meetings that clarified all the questions I had, from small to large, and all the feedback and structure you provided were extremely helpful. You also made the week at CCWI a lot of fun and helped me meet fantastic colleagues.

My sincere appreciation goes out to my family and friends for their unending support too. Your encouragement and belief in me helped me persevere during the demanding moments of thesis writing.

Last but not least, to my fiancée Maria, who has made the past two years in the Netherlands an incredible journey and who supported me every step of the way. The walks, the coffees, and the ramens have powered both of us through our theses, but being next to you has made every minute of work extra special.

*Albert Solà Roca*
*Delft, November 2023*

# Abstract

Water distribution networks (WDNs) provide drinking water to urban and rural consumers through a network of pipes that transport water from reservoirs to junctions. Water utilities rely on tools such as EPANET to simulate and analyse the performance of water distribution networks (WDNs). EPANET solves the flow continuity and headloss equations through pressurised piped networks under steady-state conditions. The EPANET solver applies the iterative Global Gradient Algorithm (GGA) of until convergence to determine unknown flows and pressures at the same time. Despite the widespread success of GGA, the speed of the algorithm may hinder several applications that require many simulations, especially for large networks.

To overcome these issues, researchers often resort to surrogate models, also known as metamodels, to significantly reduce the simulating times while approximating the behaviour of hydrodynamic models. Machine learning methods are increasingly being used as surrogate models for WDN analysis. Among these, the multi-layer perceptron (MLP) is the preferred metamodeling choice. MLPs provide a fast alternative to hydrodynamic models, but may lack the desired level of accuracy for complex case studies and applications. This is due to the lack of domain knowledge. Domain knowledge in WDNs can be divided into the topology of the network and the physical equations that govern the system. Recently, researchers have expanded MLPs to include domain knowledge in the form of the topology of the system. These new machine learning-based surrogates are called GNNs. However, GNNs present the problem that they lack speed compared to MLPs and still do not include information on the physical equations of the system.

For this reason, we propose applying algorithm unrolling to the GGA to include information on the physical equations that govern WDNs. Algorithm unrolling (AU) is a promising technique within the field of model-based deep learning that provides an alternative to traditional data-driven methods for approximating the output of iterative algorithms. This approach involves deconstructing the iterative algorithm used to solve the optimisation problem of interest into blocks that can be represented as layers in a neural network. Our architecture identifies two steps in the GGA and designs layers of a deep neural network following the steps of the GGA. In addition, the architecture includes information of the system features, those that remain constant throughout the steps of the GGA. We do so by adding residual connections to the variables. We evaluate the proposed metamodel by predicting the heads of five WDNs with varying characteristics and present an ablation study to understand the effect of the different components of our architecture. Our results show that our metamodel improved the accuracies with respect to MLPs and GNNs while being up to 2000 times faster than EPANET. We believe that the increase in accuracy with respect to the state-of-the-art baselines and the increase in speed with respect to EPANET justify the use of this metamodel for applications in the field of WDNs that rely on many simulations of EPANET.

# Contents

# 1

# Introduction

Water distribution networks (WDNs) provide drinking water to urban and rural consumers through a network of pipes that transport water from reservoirs to junctions. Water utilities rely on hydrodynamic models such as EPANET [1] to analyse, control [2] and optimise [3] these WDNs. EPANET allows water utilities to calculate the physical and chemical components of the state of a WDN. The physical state of a WDN consists of the nodal pressure and the link flows. EPANET computes the flow rates and pressures at all the pipes and junctions of a given WDN by solving the underlying equations of mass and energy conservation. These are calculated using the Global Gradient Algorithm (GGA). The GGA makes use of the Newton-Raphson algorithm to find solutions to the system of equations derived from the equations of mass and energy conservation. This results in an iterative algorithm that converges to the mathematical solution of the heads and flows.

However, the GGA presents the problem that this algorithm is computationally demanding because it iteratively updates the heads and flows of the network until convergence. This iterative process includes updating several large matrices and calculating their inverses at each step, which is computationally expensive. This is especially important in applications where many simulations must be run [4], [5]. To solve this, water modelers have recently resorted to surrogate models, also known as metamodels, to substitute computationally costly models [6], such as EPANET.

Surrogate models are simplified approximations of more complex models [7] that trade accuracy of the model for speed. The simulation of complex models can take from minutes to days, and some applications need to run them hundreds or thousands of times [8]. Surrogate models have been applied in fields such as manufacturing [9], calibration of environmental models [10], and water distribution networks [6], [11]. In the field of water distribution networks, surrogate models have been applied to network design and optimisation [12], risk assessment [13], flood prediction [14] or state estimation [15]. Other surrogate models for EPANET simulations that have been proposed [16]–[18] involve reducing the water network model to run the GGA algorithm faster. Although they are faster, this compromise reduces their fidelity [6], i.e., level of detail. More recent GGA surrogates involve using machine learning to predict the state of the network [15], [19], [20]. The most widely used machine learning tool to predict the state of a water distribution network is the multilayer perceptron (MLP). MLPs are trained in pairs of inputs and outputs of data but without considering any domain knowledge. Consequently, they require large amounts of data to be trained up to practical accuracy, which could not be easily available for WDNs.

Domain knowledge in WDN state estimation can be divided into the topology of the network and the physical equations that govern the system. With regards to the topology, water distribution networks can be represented as graphs. For a long time, graph-based techniques have been successfully applied to WDN [21]–[24], but with the rapid increase of AI techniques, new ML surrogates, such as graph neural networks, have been used to predict the state of the network [20], [25]. Graph neural networks (GNNs) are built upon MLPs by including the topological information of the system. Although GNNs have been shown to improve the accuracy of ANNs [20], they present the problem that they are significantly slower than MLPs. In addition, both GNNs and MLPs share the problem that neither includes information on the physical equations that govern the system.

Model-based neural networks incorporate domain knowledge in the form of an established model-based algorithm that is suitable for the problem at hand, combined with the ability to learn from data through deep learning [26]. These architectures have been used in domains such as graph signal processing [27],

symbol detection [28], and state estimation in the domain of power systems [29]. One technique within the domain of model-based neural networks is algorithm unrolling. Algorithm unrolling (AU) includes domain knowledge by converting an iterative algorithm into layers of a neural network [30]. Algorithm unrolling has been successfully applied in many applications where optimisation problems are solved, such as power system state estimation [29], compressive sensing [31], or image deblurring [32].

Therefore, in this thesis, we present a WDN steady-state estimation metamodel that includes information on the physical information of the system by applying algorithm unrolling to the GGA. Thus, we aim to answer the question:

***RQ**: How can we develop a deep neural network that includes the physical information of the WDN?*

We propose to create an architecture inspired by DetNet [33], in which we identify two steps of the GGA and design layers following the same steps as the GGA. The GGA contains a head update step and a gradient descent step of the flows, and we propose a neural network that learns each step separately. We argue that applying algorithm unrolling to the iterative equations of the GGA will result in a fast metamodel for steady-state estimation of WDNs that improves the accuracy of state-of-the-art metamodels for steady-state estimation of WDNs. Furthermore, we propose adding residual connections to the variables in the water network that remain constant throughout the GGA iterations. Residual connections add robustness to the network and allow faster convergence in training [34]. Residual connections can be derived naturally from the GGA algorithm, and our hypothesis was that the inclusion of these increases the accuracy of the model. Therefore, this leads to the following question:

***RQ1**: What is the effect of applying algorithm unrolling to the GGA equations to create a metamodel in terms of accuracy and speed?*

To answer this question, we measured the RMSE metric of different metamodels when predicting the nodal head at all the nodes of five different WDNs. We found that our metamodel is up to 60% more accurate than state-of-the-art metamodels, while being up to 2000 times faster than EPANET. We observed that, similarly to the GNN, our model predicts nodal head of nodes close to reservoirs accurately. However, in contrast to the GNN, our model also reduces the error for nodes further away from the reservoirs.

In addition, to evaluate our hypothesis regarding the inclusion of static information in the form of residual connections, we pose the following question:

***RQ2**: What is the effect of including static information of the water networks, such as supply, reservoir head, and pipe characteristics to our model?*

To answer this question, we present an ablation study that removes components of our architecture and evaluates the effects they have in terms of accuracy and speed. We find that the inclusion of the pipe characteristic have the highest importance in terms of accuracy for most of the WDNs. However, the residual connections to the other system features also improve the accuracy of the model.

In summary, the answers to these questions have led this master thesis to provide the following contributions:

- A novel architecture for steady-state estimation of WDNs derived from applying algorithm unrolling to the equations of the GGA.
- An evaluation of the metamodel in terms of accuracy and speed over five different water networks with varying characteristics.
- An ablation study to understand the effect of the different components of the architecture.
- A presentation at the 19th Computing and Control for the Water Industry Conference 2023 under the title "EPANET Metamodel with Deep Unrolling of the Global Gradient Algorithm".

This thesis is organised as follows. Chapter 2 introduces water distribution networks, EPANET and model-based neural networks. Chapter 3 contains literature on surrogate models for steady-state estimation in WDNs, graph-based techniques in WDNs, and applications of model-based neural networks. Chapter 4 describes the methodology used for this thesis, describing the proposed metamodel, the baselines that will be used to compare our model and the experimental setup. Chapter 5 evaluates the proposed metamodel in five different WDNs and discusses the tradeoff between accuracy and speed. Finally, Chapter 6 concludes this thesis and provides future work towards the application of model-based neural networks in the field of state estimation for WDNs. The code for this thesis is publicly available at `https://github.com/albertsolaroca/GGANet`.

# 2

# Background

In this chapter, we introduce the background information necessary for the following chapters. The following chapters of the thesis contain descriptions of methods in both computer science and civil engineering and, therefore, we describe both topics.

This chapter is structured as follows. Section 2.1 discusses water distribution networks (WDNs). Section 2.2 describes concepts the most widely used tool for WDN steady-state estimation, EPANET, and its underlying algorithm. Section 2.3 describes hybrid models named Model-Based Neural Networks that combine domain knowledge with data, focussing on a technique named Algorithm Unrolling. We end this chapter with a summary in Section 2.4.

## 2.1. Water Distribution Networks

Water distribution networks provide drinking water to users. Simulation-based optimisation is widely used to design and manage WDNs. EPANET is the most widely used software tool to compute the state of the network.

### 2.1.1. Definitions

Water distribution networks transport water [35] through pipe systems from storage facilities to users. Water flows from storage facilities, such as reservoirs and elevated tanks, to customers through pipes connected by junctions. Other components in the network include pumps that pressurise water; or valves that control the water flow and pressures by opening, closing, or partially obstructing the water flow.

WDNs are monitored through sensors such as flow meters, pressure sensors, smart meters for water demand, water level meters in tanks, and water quality sensors and probes throughout the network [36]. These devices measure the state of the water distribution, namely, the hydraulics of the system or the quality of the water. This thesis will focus on the hydraulic state of the water; therefore, the measures of interest for this thesis are:

- Water flow **q**: The water flow **q** is the amount of water that flows through a pipe. Since the density of water is assumed constant, the volumetric flow is equivalent to the mass flow, and consequently, it follows the equation of mass conservation.
- Water head **h**: The hydraulic head measures the total potential energy per unit weight of fluid in a hydraulic system, accounting for elevation, pressure, and kinetic energy. However, kinetic energy is often negligible compared to the pressure and elevation in most practical steady-state conditions. This is due to the limited speed of water in the pipes. Therefore, the head of a node is the sum of its pressure and elevation, and the elevation of the node is known.

Further definitions are necessary to understand the multiple features that contribute to calculating the steady-state of a water distribution network. These are

- Nodal demand **s**: The nodal demand, also known as nodal supply, is the amount of drinking water required by customers at a node.
- Pipe characteristics: The pipe characteristics consists of the length **l**, diameter **d**, and roughness coefficient **c**. The roughness coefficient represents the resistance of a pipe to the flow of water.
- Reservoir heads $\mathbf{h}_0$: Water distribution networks contain a source node from which water initially flows. Throughout this thesis, this source node will be a reservoir, and these reservoirs will have a constant head.

### 2.1.2. Graph Representation of Water Distribution Networks

A WDN can be modelled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of vertices $\mathcal{V}$ corresponds to the $N$ nodes of the WDN, such as demand or connection junctions, storage tanks and reservoirs; and the set of edges $\mathcal{E}$ corresponds to the $M$ links of the water system (e.g. pipes, pumps and valves). To simplify modelling, this thesis will focus on WDNs that contain only junctions, reservoirs, and pipes. In WDNs, the graph also has the particular characteristic that every node or edge should have at least one path of edges connecting it to a source node (tanks and/or reservoirs) [37]. Water distribution networks can be represented as directed or undirected graphs, depending on the problem at hand. Figure 2.1 illustrates the graph representation of a real life water network, the Fossolo water network, as an undirected graph with nodes and edges.

WDNs can be grouped by the type of distribution scheme and the configuration of the network [35]. Distribution schemes are linked to the topographical conditions of the network and can be gravity-based, pump-based, or a combination of the two. WDNs can be serial, branched, grid-like, or a combination thereof [35][38]. However, fully branched directed networks with a single reservoir appear only in simple cases, such as rural areas [39], and generally more complex networks containing loops are the focus of study. Figure 2.2 shows examples of the different structural configurations of a network.

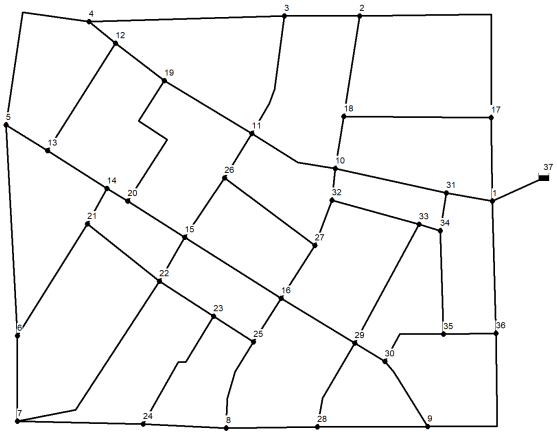
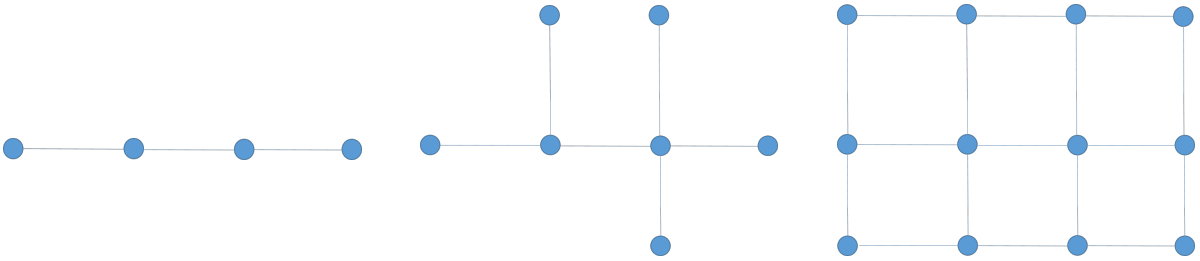



Figure 2.1: Graph representation of the Fossolo water network. The Fossolo water network contains 36 nodes, 58 pipes, and one reservoir. Node 37 represents the reservoir, while all the other nodes are junctions.



(a) Serial Network (b) Branched Network (c) Grid Network

Figure 2.2: Different possibilities for WDN configurations.

## 2.2. EPANET and the GGA Algorithm

Water utilities rely on hydrodynamic models such as EPANET [1] to design, control and analyse WDNs by running physics-based simulations of the network state. The hydraulic state of a WDN consists of the nodal pressure and the link flows. EPANET simulates the flow rates and pressures in all pipes and junctions of a given WDN by solving the underlying equations of mass and energy conservation [40]. In this section, we introduce the underlying algorithm used by EPANET for steady-state estimation. The different variables and matrices used are summarised in Table 2.1 for convenience.

The conservation principles lead to two sets of equations that describe the distribution of flows $\mathbf{q} \in \mathbb{R}^{Mx1}$ and hydraulic heads $\mathbf{h} \in \mathbb{R}^{Nx1}$ where $M$ is the number of pipes and $N$ the number of junctions. The conservation of energy is given by

$$\mathbf{A}_{12}\mathbf{h} + \phi(\mathbf{q}) = -\mathbf{A}_{10}\mathbf{h}_0, \tag{2.1}$$

where $\phi(\mathbf{q})$ represents head loss due to friction as a function of flow through each pipe, $\mathbf{h}_0$ is a vector containing the head in the reservoirs and $\mathbf{A}_{12}$ and $\mathbf{A}_{10}$ represent the directed incidence matrices between pipes-nodes and pipes-reservoirs, respectively. Head loss due to friction $\phi(\mathbf{q})$ is expressed as

$$\phi(\mathbf{q}) = \mathbf{r} \odot |\mathbf{q}|^{\alpha-1} \odot \mathbf{q} \tag{2.2}$$

where $\mathbf{r}$ is a vector of coefficients that varies per pipe, the operator $\odot$ represents the element-wise product between vectors and

$$|\mathbf{q}|^{\alpha-1} = (|q_1|^{\alpha-1}, |q_2|^{\alpha-1}, \ldots, |q_M|^{\alpha-1}) \tag{2.3}$$

The specific pipe coefficient depends on the diameter of the pipe $\mathbf{d}$, its length $\mathbf{l}$, and the roughness coefficient $\mathbf{c}$. The Hazen-Williams formula was designed to approximate the head loss of water [1] and we will calculate the coefficient $\mathbf{r}$ as follows

| Notation | Dimensions | Description |
|---|---|---|
| $\mathbf{q}$ | $M$ x 1 | flowrates in each pipe |
| $\mathbf{h}$ | $N$ x 1 | nodal heads |
| $\mathbf{h}_0$ | $N_0$ x 1 | reservoir heads |
| $\mathbf{s}$ | $N$ x 1 | nodal demands (supply) |
| $\mathbf{c}$ | $M$ x 1 | pipe roughness coefficients |
| $\mathbf{r}$ | $M$ x 1 | Hazen-Williams pipe coefficients |
| $\mathbf{l}$ | $M$ x 1 | pipe lengths |
| $\mathbf{d}$ | $M$ x 1 | pipe diameters |
| $\mathbf{A}_{12} = \mathbf{A}_{21}^T$ | $M$ x $N$ | pipes-nodes incidence matrix |
| $\mathbf{A}_{10} = \mathbf{A}_{01}^T$ | $M$ x $N_0$ | pipes-reservoirs incidence matrix |

Table 2.1: Notation used throughout the thesis. $N$ represents the number of heads, $M$ the number of pipes, and $N_0$ the number of reservoirs. We add the subindices to indicate the connections in the incidence matrices. 0 represents reservoirs, 1 represents junctions, and 2 represents pipes.

$$\mathbf{r} = 10.67\ \mathbf{l} \odot \mathbf{c}^{-1.852} \odot \mathbf{d}^{-4.871} \tag{2.4}$$

Using the Hazen-Williams equations also means that the coefficient $\alpha$ is 1.852. Equation (2.2) can therefore be substituted in Equation (2.1) to obtain

$$\mathbf{A}_{12}\mathbf{h} + \mathbf{r} \odot |\mathbf{q}|^{\alpha-1} \odot \mathbf{q} = -\mathbf{A}_{10}\mathbf{h}_0. \tag{2.5}$$

Additionally, WDNs follow mass conservation, which states that the total inflow equals the total outflow at each node. Therefore, we obtain that

$$\mathbf{A}_{21}\mathbf{q} = \mathbf{s} \tag{2.6}$$

where $\mathbf{s}$ is the vector of nodal demands at the junctions and $\mathbf{A}_{21} = \mathbf{A}_{12}^T$, where $\mathbf{A}_{12}^T$ represents the matrix transpose of $\mathbf{A}^{12}$. Simulations can be either demand-driven or pressure-driven. This thesis focuses on demand-driven simulations, which means that all of the above equations are solved assuming that node demands are known and met. Under demand-driven simulations, full nodal demands are always met even if negative pressures result. This assumption is reasonable under normal operating conditions [41]. In pressure-driven simulations, nodal demands depend on the pressure. We can rewrite the system of equations formed by (2.5) and (2.6) in matrix form as

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{h} \end{bmatrix} = \begin{bmatrix} -\mathbf{A}_{10}\mathbf{h}_0 \\ -\mathbf{s} \end{bmatrix} \tag{2.7}$$

where

$$\mathbf{A}_{11} = \mathrm{diag}(\mathbf{r}|\mathbf{q}|^{\alpha-1}) \tag{2.8}$$

is an $M \times M$ diagonal matrix. Todini and Pilati [42] derived the Global Gradient Algorithm (GGA) from the system of equations in (2.5) and (2.6). The GGA makes use of the Newton-Raphson method to find a solution to the minimisation problem

$$\begin{aligned} &\min_{\mathbf{q}\in\mathbb{R}^M} \mathrm{C}(\mathbf{q}) = \frac{1}{\alpha+1}\langle \mathbf{r}, |\mathbf{q}|^{\alpha+1}\rangle + \mathbf{h}_0^T\mathbf{A}_{01}\mathbf{q} \\ &\text{s.t.} \quad \mathbf{A}_{21}\mathbf{q} = \mathbf{s} \end{aligned} \tag{2.9}$$

where $\mathbf{A}_{01} = \mathbf{A}_{10}^T$ is the pipes-reservoirs incidence matrix and the operator $\langle\cdot\rangle$ represents the vectorial dot product, i.e.

$$\langle \mathbf{r}, |\mathbf{q}|^{\alpha+1}\rangle = \sum_{i=1}^{M} r_i |q_i|^{\alpha+1} \tag{2.10}$$

This equation finds the value of flows in which the integral of the head loss due to friction is equal to the energy inflow from the reservoirs, i.e. solving this minimisation problem finds the steady-state of the network. We will now show that the solution to this minimisation problem is unique and equivalent to solving the system of equations (2.7).

To show that there exists a unique solution to the minimisation problem (2.9) we begin by introducing the Lagrange multiplier $\boldsymbol{\lambda} \in \mathbb{R}^{Nx1}$ to this problem and convert it from a constrained optimisation problem to an unconstrained one. By solving the unconstrained variant of this optimisation problem using the Lagrange multipliers, one finds that the solution to this minimisation problem as follows. Let $\mathbf{q}$ be the unknown pipe flows and $\mathbf{h}$ the unknown nodal heads. We state the unconstrained minimisation problem as

$$\min_{\mathbf{q}\in\mathbb{R}^M, \boldsymbol{\lambda}\in\mathbb{R}^N} \mathsf{L}(\mathbf{q},\boldsymbol{\lambda}) = \frac{1}{\alpha+1}\langle \mathbf{r}, |\mathbf{q}|^{\alpha+1}\rangle + \mathbf{h}_0^T\mathbf{A}_{01}\mathbf{q} + \boldsymbol{\lambda}^T(\mathbf{A}_{21}\mathbf{q}-\mathbf{s}) \tag{2.11}$$

The existence and uniqueness of a solution has previously been proven in [43] by showing that because all the $r_i$'s are positive when $\alpha > 0$, then $\mathsf{L}(\mathbf{q},\boldsymbol{\lambda})$ is convex. The solution to the minimisation problem can then be found by setting the gradient of $\mathsf{L}$ to zero.

$$\begin{aligned} \frac{\partial \mathsf{L}(\mathbf{q},\boldsymbol{\lambda})}{\partial \mathbf{q}_i} &= 0 \quad \text{and,} \\ \frac{\partial \mathsf{L}(\mathbf{q},\boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}_i} &= 0 \end{aligned} \tag{2.12}$$

Solving (2.12) leads to the system of equations

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{A}_{10}\mathbf{h}_0 \\ \mathbf{s} \end{bmatrix} \tag{2.13}$$

Comparing (2.13) and (2.7) we note that the Langrange multipliers $\boldsymbol{\lambda}$ must be the unknown heads $\mathbf{h}$, and therefore the solution to the minimisation problem (2.9) is equivalent to solving the system of equations:

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{h} \end{bmatrix} = \begin{bmatrix} -\mathbf{A}_{10}\mathbf{h}_0 \\ \mathbf{s} \end{bmatrix} \tag{2.14}$$

In the following section, we proceed to obtain the solution to this system of equations.

### 2.2.1. Iterative solution to the GGA

To find the solutions $\mathbf{q}$ and $\mathbf{h}$, the GGA uses the Newton-Raphson method to find the minimum of the system of equations (2.14). The Newton-Rhapson method is an iterative algorithm to find the zeros of a function by taking steps in the direction of the gradient. Given a function $\mathsf{F}$ and its Jacobian $\mathsf{J}_\mathsf{F}$, the Newton-Raphson algorithm states that the zeros of the system $\mathsf{F}(\mathbf{x}) = 0$ can be obtained via:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathsf{J}_\mathsf{F}(\mathbf{x}^{(k)})^{-1}\mathsf{F}(\mathbf{x}^{(k)}) \tag{2.15}$$

We can now rewrite the system of equations (2.14) in the form $\mathsf{F}(\mathbf{x}) = 0$ by defining

$$\mathsf{F} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{q} + \mathbf{A}_{12}\mathbf{h} + \mathbf{A}_{10}\mathbf{h}_0 \\ \mathbf{A}_{21}\mathbf{q} - \mathbf{s} \end{bmatrix} := \begin{bmatrix} \mathsf{f}_1 \\ \mathsf{f}_2 \end{bmatrix} \tag{2.16}$$

and calculate its Jacobian

$$\mathsf{J}_\mathsf{F} = \begin{bmatrix} \frac{\partial \mathsf{f}_1}{\partial \mathbf{q}} & \frac{\partial \mathsf{f}_1}{\partial \mathbf{h}} \\ \frac{\partial \mathsf{f}_2}{\partial \mathbf{q}} & \frac{\partial \mathsf{f}_2}{\partial \mathbf{h}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathsf{f}_1}{\partial \mathbf{q}}\mathbf{A}_{11}\mathbf{q} + \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & 0 \end{bmatrix} = \begin{bmatrix} \mathsf{diag}((\alpha-1)\mathbf{r}|\mathbf{q}|^{\alpha-1}) + \mathsf{diag}(\mathbf{r}|\mathbf{q}|^{\alpha-1}) & \mathbf{A}_{12} \\ \mathbf{A}_{21} & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{D} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & 0 \end{bmatrix} \tag{2.17}$$

with $\frac{\partial \mathsf{f}_i}{\partial \mathbf{q}}$, $i = 1,2$, representing the partial derivative of $\mathsf{f}_i$ with respect to the flows $\mathbf{q}$; $\frac{\partial \mathsf{f}_i}{\partial \mathbf{q}}$, $i = 1,2$, representing the partial derivative of $\mathsf{f}_i$ with respect to the heads $\mathbf{h}$ and

$$\mathbf{D} = \mathsf{diag}(\alpha\mathbf{r}|\mathbf{q}|^{\alpha-1}). \tag{2.18}$$

We can now obtain the inverse of the Jacobian

**Algorithm 1:** Summary of the EPANET algorithm

**Data:** $\mathbf{h}_0, \mathbf{s}, \mathbf{c}, \mathbf{l}, \mathbf{d}, \mathbf{A}_{12}, \mathbf{A}_{10}$
**Result:** $\mathbf{h}, \mathbf{q}$
Compute $\mathbf{r}$ for every pipe using (2.4);
Estimate initial flows;
**while** $|\mathbf{q}^{(k)} - \mathbf{q}^{(k+1)}| > \epsilon$ **do**
  Update $\mathbf{D}$ and $\mathbf{A}_{11}$ using (2.23) and (2.24);
  Compute $\mathbf{f}$ and $\mathbf{A}$ using (2.26) and (2.25);
  Update $\mathbf{h}$ using (2.27);
  Update $\mathbf{q}$ using (2.28);
**end**

$$\mathsf{J}_{\mathsf{F}}^{-1} = \begin{bmatrix} \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{A}_{12}(\mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{12})^{-1}\mathbf{A}_{21}\mathbf{D}^{-1} & \mathbf{D}^{-1}\mathbf{A}_{12}(\mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{12})^{-1} \\ (\mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{12})^{-1}\mathbf{A}_{21}\mathbf{D}^{-1} & -(\mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{12})^{-1} \end{bmatrix} \tag{2.19}$$

Hence,

$$\mathsf{J}_{\mathsf{F}}^{-1}\mathsf{F} = \begin{bmatrix} \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{A}_{12}(\mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{12})^{-1}\mathbf{A}_{21}\mathbf{D}^{-1} & \mathbf{D}^{-1}\mathbf{A}_{12}(\mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{12})^{-1} \\ (\mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{12})^{-1}\mathbf{A}_{21}\mathbf{D}^{-1} & -(\mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{12})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11}\mathbf{q} + \mathbf{A}_{12}\mathbf{h} + \mathbf{A}_{10}\mathbf{h}_0 \\ \mathbf{A}_{21}\mathbf{q} - \mathbf{s} \end{bmatrix} \tag{2.20}$$

$$= \begin{bmatrix} \mathbf{D}^{-1}(\mathbf{A}_{11}\mathbf{q} + \mathbf{A}_{12}\mathbf{h} + \mathbf{A}_{10}\mathbf{h}_0) \\ \mathbf{h} + \mathbf{A}^{-1}(\mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{11}\mathbf{q} + \mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{10}\mathbf{h}_0 - \mathbf{A}_{21}\mathbf{q} + \mathbf{s}) \end{bmatrix} \tag{2.21}$$

With

$$\mathbf{A} = \mathbf{A}_{21}\mathbf{D}^{-1}\mathbf{A}_{12} \tag{2.22}$$

We can now substitute (2.21) in (2.15) to obtain the iterative algorithm to estimate the unknown flows and heads by first calculating:

$$\mathbf{D}^{(k+1)} = \mathrm{diag}(\alpha \mathbf{r} |\mathbf{q}^{(k)}|^{\alpha-1}) \tag{2.23}$$

$$\mathbf{A}_{11}^{(k+1)} = \mathrm{diag}(\mathbf{r} |\mathbf{q}^{(k)}|^{\alpha-1}) \tag{2.24}$$

$$\mathbf{A}^{(k+1)} = \mathbf{A}_{21}\mathbf{D}^{(k+1)^{-1}}\mathbf{A}_{12} \tag{2.25}$$

$$\mathbf{f}^{(k+1)} = \mathbf{A}_{21}\mathbf{q}^{(k)} - \mathbf{s} - \mathbf{A}_{21}\mathbf{D}^{(k+1)^{-1}}\mathbf{A}_{11}\mathbf{q}^{(k)} - \mathbf{A}_{21}\mathbf{D}^{(k+1)^{-1}}\mathbf{A}_{10}\mathbf{h}_0 \tag{2.26}$$

where $\mathbf{f}^{(k+1)}$ is the net flow imbalance at the nodes. Finally, we can obtain the unknown heads and flows iteratively as:

$$\mathbf{h}^{(k+1)} = \mathbf{A}^{(k+1)^{-1}}\mathbf{f}^{(k+1)} \tag{2.27}$$

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - \mathbf{D}^{(k+1)^{-1}}(\mathbf{A}_{11}^{(k+1)}\mathbf{q}^{(k)} + \mathbf{A}_{12}\mathbf{h}^{(k+1)} + \mathbf{A}_{10}\mathbf{h}_0) \tag{2.28}$$

This algorithm then runs until the changes in the flows become negligible. We summarise the GGA in Algorithm 1, which is the principal algorithm of EPANET. The speed of this algorithm is often insufficient for different applications of WDNs, such as optimisation design [44] or criticality assessment [45]. This problem becomes especially notable in large water networks with many nodes and pipes.

The Global Gradient Algorithm is computationally demanding because the underlying physical problem requires solving a potentially very large set of equations. As shown in Equation (2.27), the execution of the iterative algorithm involves computing the inverse matrix of $\mathbf{A}$ at each iteration, since $\mathbf{A}$ is updated at each step. For this reason, WDN modellers have resorted to surrogate models to replace computationally costly models [6]. These surrogate models trade accuracy at the benefit of speedup.

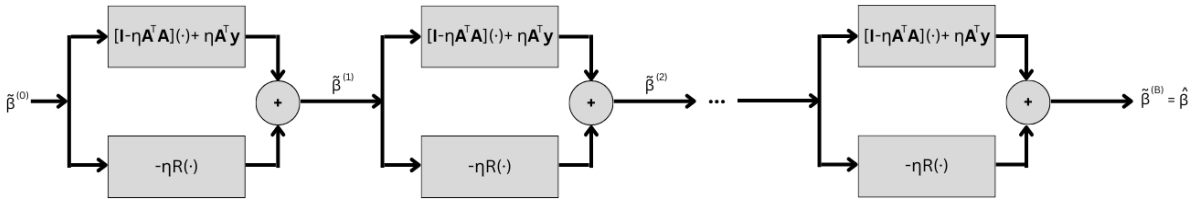



Figure 2.3: Unrolled gradient descent network of the projected gradient descent algorithm. In this case $-\eta\mathsf{R}(\cdot)$ is substituted by an MLP and added to the output of $[\mathbf{I}-\eta\mathbf{A}^T\mathbf{A}](\cdot)+\eta\mathbf{A}^T\mathbf{y}$ to obtain the output for the next layer of the unrolled network.

## 2.3. Model-based Neural Networks

Classical models, also known as analytical models, are based on mathematical equations that describe the behaviour of a system. These models are exact and converge to a solution, but can take a long time to solve, especially for complex systems [26]. In GGA's case, the Newton-Raphson algorithm involves the calculation of a large inverse matrix, which is computationally expensive. Model-based deep learning schemes incorporate domain knowledge in the form of an established model-based algorithm that is suitable for the problem at hand, combined with the ability to learn from data through deep learning [26]. This approach enables the model to be more precise in its predictions, while still retaining the speed and flexibility of neural networks. The inclusion of domain knowledge makes the models more interpretable, making them more easily accepted by decision makers. For these reasons, model-based NNs are often used in control systems and optimisation problems, where statistical or physical models are commonly used. By incorporating some data, model-based NNs can improve the accuracy of statistical or physical models and make them more robust to changes in the system.

### 2.3.1. Algorithm unrolling

An inverse recovery problem considers reconstructing an unknown p-dimensional signal $\boldsymbol{\beta}^\star \in \mathbb{R}^p$ from a set of noisy observations

$$\mathbf{y} = \mathbf{A}\boldsymbol{\beta}^\star + \boldsymbol{\varepsilon} \tag{2.29}$$

where, $\mathbf{y} \in \mathbb{R}^m$ are the observations, $\mathbf{A} \in \mathbb{R}^{m\times p}$ and $\boldsymbol{\varepsilon} \in \mathbb{R}^m$ is a noise vector, one can find the solution to (2.29) via the minimisation problem

$$\tilde{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\, \frac{1}{2}||\mathbf{y}-\mathbf{A}\boldsymbol{\beta}||_2^2. \tag{2.30}$$

However, in some settings, one might have prior knowledge about what values of $\boldsymbol{\beta}^\star$ are more likely [46], which leads to the following formulation of the problem

$$\tilde{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\arg\min}\, \frac{1}{2}||\mathbf{y}-\mathbf{A}\boldsymbol{\beta}||_2^2 + \mathsf{r}(\boldsymbol{\beta}) \tag{2.31}$$

where $\mathsf{r}(\boldsymbol{\beta})$ is a fixed regulariser function for the problem. This optimisation problem can be solved through various techniques such as iterative shrinking thresholding or proximal gradient descent. In the case of projected gradient descent, we begin with an initial approximation $\boldsymbol{\beta}^{(0)}$ and compute:

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \eta[\mathbf{A}^T(\mathbf{A}\boldsymbol{\beta}^{(k)} - \mathbf{y}) + \mathsf{R}(\boldsymbol{\beta}^{(k)})] \tag{2.32}$$

where $\mathsf{R}(\boldsymbol{\beta}) = \nabla\mathsf{r}(\boldsymbol{\beta})$ and $\eta > 0$ is a step size. The final result after $B$ iterations is $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}^{(B)}$. Algorithm unrolling is a technique used in model-based NNs that involves designing neural network layers that imitate the free parameters of each iteration of an iterative optimisation algorithm such as (2.31). Thus, passing through the network is analogous to running the iterative algorithm a finite number of times [30]. This approach allows the model to use data to estimate the classical model's parameters as they become part of the networks' parameters by training the network end-to-end. In this case, we can substitute $\mathsf{R}$ for a trainable neural network. We summarise the resulting unrolled network for Equation (2.32) in Figure 2.3.

However, different optimisation problems lead to different unrolled architectures. With regard to this thesis, the algorithm unrolling architecture that inspired our proposed methodology is DetNet [33]. DetNet uses projected gradient descent to find solutions to the minimisation problem

$$\hat{\mathbf{s}} = \underset{\mathbf{s}\in\{\pm 1\}^K}{\arg\min} ||\mathbf{x} - \mathbf{H}\mathbf{s}||^2 \tag{2.33}$$

where the task is to recover the K-dimensional vector $\hat{\mathbf{s}}$ from $N \times 1$ observations $\mathbf{x}$ that relate to each other via

$$\mathbf{x} = \mathbf{H}\mathbf{s} + \mathbf{w} \tag{2.34}$$

with $\mathbf{w}$ being $N$ i.i.d Gaussian random variables. This problem is solved via projected gradient descent and it iteratively approximates the solution at each step $i$ via

$$\hat{\mathbf{s}}^{(i+1)} = \mathsf{P}_{\mathscr{S}}(\hat{\mathbf{s}}^{(i)} - \eta\mathbf{H}^T\mathbf{x} + \eta\mathbf{H}^T\mathbf{H}\hat{\mathbf{s}}^{(i)}) \tag{2.35}$$

where $\mathsf{P}_{\mathscr{S}}(\cdot)$ represents the projection into the set $\mathscr{S} = \{\pm 1\}^K$. DetNet unrolls the projected gradient descent by building upon the observation that it is composed of two sub-layers. The first sub-layer learns the gradient descent stage

$$\hat{\mathbf{s}}^{(i)} - \eta\mathbf{H}^T\mathbf{x} + \eta\mathbf{H}^T\mathbf{H}\hat{\mathbf{s}}^{(i)} \tag{2.36}$$

by treating the step-size as a learnt parameter and applying a single MLP layer $\phi$ with a ReLU activation function to the obtained value. This results in layer $i$ of DetNet being

$$\mathbf{z}^{(i+1)} = ReLU(\phi((\mathbf{I} + \delta_{1,i}\mathbf{H}^T\mathbf{H})\hat{\mathbf{s}}^{(i)} - \delta_{2,i}\mathbf{H}^T\mathbf{x})) \tag{2.37}$$

where

$$\phi(\mathbf{x}) = \mathbf{W}_{1,i}\mathbf{x} + \mathbf{b}_{1,i} \tag{2.38}$$

with $\{\mathbf{W}_{1,i}, \mathbf{b}_{1,i}, \delta_{1,i}, \delta_{2,i}\}$ learnable parameters in every layer. The second sub-layer learns the projection operator by applying

$$\hat{\mathbf{s}}^{(i+1)} = \text{soft sign}(\psi(\mathbf{z}^{(i+1)})) \tag{2.39}$$

where

$$\psi(\mathbf{z}) = \mathbf{W}_{2,i}\mathbf{z} + \mathbf{b}_{2,i} \tag{2.40}$$

with $\{\mathbf{W}_{2,i}, \mathbf{b}_{2,i},\}$ again learnable parameters.

## 2.4. Summary

This chapter presented the background material that will be used for the remainder of the Thesis. Section 2.1 presented the definitions regarding WDNs. Section 2.2 presented the main algorithm studied, the Global Gradient Algorithm, from which we will develop a surrogate model. Finally, Section 2.3 presented the technique we used to create the proposed surrogate model, namely algorithm unrolling.

# 3

# Related Work

Surrogate models in water distribution networks have been previously used for modelling, monitoring, and forecasting the state of WDNs. This chapter reviews the related work in this regard. Section 3.1 describes the works on surrogate models for WDN state estimation with a focus on machine learning approaches. Section 3.2 shows the uses that graphs and specifically GNNs have had in different applications of WDNs. Finally, Section 3.3 describes the advantages model-based NNs over ANNs and GNNs and their applications in other fields.

## 3.1. Surrogate Models for Steady State Estimation in WDNs

Hydraulic steady-state simulations are performed using software such as EPANET. Running the GGA within EPANET is computationally expensive, especially in applications where many simulations need to be run [4], [5]. Surrogate models replace the computationally expensive simulations of physically-based models to obtain accurate results in a fraction of the time [6]. Surrogate models have been classified in [11] as lower-fidelity physically-based surrogates (LFPB) or response surface surrogates (RS). LFPBs modify the original model to reduce its computational effort by lowering the resolution of inputs, outputs or internal processes [6], while RSs simplify the original model by replacing parts of the model with faster-to-run alternatives. In this thesis, we present a response surface surrogate as parts of the GGA are substituted by neural networks. A RS model was chosen for this thesis because RS models maintain the fidelity of the original model [6]. However, the metamodel presented in this thesis provides a step closer to lower-fidelity physically-based surrogates by incorporating the equations of the GGA.

Surrogate models for WDN state estimation have involved reducing the network to a simplified version to run the GGA algorithm faster. Nonlinear programming-based reduction strategies are provided in both [16] and [17]. The method developed in [16] reduced the network by, for example, combining pipes that were in parallel or in series. This method was also later used in [18] to meet the computational efficiency requirements for the estimation of hydraulic states in real time for urban prototype networks. On the other hand, [17] used spanning trees to simplify the network. However, these simplified models are based on the assumption that low-fidelity models share the same basic features and are correlated to their high-fidelity counterparts [6], which is not always the case.

Machine learning surrogate models have been applied in all areas of water distribution networks, such as design [12], risk assessment [13], or flood prediction [14], as well as state estimation [47]. For example, artificial neural networks (ANNs) were used to estimate nodal pressures in [19]. ANNs were used to extrapolate pressure data by using field data obtained from strategically located pressure sensors and training a neural network to predict pressures at unknown nodes. The authors created a database of the same network by modifying the roughness coefficients in $p$ pipes and the nodal supply at $n$ nodes and performed hydraulic simulations with the new data to obtain the pressures as outputs to train the network. Although their results claim to reduce the degrees of freedom of the calibration problem for applications such as network design, these are not the only values that may vary. Other characteristics of the pipe, such as the length and diameter of the pipes, should also be included when training the ANN. A similar model was developed in [15], where an ANN was used to estimate the pressures at all nodes. The authors used an ANN to improve the calibration of unknown pipe roughness coefficients. ANN's have also been used in [48] as a surrogate model for simulation of pressures and leakages to reduce the computational time of simulating the hydraulic characteristics of the system.

## 3.2. Graph-based Techniques in WDNs

Graph theory has been used in WDN applications for many years, including state estimation. Before the 2000s, the main use of graphs in network design aimed to optimally design WDNs by facilitating integer [21], linear [22], and non-linear programming [23], [24]. As seen in Section 2.2, water in WDNs is governed by basic hydraulic principles and must follow the conservation laws of mass and energy. WDNs can be analysed with a hydraulic approach, taking into account the aforementioned properties, together with pressure, volume requirements, pipe diameter and length and node elevation. However, these hydraulic principles are modelled by computationally expensive equations. The articles mentioned intended to reduce these computational costs by reducing the size of the network.

The work in [21] attempted to maximise the regularity of the network while also reducing redundancies. Regularity aims to have similar degrees in all of the nodes. In [24] graph theory was applied to develop the design of functional and least cost water distribution systems with multiple reservoirs. Similarly, in [23] the authors proposed a reduced successive quadratic programming method for optimal design of pipeline networks where they used graphs to reduce the problem. The authors in [22] proposed an optimal design algorithm suited for WDNs where the WDN has no pumps and multiple reservoirs. The authors used graph decomposition to divide the network into subnetworks based on the connectivity of the networks' components and then proceeded to optimise the subnetworks using a genetic algorithm.

### 3.2.1. Data-Driven Graph-based Techniques in WDNs

Representing WDNs as graphs has helped water modellers to represent these complex systems. Modern WDNs also rely on industrial control systems (e.g. SCADA) for monitoring and controlling operations that provide a large amount of data. All water distribution networks must be instrumented so that state variables can be monitored and state components can be controlled. This, together with recent developments in graph signal processing techniques and their applications in other fields such as smart cities [49] or the power grid [50], has urged researchers to consider the hydraulic characteristics of a WDN. The latter are used to develop more accurate and convenient approaches to estimate features such as unknown nodal pressures through limited measurements [51].

The work in [51] proposed a graph head reconstruction method that uses WDN hydraulics to enhance signal smoothness in the graph domain. From there, the authors reconstruct unknown nodal heads by calculating the low-frequency parts of the Laplacian spectrum. By choosing one of their four proposed metrics, the weights corresponding to different pipes can be calculated with the available WDN information. Of the four metrics proposed, one is based on the hydraulic model of the network, whereas the other three assume that this model is unknown. Finally, the slow-varying parts of the original signal are reconstructed using this information and can be used to estimate the unknown nodal heads. GSP has also been applied to time-varying pressure signals in graphs in [52]. The authors of the paper used the PageRank algorithm [53] to rank sets of vertices based on their pressures at different time points and if there were significant changes in the ordering of the vertices, an anomaly is detected and associated with leaks in the network.

Graph neural networks have been used to estimate flows and pressures in unmonitored locations. For example, the work in [54] used a GNN architecture to localise cyber-physical attacks in water distribution systems. However, the main use of GNNs in WDNs has been to estimate flows and pressures in unmonitored nodes and pipes. The work in [25] used a temporal GNN to estimate pump speeds from pressure and flow measurements. The estimated pump speeds, together with the already measured pressures and flow rates, are used to calculate the flow rates and pressures in the non-monitored nodes. Similarly, in [47] the authors use a GNN to estimate flows and heads in unmonitored locations based on measurements from monitored locations. GNNs have also been applied in [55] in a similar manner, where the authors used a GNN to reconstruct nodal pressures. More recently, the work in [56] used a GNN-based metamodel to estimate network behaviour based on measurements from a limited number of sensors. GNNs show potential for WDN applications, including in applications such as state estimation. However, most of the studies use a small networks and only recently was the transferability of GNNs for WDNs studied in [20].

## 3.3. Applications of Model-based Neural Networks

The computational burden of training and using highly parameterised ANNs and GNNs, as well as the fact that massive datasets are required to train them, are major disadvantages in control applications and signal processing [57]. Most deep learning techniques are purely data-driven, and their underlying structures are difficult to interpret [30]. On the other hand, classical models like the ones mentioned at the beginning of Section 3.2 do not rely on data. However, they rely on simplifying the model assumptions and in doing so fail to represent the nuances of complex systems with the underlying dynamic systems [26]. These systems can be modelled by more complex models albeit compromising computational efficiency. The limitations presented above have led to hybrid models known as model-based algorithms, such as model-based neural networks that combine deep learning with signal processing techniques [26].

Model-based neural networks combine domain knowledge with data-driven modelling. Model-based neural networks can have different structures for different tasks. These structures can be classified as neural building blocks [26], [58], structure agnostic networks [59], [60], structure oriented networks [61], [62], neural augmentation [63], [64], and deep unrolling. We will focus on analysing deep unrolling, also known as algorithm unrolling.

Algorithm unrolling involves deconstructing the iterative algorithm used to solve the optimisation problem of interest into blocks that can be represented as layers in a neural network [30]. Algorithm unrolling was first developed in [27] to improve the computational efficiency of sparse coding algorithms. Following this, many more applications appeared. Algorithm unrolling has been used mainly in the fields of image and signal processing [30], but also in physics-based applications such as power systems [29], where it improves the prediction accuracy compared to simpler deep learning models. One clear advantage that unrolled networks have over generic neural networks is that unrolled networks generally contain significantly fewer parameters, as they encode domain knowledge [30]. Since these neural networks are based on a specific model rather

than general MLPs, they are also more transparent. Algorithm unrolling has been successfully applied to civil infrastructure networks such as power systems [29], where the authors unrolled an iterative physics-based prox-linear solver.

GNN's have mainly been used as model-based neural networks to perform neural augmentation [26]. Neural augmentation utilises the complete model-based algorithm for inference, while the GNN is used to correct some intermediate computations. For example, the authors of [65] created an inference model as a message passing scheme where the nodes of a probabilistic graphical model can send messages to each other to infer estimates of the states.

## 3.4. Summary

Machine-learning-based surrogate models have been applied to WDNs with some amount of success but suffer from the curse of dimensionality. Hybrid models that take into account both the underlying physics and the data have been proven to work effectively for different applications such as graph signal processing or power system state estimation. The inclusion of the underlying topology in the form of using GNN also provides more interpretability to the model. It is clear that WDNs have benefited from the use of GNNs in the past and can now also benefit from the use of hybrid models. In the following sections, we propose a model-based neural network for steady-state estimation in WDNs that combines the accuracy of GNNs with the interpretability from a GSP standpoint.

# 4

# Methodology

The main objective of this thesis is to test the hypothesis that an AU model for metamodeling EPANET has better accuracy and speed than current state-of-the-art metamodels. In this chapter, we describe the method we developed to test this hypothesis. Section 4.1, describes the surrogate model based on the GGA by applying algorithm unrolling. We describe our proposed surrogate approach to the GGA by using algorithm unrolling to estimate the nodal heads of a water network. The main idea is to include the physical equations of the GGA to create layers of the deep learning model to better approximate the iterative approach that the GGA employs. Section 4.2 describes the baseline models we used to compare our algorithm unrolling approach in terms of both accuracy and speed. Section 4.3 describes the hyperparameters studied for all of the models and Section 4.4 describes how we evaluated each surrogate model. Finally, Section 4.5 details how we generated the data for our experiments and the characteristics of the different networks used.

## 4.1. Proposed Metamodel

In this Section, we define the system and hydraulic inputs and the outputs of the models. The algorithm unrolling architecture uses a combination of system and hydraulic features, obtained from the GGA. The system features of the junctions and edges represent constant values of the system, whereas the hydraulic features vary throughout each iteration of the model.

### 4.1.1. Inputs and Outputs

We are dealing with a steady-state estimation, but we have features that stay constant throughout every iteration of the algorithm and some that change for each iteration. The inputs to our model can be divided into those that remain constant throughout the entire run of the algorithm (system features) and those that vary at each iteration (hydraulic features). We will now go into more detail for both cases.

#### System features

The system features are composed of the features of the junctions and of the edges, and can be written as

$$\mathbf{x}_s = \{\mathbf{s}, \mathbf{l}, \mathbf{d}, \mathbf{c}, \mathbf{h}_0\} \tag{4.1}$$

where $\mathbf{x}_s$ comprises the nodal supply $\mathbf{s} \in \mathbb{R}^{Nx1}$, the reservoir head $\mathbf{h}_0 \in \mathbb{R}^{N_0 x1}$, the pipe diameter $\mathbf{d} \in \mathbb{R}^{Mx1}$, length $\mathbf{l} \in \mathbb{R}^{Mx1}$, and the roughness coefficient $\mathbf{c} \in \mathbb{R}^{Mx1}$. These are all defined in Section 2.2. To facilitate the training of the different models, all the system features were normalised following the procedure in [20], where a *log* transformation was fit for the pipe length $\mathbf{l}$ and min-max scaling was used for all the other features.

#### Hydraulic features

The algorithm unrolling approach we will describe in Section 4.1.2 closely follows the steps of the GGA. Each iteration $k$ of the GGA corresponds to a block in the AU approach. Each block $k$ receives as hydraulic features the approximated heads $\mathbf{h}^{(k-1)}$ at the nodes and the flows $\mathbf{q}^{(k-1)}$ in the pipes of the previous block. We can write the hydraulic inputs to each iteration as

$$\mathbf{x}_h^{(k)} = \{\mathbf{h}^{(k-1)}, \mathbf{q}^{(k-1)}\}. \tag{4.2}$$

Since we are dealing with an iterative algorithm, we need an initial value for the flows $\mathbf{q}^0$, which we define as

$$\mathbf{q}^0 = \frac{\pi}{4}\mathbf{d}^2 \tag{4.3}$$

to ensure that flows have a velocity of 1 m/s in each of the pipes. We chose this velocity as an initial condition because EPANET uses an initial velocity of 1ft/s and instead we wanted to ensure using metric system values. Given that $\mathbf{h}^{(0)}$ depends only on the initial supposition of the flow, this will be calculated in the first block of our model.

#### Output

The model outputs the estimated pressure head over the nodes after $K$ iterations of the algorithm, $\mathbf{h}^{(K)} \in \mathbb{R}^{Nx1}$. To facilitate the training of the model, the pressures were normalised using a *log* transformation, as in [20].

### 4.1.2. Unrolling the Global Gradient Algorithm

We propose a metamodel based on the steps in Algorithm 1. The architecture updates the heads and the flows at each block and includes the system features information via residual connections. We begin this Section with a reminder of the equations of EPANET

$$\mathbf{D}^{(k+1)} = \text{diag}(\alpha \mathbf{r} |\mathbf{q}^{(k)}|^{\alpha-1}) \tag{4.4}$$

$$\mathbf{A}_{11}^{(k+1)} = \text{diag}(\mathbf{r} |\mathbf{q}^{(k)}|^{\alpha-1}) \tag{4.5}$$

$$\mathbf{A}^{(k+1)} = \mathbf{A}_{21}\mathbf{D}^{(k+1)^{-1}}\mathbf{A}_{12} \tag{4.6}$$

$$\mathbf{f}^{(k+1)} = \mathbf{A}_{21}\mathbf{q}^{(k)} - \mathbf{s} - \mathbf{A}_{21}\mathbf{D}^{(k+1)^{-1}}\mathbf{A}_{11}^{(k+1)}\mathbf{q}^{(k)} - \mathbf{A}_{21}\mathbf{D}^{(k+1)^{-1}}\mathbf{A}_{10}\mathbf{h}_0 \tag{4.7}$$

$$\mathbf{h}^{(k+1)} = \mathbf{A}^{(k+1)^{-1}}\mathbf{f}^{(k+1)} \tag{4.8}$$

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - \mathbf{D}^{(k+1)^{-1}}(\mathbf{A}_{11}^{(k+1)}\mathbf{q}^{(k)} + \mathbf{A}_{12}\mathbf{h}^{(k+1)} + \mathbf{A}_{10}\mathbf{h}_0) \tag{4.9}$$

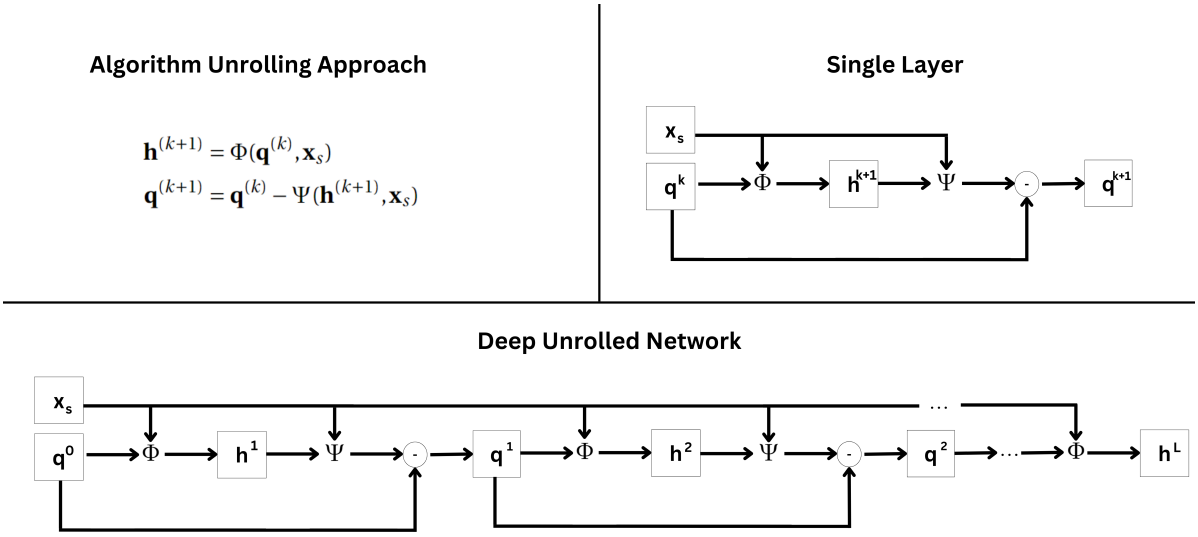



Figure 4.1: Illustration of the base block forming the algorithm unrolling of the GGA. The top left Figure shows the abstraction of the head and flow updates as functions of the previous heads, flows and system variables. The top right Figure shows a diagram of a single block of the algorithm unrolling approach and the bottom Figure shows how stacking algorithm unrolling blocks together forms the entire deep neural network with residual connections. The last layer $L$ of the algorithm unrolling approach finishes with half of a block that returns the heads, as we are not using the flows for this project. This is possible because the dimensions of the inputs and outputs remain the same as the dimensions of the heads and flows throughout the blocks.

Similarly to the DetNet algorithm explained in Section 2.3.1, our methodology consists of identifying two stages in the GGA algorithm. From an EPANET perspective, the GGA has a head update step (4.8) and a gradient step for flows (4.9). Therefore, each unfolded iteration is represented as two sub-layers: the first sub-layer learns to update the heads from the flows and the system features, while the second sub-layer learns to compute the gradient stage by treating matrix $\mathbf{D}$ as learnable parameters at each layer. We can write the two sub-layers as functions of $\mathbf{q}^{(k)}, \mathbf{h}^{(k+1)}$ and $\mathbf{x}_s$ in general as

$$\mathbf{h}^{(k+1)} = \Phi(\mathbf{q}^{(k)}, \mathbf{x}_s) \tag{4.10}$$
$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - \Psi(\mathbf{h}^{(k+1)}, \mathbf{x}_s) \tag{4.11}$$

where $\mathbf{h}^{(k+1)}$ and $\mathbf{q}^{(k+1)}$ correspond respectively to the approximations of the heads and flows at step $k+1$ of the iterative algorithm, $\mathbf{x}_s$ are the system features of our WDN and $\Phi(\cdot)$ and $\Psi(\cdot)$ represent the functions that update the hydraulic variables $\mathbf{q}$ and $\mathbf{h}$, respectively. We update the heads from the flows and vice versa at every iteration of the algorithm using a 1-layer MLPs. Equation (4.11) contains a subtraction instead of an addition to remain as close as possible to Equation 4.9. The different MLPs used will be further detailed. We summarise the algorithm unrolling in Figure 4.1. The final block does not calculate the flows, but stops after the function $\Phi(\cdot)$ to obtain the heads. We chose to obtain only heads as this hydraulic feature is the most widely used for the optimisation of WDNs [66], [44], [67], [68].

### System feature embeddings

As we have seen in Equations (4.10) and (4.11), both steps of the algorithm unrolling approach use the system features. However, the dimensions of the system features might not match the dimensions of the hydraulic features when included in the model. For this reason, we use MLPs to include the system features of the GGA at each step of the algorithm unrolling approach by embedding them at the start of the algorithm and using additions to include them at each necessary step. The only system feature that appears in both Equations (4.8) and (4.9) is $\mathbf{h}_0$, thus we create two separate embeddings of it; as it has different dimensions in both cases. All of the system variables are encoded separately via MLPs as

$$\mathbf{e}_s = \phi_s(\mathbf{s}), \tag{4.12}$$
$$\mathbf{e}_{h_0,h} = \phi_{h_0,1}(\mathbf{h}_0), \tag{4.13}$$
$$\mathbf{e}_{h_0,q} = \phi_{h_0,2}(\mathbf{h}_0), \tag{4.14}$$
$$\mathbf{e}_\varepsilon = \phi_\varepsilon(\mathbf{d},\mathbf{l},\mathbf{c}) \tag{4.15}$$

with $\mathbf{e}_s \in \mathbb{R}^{Mx1}$, $\mathbf{e}_{h_0,h} \in \mathbb{R}^{Nx1}$, $\mathbf{e}_{h_0,q} \in \mathbb{R}^{Mx1}$, and $\mathbf{e}_\varepsilon \in \mathbb{R}^{Mx1}$. We chose the embedding dimensions to be the same as the number of nodes ($N$) for the features used in Equation (4.11) and the same as the number of pipes ($M$) for the features used in Equation (4.10) because this allowed us to use the values of $\mathbf{q}^{(k)}$ and the heads $\mathbf{h}^{(k+1)}$ without embedding them. We avoid embedding the heads $\mathbf{h}$ and the flows $\mathbf{q}$ at every iteration because this allows for further interpretability of the intermediate approximations. The functions

$$\phi_s : \mathbb{R}^N \to \mathbb{R}^N, \tag{4.16}$$
$$\phi_{h_0,1} : \mathbb{R}^{N_0} \to \mathbb{R}^N, \tag{4.17}$$
$$\phi_{h_0,2} : \mathbb{R}^{N_0} \to \mathbb{R}^M, \tag{4.18}$$
$$\phi_\varepsilon : \mathbb{R}^{3M} \to \mathbb{R}^M \tag{4.19}$$

are 1-layer MLPs, followed by a $ReLU$ activation function. The system features do not change over each step, and, therefore, we calculate their embeddings before running any block of the algorithm.

### Dynamic feature embedding

At each iteration $k$, a transformation of the flows is introduced,

$$\mathbf{e}_q^{(k)} = \phi_q(\mathbf{q}^{(k)}) \tag{4.20}$$

by using a 1-layer MLP, followed by a $PReLU$ activation function. A $PReLU$, is an activation function that generalises the traditional rectified unit with a slope for negative values.

$$\phi_q : \mathbb{R}^M \to \mathbb{R}^N \tag{4.21}$$

### Matrix Embeddings

We now proceed to obtain the embeddings of matrix $\mathbf{D}^{(k)}$ from (4.4), which is used in Equations (4.6) and (4.9). Matrix $\mathbf{D}^{(k)}$ needs to be updated at every step, and therefore, we will update the embeddings at every step too. In addition, matrix $\mathbf{D}^{(k)}$ is a diagonal matrix that contains values for all of the pipes. For this reason, we will include the embedded values on the diagonal of matrix $\mathbf{D}^{(k)}$. We create the embedding of $\mathbf{D}^{(k)}$ via

$$\hat{\mathbf{D}}_q^{(k)} = \mathrm{diag}(\phi_{D,q}(\mathbf{e}_\epsilon \odot \mathbf{e}_q^{(k)})) \tag{4.22}$$

with

$$\phi_{D,q} : \mathbb{R}^M \to \mathbb{R}^M \tag{4.23}$$

to approximate $\hat{\mathbf{D}}_q^{(k)}$ from Equation (4.6) and $\odot$ being the element-wise product of the two vector embeddings. In this case, $\phi_{D,q}(\cdot)$ is a 1-layer MLP with a $PReLU$ activation function. Similarly, we will include the system features in equation (4.11) through a matrix of dimension $N \times N$, which we name $\hat{\mathbf{D}}_h^{(k)}$ and is created analogously to $\hat{\mathbf{D}}_q^{(k)}$. We will obtain the embedding for the matrix as

$$\hat{\mathbf{D}}_h^{(k)} = \mathrm{diag}(\phi_{D,h}(\hat{\mathbf{D}}_q^{(k)})) \tag{4.24}$$

where $\phi(\cdot)_{D,h}$ is a 1-layer MLP followed by an activation function $ReLU$

$$\phi_{D,h} : \mathbb{R}^M \to \mathbb{R}^N \tag{4.25}$$

to ensure that the diagonal values of the approximation remain positive. We ensure that the diagonal values of the approximation are positive because the intermediate values of $\mathbf{D}$ in the GGA are also positive, as they represent the headloss at each pipe. For this reason, we use an $ReLU$ activation function after the MLP, as it converts all negative values to zero.

**Algorithm 2:** Summary of the Algorithm unrolling approach

**Data:** $\mathbf{x}_s, \mathbf{q}^{(0)}$
**Result:** $\mathbf{h}$
Compute embeddings of system variables using (4.12), (4.14), (4.13) and (4.15);
**for** *k=1 & k < num_blocks* **do**
  Compute flow embedding $\mathbf{e}_{q,h}^{(k)}$ using (4.20);
  Compute $\hat{\mathbf{D}}_h^{(k)}$ using (4.24);
  Compute $\hat{\mathbf{D}}_q^{(k)}$ using (4.22);
  Update heads $\mathbf{h}^{(k)}$ using (4.26);
  Update flows $\mathbf{q}^{(k)}$ using (4.27);
**end**

### General Architecture

Once we have all the embeddings defined, we can define $\Phi(\cdot)$ and $\Psi(\cdot)$ that transform our hydraulic variables together with the embeddings of the system variables. The layers of our model can be described as

$$\mathbf{h}^{(k+1)} = \Phi(\hat{\mathbf{D}}_q^{(k)}(\mathbf{q}^{(k)} + \mathbf{e}_s + \mathbf{e}_{h_0,q})) \tag{4.26}$$

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - \Psi(\hat{\mathbf{D}}_h^{(k)}(\mathbf{h}^{(k+1)} + \mathbf{e}_{h_0,h} + \mathbf{e}_q^{(k)})) \tag{4.27}$$

where $\Phi(\cdot)$ and $\Psi(\cdot)$ are 1-layer MLPs. $\Phi(\cdot)$ is followed by a *ReLU* activation function, while $\Psi(\cdot)$ is followed by a *PReLU* activation function. The intermediate values of $\mathbf{h}^{(k+1)}$ and $\mathbf{q}^{(k+1)}$ have dimensions $N \times 1$ and $M \times 1$ respectively. This allows for interpretability of the network since the intermediate values obtained also have a physical meaning. In each block, we will add a residual connection to the previous block of our network by subtracting the previous flow from the new one obtained. This is done because (4.9) uses the previous flows at each iteration to compute the gradient step towards the new flows. However, this is not done for the heads $\mathbf{h}$ because these depend solely on the flows we calculated. Moreover, we use subtraction instead of addition in this step to follow more closely the GGA equations. We summarise the steps of the entire algorithm unrolling approach in Algorithm 2. The network is subsequently trained using paired inputs and outputs via back-propagation to optimise the parameters of all of the MLPs.

## 4.2. Baseline Models

In this section, we present the two models that will be compared to the algorithm unrolling approach, a multi-layer perceptron (MLP) and a graph neural network (GNN). We chose to compare our algorithm with an MLP and a GNN because MLPs are the most widely used machine learning surrogate model for the estimation of nodal heads [15], [19]. On the other hand, GNNs were chosen because they include topological information about the network and provide an improvement in accuracy over MLPs [20]. That being said, the improvement in accuracy comes at the cost of speed, which we believe our model will be able to overcome. We now detail the exact structure of both baselines.

### 4.2.1. Multi-Layer Perceptron

Multi-layer perceptrons consist of a series of linear operations applied to input data using a parameter matrix, followed by nonlinear activation functions like ReLU, sigmoid, or tanh. MLPs are the most popular algorithm for WDN metamodelling [6] and we will be using an MLP to approximate the nodal heads. The updating rule for a generic MLP layer can be represented as

$$\mathbf{y}_{l+1} = \phi_l(\mathbf{Y}_l) = \sigma_l(\mathbf{W}_l\mathbf{y}_l + \mathbf{b}_l) \tag{4.28}$$

In this equation, $\mathbf{y}_{l+1}$ represents the output of layer $l$ and has dimensions $H \times 1$, where $H$ is the hidden dimension of the layer; $\sigma(\cdot)$ is the activation function, $\mathbf{W}_l$ is a trainable weight matrix, $\mathbf{y}_l$ is the input to the layer, and $\mathbf{b}_l$ is the learnable bias term. By repeating Equation (4.28) for a certain number of layers $L$, a fully-connected network is formed, that is, the output of layer $l$, $\mathbf{y}_l$ is used as input for layer $l+1$. In the context of WDN state-estimation with $N$ nodes and $M$ pipes, the input to the MLP $\mathbf{y}^0 = \mathbf{x}_s$, as described in Section 4.1.1, and the output $\mathbf{h} = \mathbf{y}^L$ have dimensions $(N + 3E + N_0) \times 1$ and $N \times 1$, respectively.

### 4.2.2. Graph Neural Network

We used the GNN proposed in [20] as another baseline for our model. This GNN consists of an initial preprocessing layer where, first, the edge attributes, namely the diameter **d**, length **l**, and roughness of the pipes **c**, are concatenated with the node attributes of adjacent nodes, namely the supplies at both nodes $\mathbf{s}_i$ and $\mathbf{s}_j$. The concatenated values are then transformed through an MLP shared over all the pipes to obtain an edge embedding $\mathbf{E}'_{ij} \in \mathbb{R}^{M \times G}$, where $G$ is the embedding dimension. For each node, the embeddings $\mathbf{E}'_{ij}$ of the pipes connected to that node are then aggregated to obtain a final node embedding $\mathbf{Y}_0 \in \mathbb{R}^{N \times G}$.

Once this embedding is obtained, $\mathbf{Y}_0$ is passed through multiple layers of ChebNet [69] GNN. The layers of the GNN can be summarised as

$$\mathbf{Y}^{(l+1)} = \sigma\left(\sum_{k=0}^{K} \mathbf{L}^{(k)} \mathbf{Y}^{(l)} \mathbf{H}_l^{(k)}\right) \tag{4.29}$$

where $\mathbf{Y}^{(l+1)}$ is the output layer for layer $l$, $\sigma$ is a $ReLU$ activation function, $\mathbf{H}_l^{(k)}$ is a shared trainable matrix and $K$ is the K-hop neighbourhood, $\mathbf{Y}^0$ as previously described and

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{\frac{-1}{2}} \mathbf{A} \mathbf{D}^{\frac{1}{2}} \tag{4.30}$$

is the Laplacian matrix, used as a graph shift operator. A graph shift operator (GSO) of a graph $\mathcal{G}$ is a matrix $\mathbf{S} \in \mathbb{R}^{NxN}$ such that

$$\mathbf{S}_{ji} = 0 \quad \text{for } (i, j) \notin \mathcal{E} \text{ and } i \neq j. \tag{4.31}$$

Finally, another MLP is used on the node values to convert the final node embeddings into a single nodal head prediction for each node as

$$\mathbf{h} = \phi_L(\mathbf{Y}_L) = \sigma(\mathbf{W}_L \mathbf{Y}_L + \mathbf{b}_L) \tag{4.32}$$

where $L$ is the number of layers of the GNN.

## 4.3. Hyperparameters and Experimental Setup

We ran the MLP and algorithm unrolling experiments using Pytorch (Version 1.12.1)[70] and the GNN experiments using Pytorch Geometric (Version 2.1.0) [71]. All metamodels were trained using the Adam [72] optimisation algorithm with a varying learning rate starting at 0.001 and a fixed step decay of 50% every 20 epochs. The training was carried out for 100 epochs with early stopping and a batch size of 256. In terms of hardware, we employed an 11th Gen Intel(R) Core(TM) i7-11700F @ 2.50GHz and an NVIDIA GeForce RTX 3060, 16Gb RAM GPU. All of the aforementioned hyperparameters are shared throughout all models. In addition, each model has its own set of extra hyperparameters.

For the proposed algorithm unrolling model, the only extra hyperparameter is the number of blocks. This is because we fix the hidden dimensions of the model to be equal to the dimensions of the number of heads and number of pipes where necessary so that the intermediate values of the model have a more clear physical meaning. The GGA converges on average in 6 iterations [1], depending on the configuration and size of the network. However, we chose the maximum number of blocks to experiment with to be 8 to allow our model for slower convergence. This holds based on the idea that each layer in the AU model corresponds to an iteration in the original algorithm. In the case of the MLP, we run a hyperparameter search by changing the number of hidden layers, $l$ from Equation (4.28), and the number of units in each layer, $H$. In the case of the GNN hyperparameters, we modified the embedding dimensions of the shared preprocessing MLP, $G$, the number of graph convolutional (ChebNet) layers after the preprocessing MLP, $l$ from (4.29), the number of output channels of the graph convolutional layers (e.g., number of hidden units) and the max K-hop neighbourhood considered by the GNN from (4.29). We summarise the chosen values of the hyperparameters for the three metamodels in Table 4.1.

| | Values | | |
|---|---|---|---|
| **Hyperparameter** | **MLP** | **AU** | **GNN** |
| Number of Layers | 2,3,4,5,6 | 2,3,4,5,6,7,8 | 1,2,3 |
| Number of Units per Layer | 16,32,64,128,256 | Fixed | 64,128 |
| Embedding dimension | ✗ | ✗ | 32,64 |
| K-hop neighbourhood | ✗ | ✗ | 3,6 |

Table 4.1: Hyperparameters analysed for the MLP, Algorithm Unrolling (AU) and GNN metamodels

## 4.4. Evaluation of the Models

We evaluate the accuracy of the models using the root mean squared error (RMSE) between the predicted hydraulic heads and the calculated heads with EPANET. The RMSE gives us the average error (in meters $H_20$) of our predictions. This metric is calculated as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(h_i - \hat{h}_i)^2} \tag{4.33}$$

It is also important to evaluate the speed-up of the models compared to EPANET. Given that surrogate models are built to overcome the computational costs of simulating using EPANET, the metric to measure speedup is as follows

$$speedup = \frac{\bar{T}_s}{\bar{T}_m} \tag{4.34}$$

Here $\bar{T}_s$ is the average computational time of an EPANET simulation and $\bar{T}_m$ is the average computational time of the model evaluated.

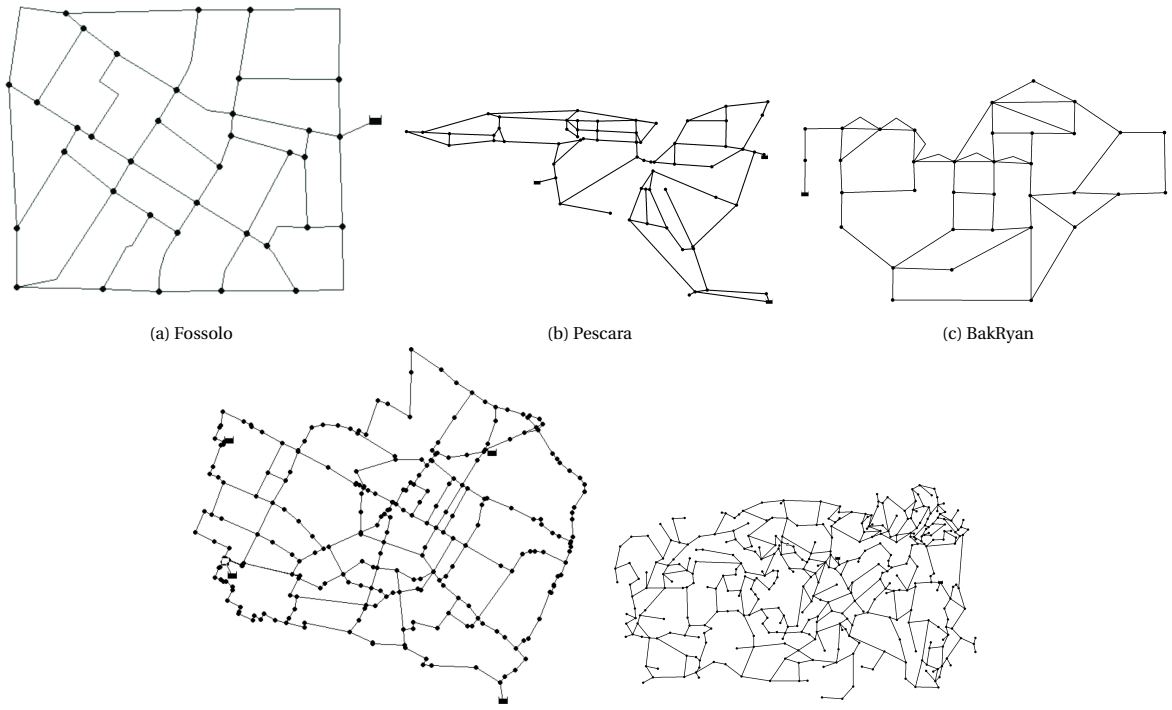
## 4.5. Experiments

All the aforementioned models had to be evaluated both in accuracy and time. All models were trained and tested on individual WDNs. We used benchmark datasets, shown in Table 4.2. We generated 10.000 samples of each WDN divided into training subsets (8000), validation subsets (1000), and test subsets (1000) following the procedure described in [20]. The procedure consisted of modifying different characteristics of the nodes and pipes. The nodal base demands **s** were selected from a uniform distribution of values ranging from 0 to 100 litres per second with increments of 0.1 litres per second. The diameters of the pipe **d** were chosen from a biased distribution that favoured larger values to reduce the possibility of unfeasible configurations and had values from 0 to 1.5 metres with increments of 2.5 cm. Finally, the roughness coefficients of the pipe **c** were selected from a uniform distribution of values ranging from 50 to 150 with increments of 1 unit. All of the above values were selected in that study to reflect commercial ranges for pipe diameters and roughness coefficients and to reflect reasonable base demands with respect to the selected networks. The described alterations ensured sufficient variability in the nodal pressures obtained by demand-driven WNTR simulations [73].

| **Name** | **Id** | **Reference** | **# nodes** | **# pipes** | **# reservoirs** |
|---|---|---|---|---|---|
| BakRyan | BAK | [74] | 36 | 58 | 1 |
| Fossolo | FOS | [75] | 37 | 58 | 1 |
| Pescara | PES | [75] | 71 | 99 | 3 |
| Modena | MOD | [75] | 272 | 317 | 4 |
| Marchi Rural | RUR | [76] | 381 | 476 | 2 |

Table 4.2: Water distribution networks featured in this study and their respective characteristics. These networks were chosen because they represent small, medium and large networks with varying numbers of reservoirs.

Figure 4.2 shows the topologies of the different networks. These networks were chosen because they do not contain pumps and valves and have sufficiently different characteristics to analyse the different methods. When the simulations were run, all other characteristics of the network remained constant, such as network



(a) Fossolo (b) Pescara (c) BakRyan

(d) Modena (e) RuralNetwork

Figure 4.2: Topologies of the different water distribution networks. All of these were chosen because they do not contain pumps and valves and provide a diverse sample of different sizes of WDNs. The characteristics of each of these networks can be found in Table 4.2.

connectivity, geographical coordinates, elevation, pipe lengths, and total head of the reservoirs. These characteristics were chosen to remain constant because we are dealing with a steady-state estimation problem where the topology of the network is constant throughout the process.

## 4.6. Discussion of the Method

In this chapter, we developed a deep neural network by applying an algorithm unrolling approach to the Global Gradient Algorithm. Our model takes a set of system and hydraulic inputs, described in Section 4.1.1 and computes the steady-state of the water distribution network through the algorithm described in Section 4.1.2. Our algorithm unrolling approach is based on the equations of the GGA. We noticed that the GGA was an iterative algorithm with two sub-layers: a head update step and a gradient step. Similarly to the DetNet architecture described in Chapter 2, we wanted to use 1-layer MLPs to learn the head update step and the flow gradient step. In addition, we added residual connections to the system variables through matrix $\hat{\mathbf{D}}^{(k)}$. We noticed that the information of the system is included solely through that matrix, which is diagonal. This allowed us to transform it as vector embeddings and perform element-wise multiplication.

In contrast to other algorithm unrolling approaches in literature, this algorithm contains a high number of parameters, which overall reduces the interpretability of the model and makes it unfeasible for large networks. This is because, in contrast to literature, such as the projected gradient descent mentioned in Chapter 2, where the learnt parameter $\eta$ is a scalar, in the case of the GGA method, we have to learn two embeddings of $\mathbf{D}$ for every iteration of the algorithm, which adds a high number of parameters. We later show that these are necessary to achieve a good performance.

In general, our experimental setup will allow us to compare our algorithm unrolling approach to the state-of-the-art metamodels, namely the MLP and GNN described in Sections 4.2.1 and 4.2.2, respectively. We train the three models to predict the heads at the nodes in the steady-state of the network, as head prediction is the most common use of metamodels in optimisation of water distribution networks. In addition, our experimental setup includes a comparison on the speed of the models, which is necessary to evaluate whether the metamodels are suitable substitutes for EPANET.

# 5

# Results

Metamodels trade accuracy of predictions for speedup when compared to the hydraulic model. In this chapter, we compare the approaches proposed in Chapter 4 on five different water networks. Section 5.1 compares the algorithm unrolling approach to the MLP and GNN baselines in terms of performance and execution time. Section 5.2 further analyses the results obtained by studying the accuracies of the models for each network and nodes. In Section 5.3, we present the results of the metamodels in terms of speedup when compared to EPANET. Section 5.4 provides an ablation study to determine the importance of the different components of our model. Finally, Section 5.5 provides an overall discussion of the results.

## 5.1. Tradeoff

Table 5.1 presents the comparison in terms of RMSE and computational speed-ups, together with the total number of parameters of the best metamodels. The algorithm unrolling (AU) approach performs better than both the MLP in terms of the overall RMSE of the model, producing better predictions for all WDNs, with improvements of 12% in PES, 15% in MOD, 20% in RUR, 30% in BAK and up to 60% in FOS. This is because the MLP tends to predict similar values for all nodes, leading to a high error in the overall network. On the other hand, our algorithm unrolling approach produces more variety for the predictions at the nodes and overall decreases the error for each network. Despite being between 2 and 3 times slower than the MLP, the AU approach still provides speedups of 732 to 2063 times compared to EPANET.

Compared to GNN, the AU approach produces better results in all networks except the PES network, with improvements of 11% in FOS, 23% in MOD, 26% in RUR and 32% in BAK, while being only 3% less accurate in PES. The GNN accurately predicts nodes closer to reservoirs, while our model distributes the error throughout the network. This means that the relative improvement with respect to the GNN is smaller in networks with a high number of reservoirs, such as PES and MOD, while we present higher improvements in networks with a lower number of reservoirs. That being said, our algorithm unrolling approach is 10 to 25 times faster than the GNN throughout all networks, as the number of computations needed is lower.

Finally, Table 5.1 includes the number of parameters for each of the best models. We can observe that the algorithm unrolling approach uses less parameters than both the MLP and the GNN for the BAK, FOS and PES water networks, which are the smaller networks. However, as the size of the network increases, the more parameters our model needs. This increase in parameters reduces the interpretability of the model and makes it unfeasible for large networks, as more samples are needed to train the network.

| | RMSE (m) | | | Speedup | | | # Parameters | | |
|---|---|---|---|---|---|---|---|---|---|
| **Network** | **MLP** | **AU** | **GNN** | **MLP** | **AU** | **GNN** | **MLP** | **AU** | **GNN** |
| **BAK** | 0.481 | **0.339** | 0.502 | **1421** | 732 | 75 | 190K | **85K** | 111K |
| **FOS** | 3.677 | **1.391** | 1.572 | **1752** | 814 | 67 | 261K | **88K** | 247k |
| **PES** | 5.813 | 5.079 | **4.921** | **2086** | 2063 | 82 | 177K | **159K** | 222K |
| **MOD** | 1.195 | **1.010** | 1.318 | **2103** | 841 | 62 | **224K** | 1.34M | 247K |
| **RUR** | 1.105 | **0.885** | 1.198 | **3633** | 1195 | 63 | 297K | 3.11M | **111k** |

Table 5.1: Summary of the accuracies, speedups and number of parameters of the best models in each network.

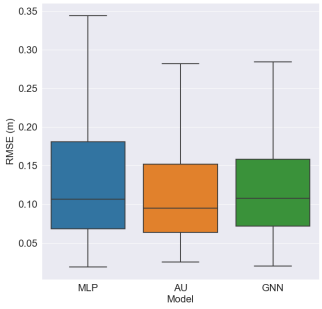
## 5.2. Accuracy

Figure 5.1 shows the RMSE in metres of $H_20$ over the 1000 test networks for BAK, FOS, PES, MOD and RUR. In these figures, we have removed the outliers to show the differences between the models more clearly. We include the outliers in Figures A.1a-A.1e in Appendix A.1.

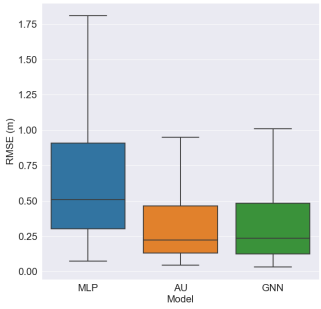
The average error across the entire network is always lower in the AU approach. More precisely, Figure 5.1a shows little difference between the averages of the three models; all of them have an average error of around 10 cm throughout the entire network. The main reason the AU model performs better in terms of RMSE is that it significantly reduces the error in outliers, as seen in Figure A.1a. A similar pattern occurs throughout all WDNs except for PES, where the AU approach reduces the overall error by better approximating the outliers. Figure 5.1b shows that the AU approach behaves similarly to the GNN, both approaches reduce the average error of the MLP from 50 cm to 25 cm. In Figure A.1b we can see that this is obtained mainly by reducing the outliers from errors of 40 m to errors of around 20 m. Figure 5.1c shows that all three models perform poorly in the PES water network, with average errors of 3 to 4 m. Figure 5.1d shows the largest differences between the three models, where the AU approach improves the averages throughout the network compared to the MLP and GNN baselines. Finally, Figure 5.1e shows once again an improvement in the AU approach averages when compared to the MLP and the GNN.

It is also important to analyse the accuracy for each of the nodes of the different networks to understand how the error is distributed through the network. Figure 5.2 shows the RMSEs on the nodes the different water networks for each of the models without outliers. The outliers are included in Figures A.2a-A.2e from Appendix A.1. The GNN has the highest standard deviation of all the models, followed by the AU approach and finally the MLP. This is because the MLP distributes the error equally throughout the network, whereas

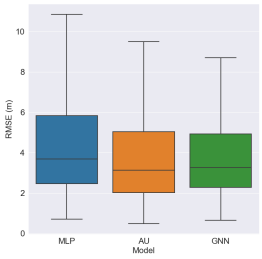
the GNN and the AU approach better approximate nodes close to the reservoirs, while having more error the further away we move from the reservoirs. This can be best seen in Figures 5.2b and 5.2c. In the case of the MLP, the error is higher and more averaged throughout the network, whereas the AU approach and the GNN focus the error in nodes further away from the reservoirs.



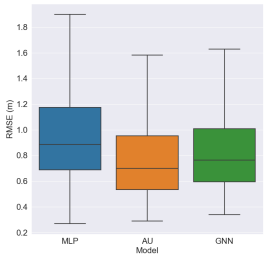

(a) RMSE of the BAK water network.



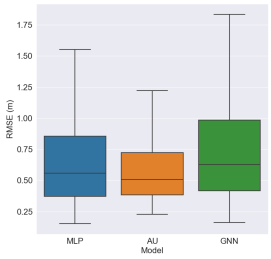

(b) RMSE of the FOS water network.




(c) RMSE of the PES water network.

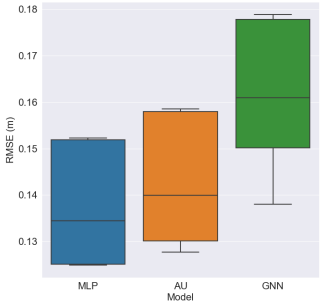



(d) RMSE of the MOD water network.

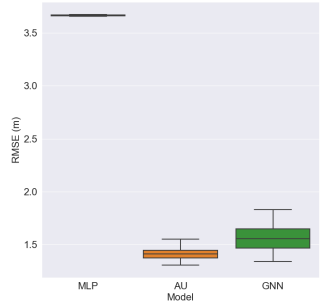



(e) RMSE of the RUR water network.

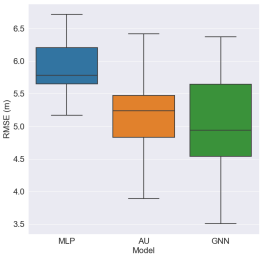
Figure 5.1: RMSEs over the generated networks of the different water networks for each of the models without outliers.



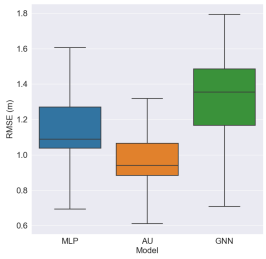

(a) RMSE of the BAK water network.



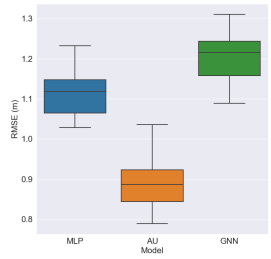

(b) RMSE of the FOS water network.




(c) RMSE of the PES water network.




(d) RMSE of the MOD water network.

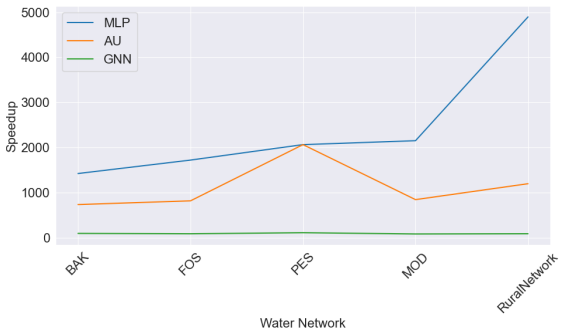



(e) RMSE of the RUR water network.

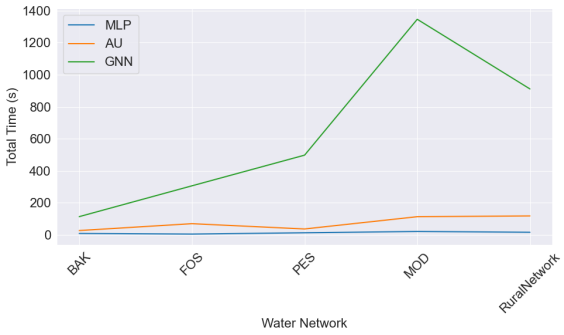
Figure 5.2: RMSEs over the nodes the different water networks for each of the models without outliers.

## 5.3. Speedup

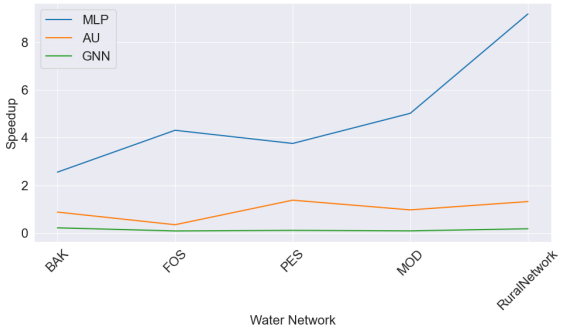
We now compare the speedups of the different models to EPANET. Figure 5.3a shows the average speedup compared to EPANET of the different models in each WDN over 1000 test samples. We observe that, once trained on 8000 samples of the WDN, the MLP is the fastest model. However, the algorithm unrolling approach still provides average speedups between 732 and 2063 times the speed of EPANET because its architecture only uses MLPs. We can also observe that the GNN model is the slowest model to run, with speedups of only 62 to 82 times the speed of EPANET.




(a) Speedup for the different models in each WDN.




(b) Total time to train and test the different models in each WDN.




(c) Overall speedup when including training and testing the different models in each WDN.

However, the speedup for the test time is not the only thing that matters when studying metamodels, as these also have to be trained before being deployed. Following this, we also provide the total run times in seconds for training and testing the distinct WDNs in Figure 5.3b. We observe that the MLP is once again the model that takes the least time to train and test. However, we observe that while the AU approach is still slower than the MLP, it's computational time is closer to the MLP than to the GNN for all WDNs. It is important to note that the total time and speedup provided are inversely proportional to each other. In other words, the more time an algorithm takes to run, the lower its speedup. For this reason, models that take little total time to run provide high speedups.

The total time that an algorithm takes is also reflected in the overall speedup when including training and testing times. Figure 5.3c shows the speedups for the total time with respect to EPANET. We can observe that the pattern remains similar to that of Figure 5.3a, with the MLP providing the highest speedups. That being said, we can observe that the scale of improvement is lower than when taking only the test times into account. We still see that the MLP provides a speedup of 2 to 8 times when including the training time. On the other hand, the algorithm unrolling approach provides a small speedup when including training only for the PES and RUR network, and is slightly slower for the rest of the networks. Finally, the GNN does not provide a speedup in any network. This is important because the MLP remains the only metamodel to provide speedups in case the metamodel was to be used only once. However, for most applications where metamodels are desired, such as optimisation of the network, the metamodels will be used multiple times. Once trained, the metamodel will have the speedups shown in Figure 5.3a for all subsequent samples.

## 5.4. Ablation Study

We now present an ablation study to understand the effect that each of the components of our model has on both the accuracy and speeds of our model. Table 5.2 shows the different components we removed in each experiment, together with the resulting RMSE in each network and the respective speedup. All the hyperparameters used in these experiments were the ones obtained for the best AU metamodel described in Section 5.1, for each network. This is because convergence in some networks was faster than in others, and, therefore, the number of blocks varies from network to network. The different components removed were

- $\hat{\mathbf{D}}_q^{(k)}$ from Equation (4.26). Resulting in the following architecture:

$$\mathbf{h}^{(k+1)} = \Phi(\mathbf{q}^{(k)} + \mathbf{e}_s + \mathbf{e}_{h_0,q}) \tag{5.1}$$

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - \Psi(\hat{\mathbf{D}}_h^{(k)}(\mathbf{h}^{(k+1)} + \mathbf{e}_{h_0,h} + \mathbf{e}_q^{(k)})) \tag{5.2}$$

- $\hat{\mathbf{D}}_h^{(k)}$ from Equation (4.27). Resulting in the following architecture:

$$\mathbf{h}^{(k+1)} = \Phi(\hat{\mathbf{D}}_q^{(k)}(\mathbf{q}^{(k)} + \mathbf{e}_s + \mathbf{e}_{h_0,q})) \tag{5.3}$$

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - \Psi(\mathbf{h}^{(k+1)} + \mathbf{e}_{h_0,h} + \mathbf{e}_q^{(k)}) \tag{5.4}$$

- $\mathbf{e}_{\mathbf{x}_s}$ represents the embeddings of all the system variables, i.e. $\mathbf{e}_s, \mathbf{e}_{h_0,q}, \mathbf{e}_{h_0,h}$ from Equations (4.26) and (4.27); and $\mathbf{e}_c$ from (4.22). Resulting in the following architecture:

$$\mathbf{h}^{(k+1)} = \Phi(\hat{\mathbf{D}}_q^{(k)}\mathbf{q}^{(k)}) \tag{5.5}$$

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - \Psi(\hat{\mathbf{D}}^{(k)}(\mathbf{h}^{(k+1)} + \mathbf{e}_q^{(k)})) \tag{5.6}$$

- $\mathbf{e}_q^{(k)}$ from Equation (4.27). Resulting in the following architecture:

$$\mathbf{h}^{(k+1)} = \Phi(\hat{\mathbf{D}}_q^{(k)}(\mathbf{q}^{(k)} + \mathbf{e}_s + \mathbf{e}_{h_0,q})) \tag{5.7}$$

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} - \Psi(\hat{\mathbf{D}}_h^{(k)}(\mathbf{h}^{(k+1)} + \mathbf{e}_{h_0,h})) \tag{5.8}$$

We can observe that for BAK, PES, MOD networks, the matrix $\hat{\mathbf{D}}_q^{(k)}$ has the highest influence of all components, as removing it produces the highest differences in terms of RMSE. On the other hand, for the RUR water network we can observe that matrix $\hat{\mathbf{D}}_h^{(k)}$ has the highest influence, since it increases the RMSE from 0.885 to 1.712. This is because matrices $\hat{\mathbf{D}}_q^{(k)}$ and $\hat{\mathbf{D}}_h^{(k)}$ are embeddings of the characteristics of the pipe $\mathbf{l}, \mathbf{d}, \mathbf{c}$ and the flows $\mathbf{q}^{(k)}$. Finally, for the FOS water network, the embeddings of the rest of the system features, $\mathbf{e}_{\mathbf{x}_s}$, have the greatest impact, increasing the RMSE from 1.391 to 2.983. Overall, we can observe that all of the components of our model have an effect on the accuracy and that the magnitude of this effect depends on the water network on which we are working.

| Settings | | | | RMSE | | | | | Speedup | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\hat{\mathbf{D}}_q^{(k)}$ | $\hat{\mathbf{D}}_h^{(k)}$ | $\mathbf{e}_{\mathbf{x}_s}$ | $\mathbf{e}_q^{(k)}$ | BAK | FOS | PES | MOD | RUR | BAK | FOS | PES | MOD | RUR |
| ✓ | ✓ | ✓ | ✓ | 0.339 | 1.391 | 5.079 | 1.010 | 0.885 | 732 | 814 | 2063 | 841 | 1195 |
| ✗ | ✓ | ✓ | ✓ | **0.654** | 1.910 | **8.024** | **2.703** | 1.173 | **1349** | 1244 | 1430 | 734 | **1515** |
| ✓ | ✗ | ✓ | ✓ | 0.649 | 1.748 | 6.965 | 2.451 | **1.712** | 1053 | **1318** | 2010 | 1630 | 1371 |
| ✓ | ✓ | ✗ | ✓ | 0.499 | **2.983** | 6.596 | 1.453 | 1.014 | 988 | 1147 | 2314 | 1569 | 1321 |
| ✓ | ✓ | ✓ | ✗ | 0.565 | 1.653 | 7.133 | 1.941 | 1.072 | 865 | 1036 | **2412** | **1644** | 1328 |

Table 5.2: Configurations used in the ablation study, together with the results RMSE over all the nodes and the speedups when compared to EPANET. The first configuration represents the metamodel proposed in this Thesis, while the other settings remove parts of this metamodel to evaluate the effect each component has on the overall accuracies and speeds.

The removal of matrix $\hat{\mathbf{D}}_q^{(k)}$ results in the highest increase in speed for the BAK and RUR networks. Similarly, removing the embedding of $\mathbf{e}_q^{(k)}$ also has the highest increase in speed in two networks, in this case PES and MOD. Finally, for the FOS water network, the removal of $\hat{\mathbf{D}}_h^{(k)}$ has the greatest impact in terms of speed. This is because removing the matrices and the flow embedding results in removing a multiplication at each step. On the other hand, the system variables result in the lowest increases in speedup. This is because they do not change throughout the iterations and as we are only reducing 3 additions per iteration, which is less costly than multiplications.

## 5.5. Discussion

Throughout this chapter, we have seen that the proposed metamodel based on the GGA presents an overall improvement in terms of accuracy compared to the MLP and GNN baselines on five different water networks. The algorithm unrolling approach produces better predictions for all WDNs, with improvements of up to 60% compared to the MLP and up to 32% compared to the GNN. In addition, the proposed metamodel presents speedups compared to EPANET of magnitudes in the hundreds and the thousands. Although the model was slower than the baseline MLP, this speed-up still presents a significant improvement to EPANET. We believe that the overall increase of accuracy with comparison to the MLP and GNN baselines, together with the speedup of hundreds to thousands of times when compared to EPANET will be useful to water modellers in tasks such as optimisation, design, and calibration of water networks.

In addition, we have observed how all components that form this metamodel are important to make it work correctly through an ablation study. We have observed that different networks require different components to function correctly. Each water network has distinct characteristics that give different importance to the different components. In terms of precision, the most important component is matrix $\hat{\mathbf{D}}_q^{(k)}$, as it includes information on the characteristics of the pipe. Whereas in terms of speed, all components are influenced differently on the basis of the characteristics of the different water networks. We believe that the differences shown are a result of small differences created by having speeds of milliseconds in all cases.

Finally, we have also observed that our model seems to suffer from the curse of dimensionality. The number of parameters increases exponentially with the size of the water network, especially with the number of pipes, since the matrices $\hat{\mathbf{D}}_q^{(k)}$ and $\hat{\mathbf{D}}_h^{(k)}$ must be calculated at every block of the AU approach. That being said, even in large networks such as MOD or RUR our model seems to be robust to the number of parameters, as the highest precision is still obtained using our metamodel for the MOD and RUR network.

# 6

# Conclusion

This master thesis aimed to create a WDN steady-state estimation metamodel based on the physical equations that govern WDNs. Our main research question was

***RQ**: How can we develop a deep neural network that includes the physical information of the WDN?*

which we answered by creating a metamodel that applied algorithm unrolling to the GGA. We found that our algorithm unrolling approach provides increases in accuracy of up to 60% when compared to state-of-the-art metamodels for steady-state estimation and an increase in speed of up to 2063 times when compared to EPANET.

In Chapter 1 we explained the need for metamodels for WDN steady-state estimation and the problem that current metamodels presented to provide motivation for the use of model-based neural networks. Chapter 2 provided the necessary background, including definitions and information on water distribution networks. In this Chapter, we also explored the most widely used tool for state-estimation in WDNs, namely, EPANET and its underlying iterative algorithm, the GGA. We also explained the need for surrogate models of EPANET and presented the main category of algorithms used in this thesis, model-based neural networks. Chapter 3 reviewed the current literature on the most widely used surrogate models in state-estimation in WDNs and showed how other fields related to WDNs apply graph-based techniques. Finally, we also included the current literature related to applications of model-based neural networks in other fields.

Chapter 4 presents the methodology of this master thesis and answers the main research question:

***RQ**: How can we develop a deep neural network that includes the physical information of the WDN?*

To do so, we proposed a metamodel for water distribution network state estimation based on applying algorithm unrolling to the Global Gradient Algorithm in Section 4.1, which is our main contribution. At the time of writing, the best metamodels for EPANET have been MLPs and GNNs. However, we postulated that our hybrid approach based on algorithm unrolling could provide a significant improvement with respect to MLPs and GNNs in terms of overall accuracies and speeds by making use of the physical equations of the GGA.

Finally, Chapter 5 presented the results in both accuracies and speeds throughout the experiments. As we mentioned throughout this thesis, metamodels trade accuracy for speed, and we showed that our model overall improved the results obtained from the baselines. We studied the accuracy and speedup of our metamodel in five different water networks with varying characteristics. We observed that our proposed metamodel increased the accuracy of state-of-the-art metamodels by up to 60% while being up to 2000 times faster than EPANET. We now summarise the main contributions to this thesis, how the individual research questions were answered and provide limitations and future work.

## 6.1. Summary of the findings

Developing a suitable metamodel for the problem of steady-state estimation requires studying the trade-off in accuracies and speeds. To determine whether the developed metamodel presented an improvement with regard to state-of-the-art metamodels, we posed the following question:

***RQ1**: What is the effect of applying algorithm unrolling to the equations of the GGA to create a metamodel in terms of accuracy and speed?*

We postulated that applying algorithm unrolling to the GGA would allow us to include the physical equations of the GGA to create an explainable metamodel that is faster than EPANET and provides more accurate results than the ML baselines used until now. We tested this hypothesis by predicting the heads of five WDNs with different characteristics when varying their nodal demands and pipe characteristics in Chapter 5. We have seen that our deep unrolled network presented higher accuracies than the MLP in the five studied networks. However, when comparing speeds of the MLP and our metamodel, the algorithm unrolling approach was half as fast as the MLP. When compared to a GNN, our metamodel presented higher accuracies in all networks except one. That being said, when compared to the GNN, our metamodel was up to ten times faster. In Section 5.5 we discussed that metamodels trade accuracy for speed and concluded that our algorithm unrolling metamodel improved on state-of-the-art metamodels by being more accurate while presenting comparable speeds.

***RQ2**: What is the effect of including static information of the water networks such as supply, reservoir head, and pipe characteristics to our model?*

We included static information about WDNs using residual connections to the length, diameter, roughness coefficient, reservoir heads, and supplies at the junctions. We answered this question in Section 5.4, where we saw that these residual connections allow the algorithm unrolling approach to improve the results obtained by the metamodel without residual connections by up to 50% in terms of RMSE.

## 6.2. Limitations and Future work

To conclude this master thesis, we discuss some limitations our metamodel shows and propose suggestions to improve them. The main limitations of our work are as follows:

1. Our algorithm unrolling approach leads to a steeper increase in parameters when compared to algorithm unrolling approaches found in literature. This is because in our case we are using algorithm unrolling to estimate entire matrices in our model. This leads to needing a high number of samples which might not be available for every WDN, as the need to create large training datasets using EPANET, which is slow.

2. The algorithm unrolling approach can not be used to substitute EPANET for applications that rely on flow prediction. Despite calculating the flows at each block of the algorithm unrolling approach, we do not use the flows as an output for the model for training and testing.

3. Finally, our algorithm unrolling approach is limited to demand-driven simulations of WDNs without pumps or valves. While we do base this algorithm on the GGA, normal network conditions were assumed. However, pressure-driven simulations were not taken into account to train the model.

Following the previous list of limitations, we suggest that future work explores the following options:

1. Regarding limitation 1, the inclusion of graph methods in the algorithm can help reduce the number of parameters, as the embeddings that are generated in the method approximate multiplications by incidence matrices. This should convert this algorithm into a metamodel more similar to GNNs, reducing the number of parameters, and consequently reducing the number of samples needed to train the metamodel. However, multiple attempts to include these incidence matrices were made throughout the thesis, but none improved the accuracy of the proposed metamodel while providing speed-up with respect to EPANET. When including the incidence matrices, the main bottleneck of the algorithm becomes the calculation of $\mathbf{A}^{-1}$, which needs to be recalculated at every iteration and is very slow since the matrix has size $M \times M$.

2. With regard to limitation 2, an extension of the algorithm that outputs the flows can be developed using L2-loss over the flows.

3. Finally, limitation 3 can be addressed by following the steps EPANET applies when dealing with pumps and values. In addition, pressure-driven simulations should be included in the training dataset.

## 6.3. Broader impact of the thesis

Water modellers rely on EPANET for computations of the steady-state of the WDN. The presented metamodel will allow water modellers to not use EPANET for applications that require many simulations of the state of

the water network. We have shown that the inclusion of physical information from the system improves the accuracy of state-of-the-art metamodels for steady-state estimation of WDNs. Our metamodel will aid water modellers in the design and optimisation phase of a WDN as they can test more designs of the network in less time.

Furthermore, our metamodel provides a starting point for models based on the physical information of the system. This can be used in different applications, both in the field of water modelling and outside. Within WDNs, the problem of real-time state estimation also depends on the physical equations of mass and energy conservation of the WDN and we encourage expanding this thesis with a study on real-time state estimation. Similarly, other areas of water modelling also use iterative algorithms to calculate the state of the network, for example, for urban drainage systems the equivalent to EPANET is SWMM [77], which uses the Euler method to obtain an iterative algorithm.

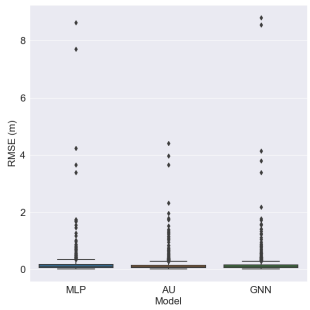
Many applications rely on numerical methods to calculate solutions to the system at hand. Methods such as the Euler method, the Gauss-Newton algorithm or gradient descent are all used to solve optimisation problems, ordinary differential equations, and to find roots of equations. All of these numerical methods result in iterative algorithms which can be unrolled into deep neural networks depending on the problem at hand. These numerical methods also share that they all result in low speeds, as many iterative steps need to be calculated to obtain the solution of the problem. We encourage researchers with problems that can be solved through iterative algorithms to consider applying algorithm unrolling to the iterative algorithm and training a neural network.

# A

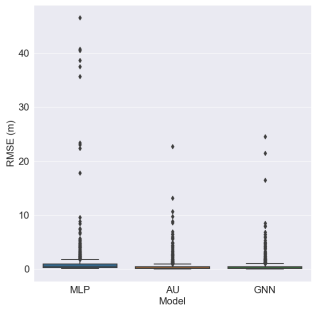
# Appendix

## A.1. Network RMSE with outliers

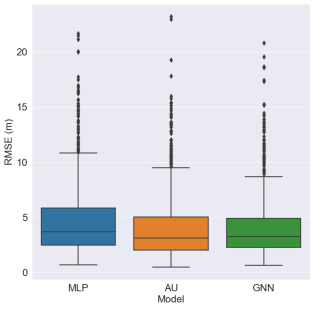
Figure A.1 shows the RMSE in metres of $H_20$ over the 1000 test networks for BAK, FOS, PES, MOD and RUR when including outliers. We can observe that the algorithm unrolling approach provided the largest increase in improvement in samples that were hard to predict in all the networks except for PES. Although the average error for all the models is less than 6 meters, the outliers can have RMSE of up to 45 meters throughout the network. In these extreme cases, our algorithm unrolling approach best works. The algorithm unrolling approach reduces the error by 50% in the FOS network compared to the MLP and 45% in the BAK network compared to the GNN in the worst case. However, the data generation for this project favoured configurations of the network with reasonable pressures. This means that the test database did not contain many scenarios with low pressures throughout the network, which is where the MLP and GNN struggled the most in comparison to the algorithm unrolling approach.



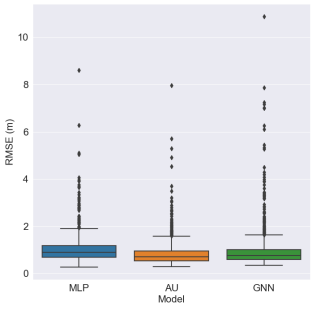

(a) RMSE of the BAK water network.



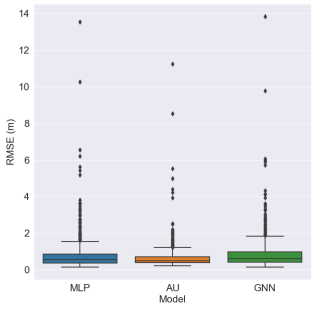

(b) RMSE of the FOS water network.




(c) RMSE of the PES water network.




(d) RMSE of the MOD water network.



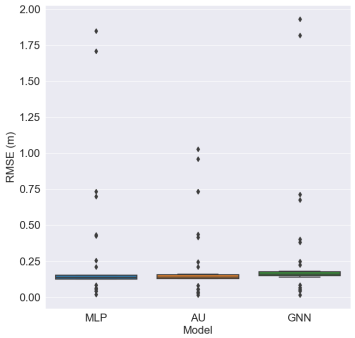

(e) RMSE of the RUR water network.

Figure A.1: RMSEs over the generated networks of the different water networks for each of the models with outliers.

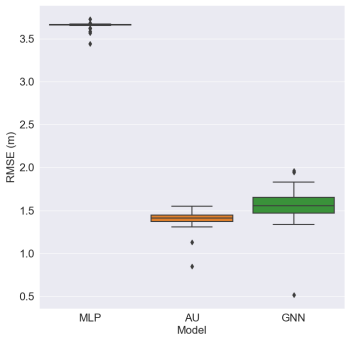
The main consequence of the improvement in outliers is the decrease in the overall RMSE, seen in Table 5.1. As we have seen, the algorithm unrolling approach presents the largest improvements in the FOS and BAK networks. This is due to the improvements seen in the outliers in Figures A.1b and A.1a.

## A.2. Nodal RMSE with outliers

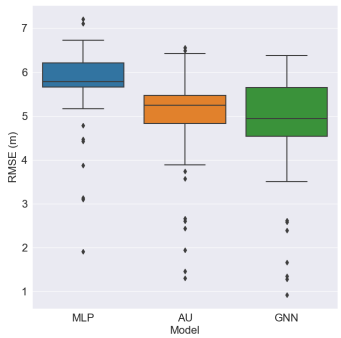
Including the outliers when studying the RMSE over the nodes shows that the algorithm unrolling approach and the GNN have nodal outliers that have a very low error. This is because both of these methods, by including the physical information of the system and the topology, respectively, very accurately predict nodes that are close to reservoirs. This can be best seen in the cases of the Fossolo and Modena water networks. In Figure A.2b we see that the two outliers that have now appeared for the FOS network are the two closest nodes to the single reservoir of the network. In Figure A.2d we see a similar pattern, with the three outliers that appeared to have the lowest values being 3 of the 4 nodes connected to a reservoir. With regard to the MLP, we see that overall it has the lowest variance of all of the models. This is because the MLP distributes the error equally throughout the network.



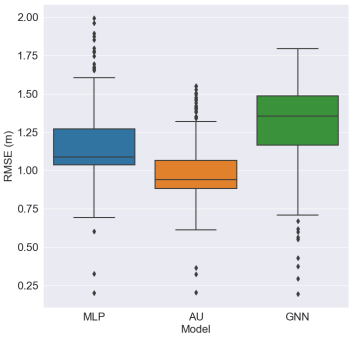

(a) RMSE over the nodes of the BAK water network.



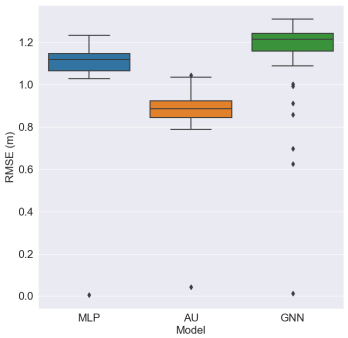

(b) RMSE over the nodes of the FOS water network.




(c) RMSE over the nodes of the PES water network.




(d) RMSE over the nodes of the MOD water network.




(e) RMSE over the nodes of the RUR water network.

Figure A.2: RMSEs over the nodes of the different water networks for each of the models with outliers.

# Bibliography


[1] L. A. Rossman, *EPANET 2: Users manual*. U.S. Environmental Protection Agency. Office of Research and Development. National Risk Management Research Laboratory, 2000.

[2] Y. Ishido and S. Takahashi, "A new indicator for real-time leak detection in water distribution networks: Design and simulation validation," *Procedia Engineering*, vol. 89, pp. 411–417, 2014, ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2014.11.206. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705814023212.

[3] D. Mora-Melia, P. L. Iglesias-Rey, F. J. Martinez-Solano, and P. Ballesteros-Pérez, "Efficiency of evolutionary algorithms in water network pipe sizing," *Water Resources Management*, vol. 29, pp. 4817–4831, 13 2015, ISSN: 1573-1650. DOI: 10.1007/s11269-015-1092-x. [Online]. Available: https://doi.org/10.1007/s11269-015-1092-x.

[4] D. Nowak, H. Krieg, and M. Bortz, "Surrogate models for the simulation of complex water supply networks," Jul. 2018.

[5] K. S. Tshehla, Y. Hamam, and A. M. Abu-Mahfouz, "State estimation in water distribution network: A review," 2017, pp. 1247–1252. DOI: 10.1109/INDIN.2017.8104953.

[6] A. Garzón, Z. Kapelan, J. Langeveld, and R. Taormina, "Machine learning-based surrogate modeling for urban water networks: Review and future research directions," *Water Resources Research*, vol. 58, 5 May 2022, ISSN: 19447973. DOI: 10.1029/2021WR031808.

[7] C. Wang, Q. Duan, W. Gong, A. Ye, Z. Di, and C. Miao, "An evaluation of adaptive surrogate modeling based optimization with two benchmark problems," *Environmental Modelling & Software*, vol. 60, pp. 167–179, 2014, ISSN: 1364-8152. DOI: https://doi.org/10.1016/j.envsoft.2014.05.026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364815214001698.

[8] S. Razavi, B. A. Tolson, and D. H. Burn, "Numerical assessment of metamodelling strategies in computationally intensive optimization," *Environmental Modelling and Software*, vol. 34, pp. 67–86, Jun. 2012, ISSN: 13648152. DOI: 10.1016/j.envsoft.2011.09.010.

[9] G. G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," *Journal of Mechanical Design*, vol. 129, pp. 370–380, 4 2007, ISSN: 10500472. DOI: 10.1115/1.2429697.

[10] "Dynamically dimensioned search algorithm for computationally efficient watershed model calibration," *Water Resources Research*, vol. 43, 1 Jan. 2007, ISSN: 00431397. DOI: 10.1029/2005WR004723.

[11] S. Razavi, B. A. Tolson, and D. H. Burn, "Review of surrogate modeling in water resources," *Water Resources Research*, vol. 48, 7 2012, ISSN: 00431397. DOI: 10.1029/2011WR011527.

[12] M. A. Andrade, C. Y. Choi, K. Lansey, and D. Jung, "Enhanced artificial neural networks estimating water quality constraints for the optimal water distribution systems design," *Journal of Water Resources Planning and Management*, vol. 142, no. 9, p. 04 016 024, 2016. DOI: 10.1061/(ASCE)WR.1943-5452.0000663.

[13] Yoon, Lee, and Jung, "Accelerated monte carlo analysis of flow-based system reliability through artificial neural network-based surrogate models.," *Smart Struct. Syst.*, vol. 26, no. 2, pp. 175–184, Aug. 2020. DOI: 10.12989/SSS.2020.26.2.175.

[14] H. I. Kim and K. Y. Han, "Urban flood prediction using deep neural network with data augmentation," *Water*, vol. 12, p. 899, 3 Mar. 2020, ISSN: 2073-4441. DOI: 10.3390/w12030899.

[15] G. Meirelles, D. Manzi, B. Brentan, T. Goulart, and E. Luvizotto, "Calibration model for water distribution network using pressures estimated by artificial neural networks," *Water Resources Management*, vol. 31, pp. 4339–4351, 13 Oct. 2017, ISSN: 15731650. DOI: 10.1007/S11269-017-1750-2/FIGURES/5.

[16] E. Anderson and K. Al-Jamal, "Hydraulic-network simplification," *Journal of Water Resources Planning and Management*, vol. 121, pp. 235–240, 3 May 1995. DOI: 10.1061/(ASCE)0733-9496(1995)121:3(235). [Online]. Available: https://doi.org/10.1061/(ASCE)0733-9496(1995)121:3(235).

[17] S. M. Kumar, S. Narasimhan, and S. M. Bhallamuda, "State estimation in water distribution networks using graph-theoretic reduction strategy," *Journal of Water Resources Planning and Management*, vol. 134, pp. 395–403, 5 2008. DOI: 10.1061/(ASCE)0733-9496(2008)134:5(395).

[18] A. Preis, A. J. Whittle, A. Ostfeld, and L. Perelman, "Efficient hydraulic state estimation technique using reduced models of urban water networks," *Journal of Water Resources Planning and Management*, vol. 137, pp. 343–351, 4 2011. DOI: 10.1061/(ASCE)WR.1943-5452.0000113.

[19] G. Lima, B. Brentan, D. Manzi, and E. Luvizotto, "Metamodel for nodal pressure estimation at near real-time in water distribution systems using artificial neural networks," *Journal of Hydroinformatics*, vol. 20, pp. 486–496, 2 2018. DOI: 10.2166/hydro.2017.036.

[20] B. Kerimov, R. Bentivoglio, A. Garzón, *et al.*, "Assessing the performances and transferability of graph neural network metamodels for water distribution systems," *Journal of Hydroinformatics*, jh2023031, Oct. 2023, ISSN: 1464-7141. DOI: 10.2166/hydro.2023.031. [Online]. Available: https://doi.org/10.2166/hydro.2023.031.

[21] P. Jacobs and I. C. Goulter, "Optimization of redundancy in water distribution networks using graph theoretic principles," *Engineering Optimization*, vol. 15, pp. 71–82, 1 1989. DOI: 10.1080/03052158908941143.

[22] F. Zheng, A. R. Simpson, A. C. Zecchin, and J. W. Deuerlein, "A graph decomposition-based approach for water distribution network optimization," *Water Resources Research*, vol. 49, pp. 2093–2109, 4 2013. DOI: https://doi.org/10.1002/wrcr.20175.

[23] K. V. K. Varma, S. Narasimhan, and S. M. Bhallamudi, "Optimal design of water distribution systems using an nlp method," *Journal of Environmental Engineering*, vol. 123, pp. 381–388, 4 1997. DOI: 10.1061/(ASCE)0733-9372(1997)123:4(381).

[24] I. Gupta, J. K. Bassin, A. Gupta, and P. Khanna, "Optimization of water distribution system," *Environmental Software*, vol. 8, pp. 101–113, 2 1993, ISSN: 0266-9838. DOI: https://doi.org/10.1016/0266-9838(93)90020-I.

[25] C. A. Bonilla, A. Zanfei, B. Brentan, I. Montalvo, and J. Izquierdo, "A digital twin of a water distribution system by using graph convolutional networks for pump speed-based state estimation," *Water (Switzerland)*, vol. 14, 4 2022. DOI: 10.3390/w14040514.

[26] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning: Key approaches and design guidelines," Institute of Electrical and Electronics Engineers Inc., Jun. 2021, ISBN: 9781665428255. DOI: 10.1109/DSLW51110.2021.9523403.

[27] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, 2010, pp. 399–406, ISBN: 9781605589077.

[28] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," Dec. 2020.

[29] L. Zhang, G. Wang, and G. B. Giannakis, "Real-time power system state estimation and forecasting via deep neural networks," Nov. 2018. DOI: 10.1109/TSP.2019.2926023.

[30] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," Dec. 2019.

[31] Y. Yang, J. Sun, H. Li, and Z. Xu, "Admm-csnet: A deep learning approach for image compressive sensing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 521–538, 3 Mar. 2020, ISSN: 19393539. DOI: 10.1109/TPAMI.2018.2883941.

[32] Y. Li, M. Tofighi, V. Monga, and Y. C. Eldar, "An algorithm unrolling approach to deep image deblurring," Feb. 2019. [Online]. Available: http://arxiv.org/abs/1902.05399.

[33] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," May 2018. DOI: 10.1109/TSP.2019.2899805. [Online]. Available: http://arxiv.org/abs/1805.07631%20http://dx.doi.org/10.1109/TSP.2019.2899805.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. arXiv: 1512.03385 [cs.CV].

[35] N. Trifunovic, *Introduction to urban water distribution*, 1st ed. Taylor & Francis/Balkema, 2006.

[36] E. Creaco, A. Campisano, N. Fontana, G. Marini, P. R. Page, and T. Walski, "Real time control of water distribution networks: A state-of-the-art review," *Water Research*, vol. 161, pp. 517–530, Sep. 2019, ISSN: 18792448. DOI: 10.1016/j.watres.2019.06.025.

[37] I.-S. Lorenz, L. C. Altherr, and P. F. Pelz, "Resilience enhancement of critical infrastructure – graph-theoretical resilience analysis of the water distribution system in the german city of darmstadt," pp. 137–149, 2021. DOI: 10.1007/978-3-030-64228-0_13.

[38] L. Mays, *Water transmission and distribution*. American Water Works Association, 2010.

[39] T. Anchieta, G. Meirelles, S. Carpitella, B. Brentan, and J. Izquierdo, "Water distribution network expansion: An evaluation from the perspective of complex networks and hydraulic criteria," *Journal of Hydroinformatics*, vol. 25, pp. 628–644, 3 May 2023, ISSN: 1464-7141. DOI: 10.2166/hydro.2023.080.

[40] E. Todini and L. A. Rossman, "Unified framework for deriving simultaneous equation algorithms for water distribution networks," *Journal of Hydraulic Engineering*, vol. 139, pp. 511–526, 5 May 2013, ISSN: 0733-9429. DOI: 10.1061/(asce)hy.1943-7900.0000703.

[41] K. Klise, D. Hart, D. Moriarty, *et al.*, "Water network tool for resilience (wntr) user manual," Sandia National Laboratories (SNL), Aug. 2017. DOI: 10.2172/1376816.

[42] E. Todini and S. Pilati, "A gradient algorithm for the analysis of pipe networks," in *Computer Applications in Water Supply: Vol. 1—Systems Analysis and Simulation*. GBR: Research Studies Press Ltd., 1988, pp. 1–20, ISBN: 0471917834.

[43] M. Collins, L. Cooper, R. Helgason, J. Kennington, and L. LeBlanc, "Solving the pipe network analysis problem using optimization techniques," *Management Science*, vol. 24, no. 7, pp. 747–760, 1978. DOI: 10.1287/mnsc.24.7.747.

[44] W. Bi, G. C. Dandy, and M. Asce, "Optimization of water distribution systems using online retrained metamodels," *Journal of Water Resources Planning and Management*, vol. 140, 11 Nov. 2014, ISSN: 0733-9496. DOI: 10.1061/(ASCE)WR.1943-5452.0000419.

[45] D. Meijer, J. Post, J. P. van der Hoek, H. Korving, J. Langeveld, and F. Clemens, "Identifying critical elements in drinking water distribution networks using graph theory," *Structure and Infrastructure Engineering*, vol. 17, pp. 347–360, 3 2021. DOI: 10.1080/15732479.2020.1751664.

[46] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, pp. 39–56, 1 May 2020. DOI: 10.1109/jsait.2020.2991563.

[47] L. Xing and L. Sela, "Graph neural networks for state estimation in water distribution systems: Application of supervised and semisupervised learning," *Journal of Water Resources Planning and Management*, vol. 148, 5 2022. DOI: 10.1061/(ASCE)WR.1943-5452.0001550.

[48] S. Nazif, M. Karamouz, M. Tabesh, and A. Moridi, "Pressure management model for urban water distribution networks," *Water Resources Management*, vol. 24, pp. 437–458, 3 Jan. 2010, ISSN: 09204741. DOI: 10.1007/s11269-009-9454-x.

[49] I. Jablonski, "Graph signal processing in applications to sensor networks, smart grids, and smart cities," *IEEE Sensors Journal*, vol. 17, pp. 7659–7666, 23 Dec. 2017, ISSN: 1530-437X. DOI: 10.1109/JSEN.2017.2733767.

[50] R. Ramakrishna and A. Scaglione, "Grid-graph signal processing (grid-gsp): A graph signal processing framework for the power grid," Mar. 2021. DOI: 10.1109/TSP.2021.3075145.

[51] X. Zhou, S. Liu, W. Xu, K. Xin, Y. Wu, and F. Meng, "Bridging hydraulics and graph signal processing: A new perspective to estimate water distribution network pressures," *Water Research*, vol. 217, 2022. DOI: 10.1016/j.watres.2022.118416.

[52] D. B. Barros, R. Gabriel, M. D. Souza, G. Meirelles, and B. M. Brentan, "Leak detection in water distribution networks based on graph signal processing of pressure data water demand time series generation for distribution network modelling and water demand forecasting view project smart water systems: Near-real time optimal operation and control view project," DOI: 10.4995/WDSA-CCWI2022.2022.14073. [Online]. Available: https://doi.org/10.4995/WDSA-CCWI2022.2022.14073.

[53] A. Altman and M. Tennenholtz, "Ranking systems: The pagerank axioms," in *Proceedings of the 6th ACM Conference on Electronic Commerce*, ser. EC '05, Vancouver, BC, Canada: Association for Computing Machinery, 2005, pp. 1–8, ISBN: 1595930493. DOI: 10.1145/1064009.1064010. [Online]. Available: https://doi.org/10.1145/1064009.1064010.

[54] L. Tsiami and C. Makropoulos, "Cyber—physical attack detection in water distribution systems with temporal graph convolutional neural networks," *Water (Switzerland)*, vol. 13, 9 2021. DOI: 10.3390/w13091247.

[55] G. Hajgató, B. Gyires-Tóth, and G. Paál, "Reconstructing nodal pressures in water distribution systems with graph neural networks," 2021. arXiv: 2104.13619 [cs.LG].

[56] A. Zanfei, A. Menapace, B. M. Brentan, R. Sitzenfrei, and M. Herrera, "Shall we always use hydraulic models? a graph neural network metamodel for water system calibration and uncertainty assessment," *Water Research*, vol. 242, p. 120 264, Aug. 2023, ISSN: 00431354. DOI: 10.1016/j.watres.2023.120264.

[57] N. Shlezinger, R. Fu, and Y. C. Eldar, "Deepsic: Deep soft interference cancellation for multiuser mimo detection," 2020. arXiv: 2002.03214 [eess.SP].

[58] M. Kocaoglu, C. Snyder, A. G. Dimakis, and S. Vishwanath, "Causalgan: Learning causal implicit generative models with adversarial training," 2017. arXiv: 1709.02023 [cs.LG].

[59] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed sensing using generative models," 2017. arXiv: 1703.03208 [stat.ML].

[60] E. K. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," 2019. arXiv: 1905.05406 [cs.CV].

[61] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Viterbinet: A deep learning based viterbi algorithm for symbol detection," 2020. arXiv: 1905.10750 [cs.LG].

[62] N. Farsad and A. Goldsmith, "Neural network detection of data sequences in communication systems," Jan. 2018. DOI: 10.1109/TSP.2018.2868322.

[63] W. Pu, C. Zhou, Y. C. Eldar, and M. R. D. Rodrigues, "Rest: Robust learned shrinkage-thresholding network taming inverse problems with model mismatch," IEEE, Jun. 2021, pp. 2885–2889, ISBN: 978-1-7281-7605-5. DOI: 10.1109/ICASSP39728.2021.9414141.

[64] K. Pratik, B. D. Rao, and M. Welling, "Re-mimo: Recurrent and permutation equivariant neural mimo detection," Jun. 2020. DOI: 10.1109/TSP.2020.3045199.

[65] V. G. Satorras, Z. Akata, and M. Welling, "Combining generative and discriminative models for hybrid inference," 2019. arXiv: 1906.02547 [stat.ML].

[66] D. R. Broad, ; H. R. Maier, G. C. Dandy, and M. Asce, "Optimal operation of complex water distribution systems using metamodels," 2010. DOI: 10.1061/ASCEWR.1943-5452.0000052.

[67] E. Salomons, A. Goryashko, U. Shamir, Z. Rao, and S. Alvisi, "Optimizing the operation of the haifa-a water-distribution network," *Journal of Hydroinformatics*, vol. 9, pp. 51–64, 1 2007, ISSN: 14647141. DOI: 10.2166/hydro.2006.017.

[68] F. Martínez, V. Hernández, J. M. Alonso, Z. Rao, and S. Alvisi, "Optimizing the operation of the valencia water-distribution network," *Journal of Hydroinformatics*, vol. 9, pp. 65–78, 1 2007, ISSN: 14647141. DOI: 10.2166/hydro.2006.018.

[69] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," Jun. 2016. [Online]. Available: http://arxiv.org/abs/1606.09375.

[70] A. Paszke, S. Gross, F. Massa, *et al.*, *Pytorch: An imperative style, high-performance deep learning library*, 2019. arXiv: 1912.01703 [cs.LG].

[71] M. Fey and J. E. Lenssen, *Fast graph representation learning with pytorch geometric*, 2019. arXiv: 1903.02428 [cs.LG].

[72] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[73] K. Klise, R. Murray, and T. Haxton, "An overview of the water network tool for resilience (wntr)," Jul. 2018.

[74] S.-C. Lee and S.-I. Lee, "Genetic algorithms for optimal augmentation of water distribution networks," *Journal of Korea Water Resources Association*, vol. 34, Jan. 2001.

[75] C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi, and P. Toth, "On the optimal design of water distribution networks: A practical minlp approach," *Optimization and Engineering*, vol. 13, pp. 219–246, 2 Jun. 2012, ISSN: 13894420. DOI: 10.1007/s11081-011-9141-7.

[76] G. Dandy, A. Wilkins, and H. Rohrlach, "A methodology for comparing evolutionary algorithms for optimising water distribution systems," vol. 140, Dec. 2011, pp. 786–798, ISBN: 978-0-7844-1203-9. DOI: 10.1061/41203(425)73.

[77] L. Rossman, "Storm water management model reference manual volume ii-hydraulics epa/600/r-17/111," 2017. [Online]. Available: www.epa.gov/water-research.