Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Kristian Lien Studsrød

Parameter Identification in Water Distribution Networks for Leak Detection

Master's thesis in Industrial Cybernetics
Supervisor: Ole Morten Aamo
Co-supervisor: Nils Christian Aars Wilhelms
June 2024

NTNU
Norwegian University of
Science and Technology

Kristian Lien Studsrød

Parameter Identification in Water Distribution Networks for Leak Detection

Master's thesis in Industrial Cybernetics
Supervisor: Ole Morten Aamo
Co-supervisor: Nils Christian Aars Wilhelms
June 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Preface

This master's thesis is part of the course TTK4900 *Engineering Cybernetics, Master Thesis*. The course rewards the equivalent of 30 ECTS points and is the final project in the 2-year master's program of *Industrial Cybernetics* at the Norwegian University of Science and Technology (NTNU).

The author expresses his sincere gratitude to supervisor, Professor Ole Morten Aamo and co-supervisor Postdoc Nils Christian Aars Wilhelmsen, for their support and guidance. Their expertise and feedback were invaluable.

In addition, special thanks go out to Jomar Sommerschild, Kjetil Isdal, and other representatives from Orkanger Municipal for their great cooperation and for providing insight and data from their network.

Kristian Lien Studsrød
June 2024, Trondheim

Abstract

This master's thesis focuses on identifying friction parameters in tree-shaped networks, specifically water distribution networks. An algorithm is developed based on analyzing network trees with available measurements in the inlet and outlet only. The algorithm imports network files and returns matrices that are used to calculate friction parameters. A simplified flow model is used to derive parameter identification equations. Graph-theory techniques such as breath first search and shortest path are used to produce friction parameter system matrices. The results are successfully verified with simulations. Requirements for the network are that the degree of internal nodes are larger then two, and that the network does not include cycles, these criteria are suggested to be included in further work.

Additionally, the thesis explores the water network files of Orkanger Municipality to identify opportunities to deploy an algorithm for calculating water friction parameters.

Finally, a leak detection algorithm based on an Adaptive Observer and Continuous Extended Kalman Filter is recreated and tuned. The leak detection algorithm is prepared to be used on a section of the water network from Orkanger Municipality.

Sammendrag

Denne masteroppgaven fokuserer på å identifisere friksjonsparametere i treformede nettverk, særlig vannfordelingsnettverk. En algoritme utvikles basert på analyse av nettverkstrær med målinger kun ved innløp og utløp. Algoritmen importerer nettverksfiler og returnerer matriser som brukes til å beregne friksjonsparametere. En forenklet strømningsmodell brukes for å utlede parameteridentifikasjonsligningene, grafteoriteknikker som *bredde-først-søk* og *korteste vei* brukes for å produsere systemmatrisene. Resultatene verifises med simuleringer. Krav til nettverket er at graden til interne noder er større enn to, og at nettverket ikke inneholder sykler; disse kriteriene foreslås å håndteres i fremtidig arbeid.

I tillegg utforsker oppgaven vannnettverksfilene til Orkanger kommune for å identifisere muligheten for å implementere en algoritme for beregning av friksjonsparametere.

Til slutt blir en lekkasjedeteksjonsalgoritme basert på en adaptiv estimator og et Kalman filter implementert og justert på data. Lekkasjedeteksjonsalgoritmen forberedes til bruk på en rør-seksjon fra vannnettverket til Orkanger kommune.

Contents

Figures

Tables

Chapter 1

Introduction

This master's project will focus on friction parameter identification in water networks. Water networks in Norway have among the highest levels of loss percentage in Europe, where some water networks lost 40 % or more of the cleaned drinking water [1]. Norwegian Institute of Public Health (FHI) estimated in 2019, that from a total drinking water production of 770 million m^3, approximately 30 % was lost due to leaks in water networks, which is equivalent to 210 million m^3 [2]. One reason for the high loss percentage is the abundant water supply, with minimal water treatment costs [3].

Access to water is one of the most basic human needs for health [4]. Fortunately, due to increased focus on cost and sustainability, the European Commission has published good practices for leakage management [5] to improve the efficiency of the use of water resources. In addition, the United Nations (UN) sustainability goal 6 works to ensure access to water and sanitation for all [4]. This is why the UN encourages governments to invest in water research and development. [4]

Leak detection and techniques to locate the position of leaks are researched topics, and many methods exist. Adegboye *et al.* [6] lists different leak detection methods used in oil and gas pipelines; some methods can be used for both water networks and oil pipelines. Adegboye divides the methods into exterior methods, visual methods and internal methods. Exterior methods mainly use sensors on the external parts of the pipelines that can detect the leaked fluid in the environment around the pipeline. Visual methods include humans, dogs or drones to detect leaks visually, by inspection. The last category, internal or computational methods, includes measurement instruments internally in the pipeline to monitor parameters such as flow and pressure. The measured parameters are used together with analytics to determine leakage.

This master project will use computational methods to identify leaks and a steady-state approach to determine network friction parameters.

1.1 Objective

The objectives of the master's project is to perform parameter identification of water distribution networks. It will be divided into sub-objectives:

- Investigate how friction parameters can be estimated with a minimized number of available measurements in different tree-shaped networks.
- Given a tree-shaped network, design and implement an algorithm to identify and return estimates of friction parameters of said network.
- Investigate how Orkanger Municipality network files can be used to find trees applicable for friction parameter identification.
- Implement, and test leak the detection algorithm from Carrera and Verde [7] on data from Orkanger Municipality.

1.2 Report Structure

This master's thesis will consist of several chapters describing the work conducted. The introduction will cover the background and objectives of the thesis. Subsequently, a chapter with a literature review will present several methods used in the literature for parameter identification and leak detection.

The theory chapter provides a theoretical foundation for the subsequent chapters. It introduces the general theory of linear algebra, partial differential equations, flow models, and conservation laws. It also briefly introduces adaptive observers and Kalman filters. The last sections of the theory chapter cover graph theory and Epanet, which is a network modelling tool.

The method is divided into three chapters. The first chapter consists of a deduction of various tree networks of pipes and junctions; from simple networks to more complicated ones. Each network is analysed in a steady state to deduce parameter equations that can be organised in system matrices. The purpose is to find a generalized setup of system matrices that can be used to identify the friction parameter in each pipe. The generalized setup shall be such that it can be automated with an algorithm.

Following the deduction of system matrices for a general network, the next chapter will consist of the method to automate the setup of system equations for an imported network tree. This chapter uses the graph theory introduced in the theory to develop an algorithm that can be used to import a general network, and return a set of matrices. In addition, the preparation for the import of the water network file supplied by Orkanger Municipality is explained.

The last method chapter will go through a the leak detection algorithm from Carrera and Verde [7]. The leak detection algorithm uses an adaptive observer to determine the friction parameter for a section of pipe. When a leak is detected, a Continuous Extended Kalman filter is used to determine the position of the leak, based on the friction factor from the adaptive observer. Data from the experiment in [7] is used to tune and test the leak detection algorithm. The chapter will also include an introduction to a section of Orkanger Municipality water network that will be used to verify the performance of the leak detection algorithm.

The result chapter is divided into three main sections. The first section comprises the results

from the steady-state analysis; different networks are imported to produce system matrices for the friction parameter and the friction factors are calculated. Simulation tools are used to verify the results. The second section gives the results from Orkanger Municipality water network analysis. The final section comprises the results from the leak detection algorithm, it is tuned and tested on data.

The last two chapters are the discussion and the conclusion.

Chapter 2

Literature Review

Billmann and Isermann [8] proposed a leak detection method for pipelines based on nonlinear adaptive state observer. The method was primarily intended for use in transporting liquids and gas in pipelines with measurements available at inlet and outlet. Their method uses a dynamic model with an adaptive state observer and estimation of the friction coefficient with the least-square method. This method is intended for a single leak.

Other more advanced techniques have come later, with the possibility of detecting several leaks, such as frequency analysis and prediction error model by Torres *et al.* [9]. This model requires excitation with a persistent input. Torres *et al.* [9] showed that leak parameters can be identified at low frequencies.

Verde [10] designed and tested a multi-leak detection method by discretising the space of the pipeline and evaluating residuals of several observers; together with a logic detection function, multiple leaks can be detected and localised.

Another observers-based leak detection method is proposed by Aamo *et al.* [11], where an adaptive observer is based on nonlinear hyperbolic partial differential equations with boundary injection.

Aamo [12] proposed an adaptive observer with backstepping transformation, for a single pipeline with measurement available at inlet and outlet. The adaptive observer design was further developed by Anfinsen and Aamo [13] to include a branched pipe. The adaptive observer for the branched pipe can estimate the total leak size and locate any number of point leaks as long as they are sufficiently separated in time.

Wilhelmsen and Aamo [14] developed an adaptive observer, to complement on [12] and [13], to include for loops in networks. With measurements at the junctions only, the observer estimates the total leak size and position of any leak as long as the leaks are sufficiently spaced in time.

Wilhelmsen and Aamo [15] further developed the adaptive observer in [13] to extend the model for $N + 1$ branched pipe flows with a recursive procedure to find solutions to kernel equations, where the kernel equations are used to update observer gains. Wilhelmsen and Aamo [16] developed the design in [14] by constructing explicit solutions for the backstepping kernel equations used to calculate observer gains.

An Extended Kalman filter is another method of leak detection. Lesyshen [17] uses Extended Kalman filters to estimate states within the leakage by placing two artificial leak-states within the model. The Extended Kalman filter will then estimate the position and magnitude of one leak. Oven [18] used a model-based leak detection method with a Kalman filter where the transmission line included a branch and two outlets.

Delgado-Aguiñaga *et al.* [19] used a Kalman filter based on a Linear Parameter Varying System to avoid linearisation normally required by the Extended Kalman Filter. Like other methods, flow and pressure measurements at the ends of the pipeline are needed. The Linear Parameter varying model is derived from a nonlinear model.

Carrera and Verde [7] combined the use of an adaptive observer to determine the unknown friction parameters, with an Continuous Extended Kalman filter to determine leak location in a pipeline Carrera and Verde [7].

Chapter 3

Theory

This chapter introduces fundamental concepts, theories, and mathematical tools essential for understanding and developing leak detection applications. It introduces partial differential equations, finite difference methods, numerical differentiation, transient flow models, and flow characteristics, which form the theoretical foundation for the practical applications discussed in later chapters. The last part will introduce graph theory to be used to explore network trees, and the network modelling and simulation tool Epanet will be introduced.

3.1 Partial Differential Equations

Partial differential equations or PDE is a group of mathematical equations that include a function of two or more independent variables, and the partial derivatives with respect to the independent variables [20]. PDEs are used to formulate mathematical equations for physical problems such as heat, fluid flow, and elasticity, where several variables are required to describe the behaviour.

3.1.1 Finite Difference method

One method to solve PDEs is the *Finite difference* methods [21]. They often consist of several steps to complete the solution, including:

- Discretization of domain
- Replace derivatives with approximations
- Replace exact solutions with approximations
- Solving the system

3.1.2 Numerical Differentiation

Given a "smooth" function $f(x)$, meaning the function has continuous derivatives over some domain [22], the derivative of f is defined as:

$$f^{'}(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{3.1}$$

where h is the step size. By removing the limit and setting h as a fixed step, we get the forward difference method [21].

$$f^{'}(x) \approx \frac{f(x+h)-f(x)}{h} \tag{3.2}$$

And when applied to a PDE function $\alpha(z,t)$ with $h = \Delta z$ the resulting approximations is:

$$\frac{\partial \alpha(z,t)}{\partial z} \approx \frac{\alpha_{z_{k+1}}(t)-\alpha_{z_k}(t)}{\Delta z_k} \tag{3.3}$$

where z_k is the starting point and z_{k+1} is the next point.

3.2 Transient Flow Models

The transient flow models are constructed using rules of conservation of both mass and momentum [23, p. 16]. These rules/equations will be presented in this sub-chapter based on equations derived in [24].

3.2.1 Flow Characteristics

There are a number of ways to characterize flow, with given conditions; some important terms are listed below [24, p. 2].

- Steady flow
  - Pressure and velocity at a point are constant.
- Unsteady flow
  - Pressure and velocity at a point change over time.
- Transient flow
  - The conditions that occur when the flow changes from one steady state to another.

3.2.2 Continuity Equation

The continuity equation is derived with the law of conservation of mass to a given control volume [24, pp. 39–42]. It is derived with respect to slightly compressible fluid in a conduit/pipe with linear elastic walls. The continuity equations are given as:

$$\frac{\partial p}{\partial t} + V\frac{\partial p}{\partial x} + \rho\, a^2\, \frac{\partial V}{\partial x} = 0 \tag{3.4}$$

where V is flow velocity, t is time, x is distance, p is pressure, and ρ is the water density. In addition, a^2 is defined as:

$$a^2 = \frac{\frac{K}{\rho}}{1+\frac{D\,K}{e\,E}} \tag{3.5}$$

where $K = \frac{dp}{d\rho/\rho}$, is the bulk modulus of elasticity of fluid, D is diameter of circular conduit/pipe, e the thickness of conduit/pipe walls and E is the E-modulus of conduit/ pipe.

3.2.3 Momentum Equation

The momentum equation is derived with the application of Reynolds Transport Theorem, Newton's second law of motion and Darcy-Weisbach friction equation [24, pp. 43–45]. The momentum equation is given as:

$$\frac{\partial V}{\partial t} + V\,\frac{\partial V}{\partial x} + \frac{1}{\rho}\,\frac{\partial p}{\partial x} + g\,sin\theta + \frac{f\;V\;|V|}{2\;D} \;=\; 0 \tag{3.6}$$

where V is flow velocity, t is time, x is distance, p is pressure, and $\rho$is the water density. In addition, f is defined as the Darcy-Weisbach friction factor.

3.2.4 Simplified Equations

Both the continuity equation Equation (3.4) and the momentum equation Equation (3.6) can be simplified. The acceleration terms ($V\frac{\partial p}{\partial x}$ and $V\frac{\partial V}{\partial x}$) are small compared to other terms and can therefore be neglected [24, p. 48]. Likewise, the slope term $sin\theta$ is often small and can be neglected. Another common practice in hydraulic engineering is to rewrite flow velocity V as volumetric flow Q, these two variables are related through:

$$Q = VA \tag{3.7}$$

where A is the area of the cross-section. Similarly, the pressure variable p is replaced with piezometric head H, these two variables are related through:

$$p = \rho g(H - h) \tag{3.8}$$

where h is the relative elevation. By taking Equation (3.7), Equation (3.8) and the simplification stated above into Equation (3.4) and bringing along Equation (3.6) the following equations is obtained:

$$g\,A\,\frac{\partial H}{\partial x} + \frac{\partial Q}{\partial t} + \frac{f\;Q\;|Q|}{2\;D\;A} = 0 \tag{3.9a}$$

$$\frac{\partial H}{\partial t} + \frac{a^2}{g\,A}\frac{\partial Q}{\partial x} = 0 \tag{3.9b}$$

where a is the wave velocity. The set of simplified Equation (3.9) are known as semi-linear, hyperbolic, partial differential equations [24, p. 59].

3.2.5 Steady State

If one is to assume that there are no changes in piezometric head and volumetric flow, i.e. Equation (3.9) are at steady state. Thus $\frac{\partial H}{\partial t} = 0$ and $\frac{\partial Q}{\partial t} = 0$, it yields:

$$g\,A\,\frac{\partial H}{\partial x} + \frac{f\;Q\;|Q|}{2\;D\;A} = 0 \tag{3.10a}$$

$$\frac{a^2}{g\,A}\frac{\partial Q}{\partial x} = 0 \tag{3.10b}$$

Assessing Equation (3.10a) and approximating the derivatives with Equation (3.3), and isolating for ΔH one is left with the Darcy-Weisbach equation [24, p. 49]:

$$\Delta H = \frac{f \ \Delta x \ Q^2}{2 \ g \ D \ A^2} \tag{3.11}$$

where ΔH is the difference of piezometric head over a pipe with length Δx, also called $J(Q(z,t))$ [25], and f is a friction function. The second part of Equation (3.9) yields that there is no change in flow Q over the distance of the pipe [24, p. 49].

$$\frac{\partial Q}{\partial x} = 0 \tag{3.12}$$

3.3 Linear Algebra

Modelling mathematical problems or equations describing reality often leads to having simultaneous linear equations. This system of equations can be set up as matrices and solved with numerical linear algebra [26, p. 19]. An example of a set of equations with n known relationship is:

$$a_{i,1}x_1 + a_{i,2}x_2 + ... + a_{i,n}x_n = b_i \qquad \text{for} \qquad i = 1, 2, ..., n \tag{3.13}$$

where $a_{i,n}$ are coefficients for unknowns x_n. Having n linear equations for n unknowns x_j, where $j = 1, 2, ..., n$, the problem can be rewritten to the matrix form as:

$$Ax = b \tag{3.14}$$

where all coefficients $a_{i,j}$ have been gathered in the matrix A. When the size of A is $n \times n$, it is considered a square matrix [27, p. 277]. A linear system is considered *consistent* if one or more solutions exist, or *inconsistent* if no solutions exist [27, p. 277]. Conditions for solving $Ax = b$ is that A has full rank or A is non-singular[26, p. 19], i.e:

$$\text{rank}(A) = n, \quad A^{n \times n}, \quad n \in \mathbb{R} \tag{3.15}$$

meaning the inverse of A exists. For a number $k < n$, where rank(A) = k, A is singular. When having a non-singular matrix A, the solution to Equation (3.14) will be:

$$x = A^{-1}b \tag{3.16}$$

3.3.1 Linear Least Squares

For systems that have $m > n$ conditions, i.e. more equations than unknowns, the matrix A will be rectangular ($m \times n$) [27, p. 258] and the system is *overdetermined*. It may not be possible to determine a unique solution x that solves Equation (3.14). One way of solving inconsistent systems is by choosing $\bar{x}$ that minimizes the error $E = ||Ax - b||$ [28, p. 154], also called the *least squares* solution of x. [26, p. 22]. One way of finding least square solution x is by multiplying Equation (3.14) with A^T, giving:

$$A^T A\bar{x} = A^T b \tag{3.17}$$

where A^TA will return a square, symmetric coefficient matrix [28, p. 155]. Equation (3.17) is called the *normal equation* in statistics. If:

$$\text{rank}(A) = n \tag{3.18}$$

then columns of A is linear independent, then A^TA is non-singular. The solution of the normal equation is:

$$\bar{x} = (A^TA)^{-1}A^Tb \tag{3.19}$$

3.4 Adaptive Observer

Modelling, monitoring or controlling a system requires information about the said system. That information can come from measurements from sensors or from known parameters such as lengths and masses. However, not all information is available, and unmeasurable disturbances are unknown; cost or technical constraints may be causes for other unknowns in a system [29, p. 1]. This is what the *Observer* is for, by re-constructing information not directly available. This section will focus on *Adaptive Observers*, which is finding an adaption law to have a convergence of the observer when parameters are unknown [29, p. 211]. The system is described with a state-space model:

$$\begin{aligned} \dot{x}(t) &= f\left(\Theta, x(t), u(t), t\right) + g(x(t), u(t))\,\Theta \\ y(t) &= h(x(t)) \end{aligned} \tag{3.20}$$

where x is the states, y measurements, u inputs and Θ the unknown parameters is assumed to be constant. f, g and h are non-linear functions. However, the system can be transformed to the canonical observer form if Θ is assumed known [29, p. 217]:

$$\begin{aligned} \dot{y} &= \alpha(y, \zeta, \nu) + \beta(y, \zeta, \nu)\Theta \\ \dot{\zeta} &= Z(y, \zeta, \nu) \end{aligned} \tag{3.21}$$

G. Besancon [29, p. 217] has proposed an adaptive state observer for Equation (3.21) where $||\hat{y} - y|| \to 0$ and $||\hat{\zeta} - \zeta|| \to 0$ as $t \to \infty$:

$$\begin{aligned} \dot{\hat{y}} &= \alpha(y, \hat{\zeta}, \nu) + \beta(y, \zeta, \nu)\hat{\Theta} - k_y(\hat{y} - y) \\ \dot{\hat{\zeta}} &= Z(y, \hat{\zeta}, \nu) \\ \dot{\hat{\Theta}} &= -k_\Theta \beta^T(y, \zeta, \nu)(\hat{y} - y)^T \end{aligned} \tag{3.22}$$

where $k_y > 0$, $k_\Theta > 0$ and where $||\hat{\Theta} - \Theta|| \to 0$ if β is persistently exciting and $\dot{\beta}$ is bounded.

3.5 Kalman Filter

The *Kalman filter* is an algorithm to estimate the state of a linear, or non-linear system [30]. The filter is named after publications of R.E Kalman [31] and is an efficient recursive filter that can reconstruct unmeasured states and/or remove noise from measurements of estimates of the states [32, p. 408]. In general, the Kalman filter estimates states of a system ($\hat{x}$) in discrete time, converging the estimates to the state (x) by means of the measurements (y). The discretized linear system model is used to define the discrete-time Kalman filter:

$$x[k+1] = A_d x[k] + B_d u[k] + E_d w[k] \tag{3.23a}$$
$$y[k] = C_d x[k] + D_d u[k] + \epsilon[k] \tag{3.23b}$$

where w is the zero-mean Gaussian process noise, and ϵ is the zero-mean Gaussian measurement noise [32, p. 409]. x are the states and y the measurements. The system matrices A, B, E, C and D are assumed constant. However, they might change with each time step. The equations of the algorithm can be split into two groups: time equations and measurement update equations [30, p. 4]. Time update equations are *predicting* into the future to obtain an estimate of the states and error covariance, while the measurement update equations are *correting* the estimates by means of the measurements. The time update equations (predictor) are:

$$\hat{x}^-[k+1] = A_d \hat{x}_k + B_d u[k] \tag{3.24a}$$
$$P^-[k+1] = A_d P[k] A_d^T + E_d Q E^T \tag{3.24b}$$

where $\hat{x}^-$ and P^- are *priori* estimate of state and estimate of error covariance. $\hat{x}$ and P is *posteriori* state estimate and estimate of error covariance. The measurement equations are:

$$K[k] = P^-[k+1] C_d^T \left(C_d P^-[k+1] C_d^T + R_d \right)^{-1} \tag{3.25a}$$
$$\hat{x}[k] = \hat{x}^-[k] + K[k](y[k] - C_d \hat{x}^-[k] - D_d u[k]) \tag{3.25b}$$
$$P[k] = (I - K[k] C_d) P^-[k] (I - K[k] C_d)^T + K[k] R_d K^T[k] \tag{3.25c}$$

where Q_d is covariance matrices for process noise and R_d is covariance matrices for measurement noise. K matrix is referred to as Kalman Gain. Measurement noise matrix R is usually measured before implementation, while the process noise matrix Q is chosen by tuning [30, p. 6]. The initialization of the Kalman filter is not described in this report.

3.5.1 Extended Kalman Filter

For non-linear systems, the *Extended Kalman filter* or EKF has to be applied. A non-linear system in discrete time is described as:

$$x[k+1] = x[k] + hf(x[k], u[k], 0) \tag{3.26a}$$
$$y[k] = h(x[k], u[k]) \tag{3.26b}$$

where $f(x, u, w)$ and $u(x, u)$ are non-linear functions. w and ϵ are white Gaussian process noise and white Gaussian measurement noise respectively. The EKF is a Kalman filter linearised about

the current estimate and covariance [30, p. 7]. Similarly to the Kalman filter, the equations for the extended Kalman filter algorithm can be grouped into time equations [32, p. 412]:

$$\hat{x}^{-}[k+1] = \hat{x} + hf(\hat{x}[k], u[k], 0) \tag{3.27a}$$
$$\hat{P}^{-}[k+1] = A_d[k]\hat{P}A_d^T[k] + E_d[k]Q_d[k]E_d^T[k] \tag{3.27b}$$

and measurement equations

$$K[k] = \hat{P}^{-}[k]C_d^T[k]\left(C_d\hat{P}^{-}[k]C_d^T + R_d[k]\right)^{-1} \tag{3.28a}$$
$$\hat{x}[k] = \hat{x}^{-}[k] + K[k](y[k] - h[k](\hat{x}-[k], u[k])) \tag{3.28b}$$
$$\hat{P}[k] = (I_n - K[k]C_d[k])\,\hat{P}^{-}[k]\,(I_n - K[k]C_d[k])^T + K[k]R_d[k]K^T[k] \tag{3.28c}$$

where the *Jacobians* are used to define the discrete-time system matrices:

$$A_d[k] = I_n + h\frac{\partial f(x[k], u[k], w[k])}{\partial x[k]}\bigg|_{x[k]=\hat{x}[k], w[k]=0} \tag{3.29a}$$
$$E_d[k] = \frac{\partial f(x[k], u[k], w[k])}{\partial x[k]}\bigg|_{x[k]=\hat{x}[k], w[k]=0} \tag{3.29b}$$
$$C_d[k] = \frac{\partial h(x[k], u[k])}{\partial x[k]}\bigg|_{x[k]=\hat{x}[k]} \tag{3.29c}$$

where I_n is the identity matrix of size n. While it is easy to establish stability and convergence for the Kalman filter, the EKF loses those properties due to the linearization [32, p. 412]. Any modelling inaccuracies may result in divergence. In some applications, convergence of the EKF may only be possible if the function f and h are weakly non-linear or have initialization close to the solution [33].

3.5.2 Continuous Extended Kalman Filter with Increased Convergence Domain

Similarly to the discrete-time extended Kalman filter, the Continuous Extended Kalman filter has problems with divergence [33]. There have been developed models with increased convergence domain of the estimation. Reif *et al.* [33] developed such a model based on a non-linear system:

$$\dot{x}(t) = f(x(t), u(t)) \tag{3.30a}$$
$$y(t) = h(x(t), u(t)) \tag{3.30b}$$

where x is the state, u input, y measurements. Both f and h are non-linear functions. With the following observer model:

$$\dot{\hat{x}} = f(\hat{x}(t), u(t)) + K(t)[y(t) - h\hat{x}(t)] \tag{3.31}$$

where K is the Kalman gain and $\hat{x}$ the state estimates. A Riccati differential equation is introduced to calculate gain:

$$\dot{P}(t) = (A(t) + \eta I)\, P(t) + P(T)\left(A^T(t) + \eta I\right) - P(t)C^T(t)R^{-1}C(t)P(t) + Q \tag{3.32a}$$

$$K(t) = P(t)C^T(t)R^{-1} \tag{3.32b}$$

where Q and R are positive definite matrices for noise covarince matrices, and $\eta > 0$. Matrices A and C is the jacobian of f and h:

$$A(t) = \frac{\partial f(\hat{x}(t), u(t))}{\partial x} \tag{3.33a}$$

$$C(t) = \frac{\partial h(\hat{x}(t))}{\partial x} \tag{3.33b}$$

3.6 Graph Theory

A graph is a structure to encode pairwise relationships among a set of objects [34, p. 73]. A graph comprises of nodes (or vertex) V and edges E. Nodes are connected by edges. A simple graph G is defined as [35, p. 148]:

- $G = (V, E)$ where:
  - V is a finite set, called vertices of G
  - E is a two-element subset of V

The edge $e \in E$ is represented as a subset of $V : e = \{u, v\}$ for some $u, v \in V$ where u and v are the ends of e. Graphs may be *directed* or *undirected*. In a *directed* graph, G', the graph consists of a set of nodes V and a set of *directed* edges E'. Each edge $e' = (u, v)$ in E is *ordered*, changing of e' to $e' = (v, u)$ will produce a different graph. [34, p. 73]. *Directed* graphs are used to encode asymmetric relationships between nodes, while *undirected* graphs represent symmetric relationships between the nodes [34].

Nodes and edges may correspond to physical objects, or, in other cases, one or both are abstractions [34, p. 74]. An example of a physical object graph is cities and roads, where cities are the nodes and roads are the edges. An example of a graph with abstractions is a social network, where people are the nodes and the relationships are the edges.

Edges between nodes may be *weighted* or *unweighted*, where the weight can represent, for example, a cost of transportation between two nodes or a distance between two cities [35, p. 177].

One of the key operations of graphs is finding *paths* from node to node, by traversing the nodes connected by edges [34, p. 76]. An example of a path is how to travel from one city to another by using the roads and visiting interconnecting cities on the route. Paths are denoted P and can either be a sequence of nodes, or edges (edgepath). If a path in a graph starts and ends at the same location, it is said that the graph contains a *cycle* [34, p. 76]. By analysing the Figure 3.1, the path $P = \{1, 2, 4, 3, 1\}$ forms a cycle.

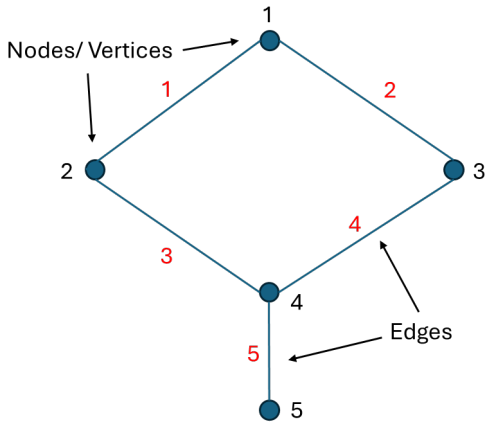


Figure 3.1: Example of undirected, unweighted graph G with a cycle.

The graph G in Figure 3.1 can be represented with nodes $V = \{1, 2, 3, 4, 5\}$ and edges $E =$

$\{1,2,3,4,5\}$. Where the edges $e \in E$ can be represented as a two-element subset of V such that the graph G can be written as:

$$G = \begin{Bmatrix} 1 & 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 4 & 5 \end{Bmatrix} \tag{3.34}$$

The *degree* of the node is another parameter to note. The *degree* is a number of how many edges are connected to a specific node [35, p. 150]. In Figure 3.1, the degree of node 1 is two. It is denoted as:

$$d(1) = 2 \tag{3.35}$$

3.6.1 Trees

A *tree* is a special form of undirected graph which does not contain cycles and is *connected* [34, p. 77]. A graph is *connected* if there is a path from u to v for every pair of nodes u and v. Trees are often *rooted* in a specific node, such that all other nodes and edges are moving outward from the root. In Figure 3.2 the root is in node 1. If a graph has one or more trees, it is defined as a *forest* [35, p. 172].

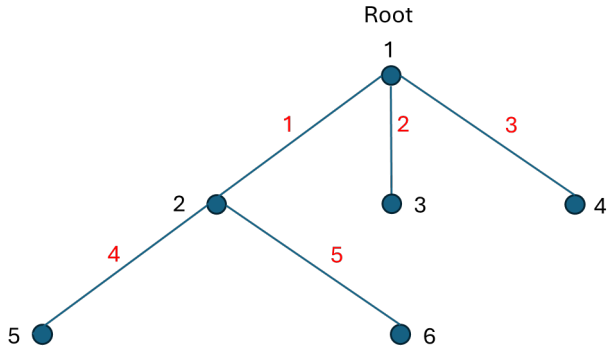


Figure 3.2: Example of tree, rooted in node 1.

Given the rooted three in Figure 3.2 some terminology can be introduced. In the path $P = \{1,2,5\}$ node 2 and 5 are *children* of node 1. Node 2 is a *parent* to node 5, and a *child* of node 1. If a node has no children it is a *leaf* [35, p. 173]. The tree used in the example above consists of a root and 2 levels, or layers, where nodes $V = \{2,3,4\}$ make up the first layer, and $V = \{5,6\}$ makes up the second layer.

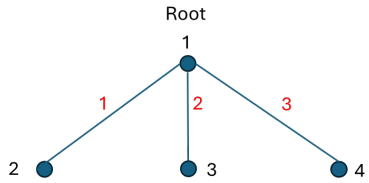
3.6.2 Shortest Path between Two Single Nodes

The problem of finding the shortest path from node u to node v can be solved with different types of algorithms dependent on the type of graph [36]. For directed and undirected graphs with unweighted edges, without cycles, *breadth first search* or *BFS* can be used to find the shortest path. The BFS algorithm searches the graph from the starting node s. It inspects all neighbour nodes, and for each of the neighbour nodes, it visits all unvisited neighbours. The algorithm does not stop until all reachable nodes have been visited [37].

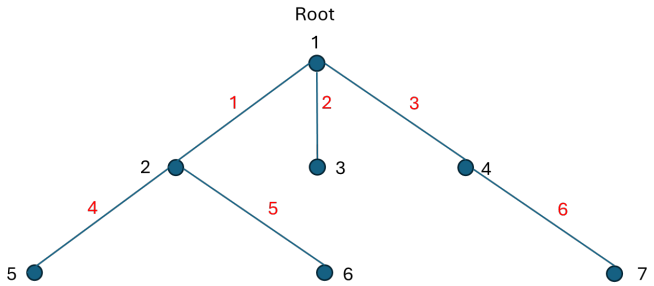
Algorithm 1: Breadth First Search

```
Input: Start node s
Output: Order in which nodes are discovered and finished
Function BFS(s):
    startnode ← s;
    discovernode ← s;
    Initialize nodelist as a queue with s as its only element;
    while nodelist is not empty do
        get C from nodelist;
        forall outgoing edges E from node C, connecting to node V do
            if V is a new node then
                mark V as discovered;
                append V to nodelist;
            else if V is a previously discovered node then
                mark V as discovered;
            else if V is a finished node then
                do nothing;
            end
        end
        mark C as finished;
    end
```

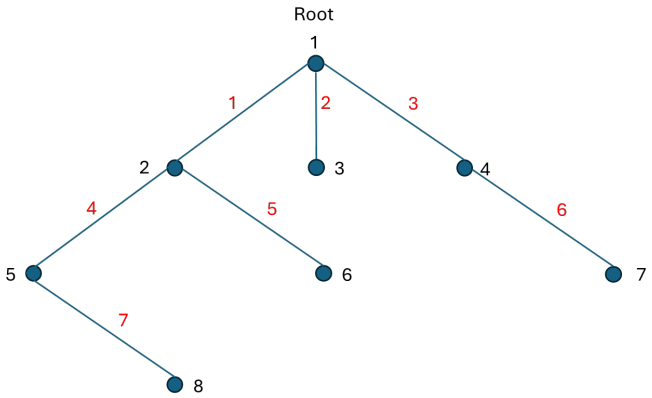
As seen in Figure 3.3, the BFS algorithm searches layer by layer, outward from the starting node.



(a) BFS algorithm step 1, discover neighbors of root.



(b) BFS algorithm step 2, discover neighbor nodes of discovered nodes.



(c) BFS algorithm step 3: continuing to explore neighbor nodes untill finnished.

Figure 3.3: BFS algorithm steps 1 to 3.

By altering the algorithm 1 to stop at node v, the returned path will be the shortest path from u to v.

3.7 Epanet

This section will give an introduction to the Epanet platform. Epanet is a software application solution used for modelling drinking water systems. It is a free, public domain software developed by the United States Environmental Protection Agency (U.S EPA) [38]. In Epanet, users can model water distribution systems as collections of links connected to nodes. Junctions, tanks and reservoirs are nodes, while links represent pipes, pumps and valves. The water distribution system can be exported as a *.INP* file consisting of all information necessary to reconstruct the network.

The .INP file structure is retrieved from the documentation of Epanet [39]. The file itself is organized in sections separated by *keywords* enclosed in square brackets. The keywords can be seen in Table 3.1.

Network Components	System Operation	Water Quality	Options	Network Map / Tags
TITLE	CURVES	QUALITY	OPTIONS	COORDINATES
JUNCTIONS	PATTERNS	REACTIONS	TIMES	VERTICES
RESERVOIRS	ENERGY	SOURCES	REPORT	LABELS
TANKS	STATUS	MIXING		BACKDROP
PIPES	CONTROLS			TAGS
PUMPS	RULES			
VALVES	DEMANDS			
EMITTERS				

Table 3.1: .INP file keywords, divided into types.

Data stored in each section varies depending on the keyword; it may consist of one or several lines of data. Examples of data from .INP file can be seen in Figure 3.4. Comments can be added individually to the data with the semicolon operator if needed.

```
[JUNCTIONS]
;ID              Elev            Demand          Pattern
 1               0               0                       ;
 2               0               0                       ;
 3               0               0                       ;
 4               0               0                       ;
 5               0               0                       ;
 6               0               0                       ;
 7               0               0                       ;
 8               0               0                       ;
```

(a) Datalines from keyword: JUNCTIONS.

```
[PIPES]
;ID              Node1           Node2           Length          Diameter        Roughness       MinorLoss       Status
 1               1               2               100             12              100             0               Open
 2               2               3               200             12              100             0               Open
 3               3               4               300             12              100             0               Open
 4               4               5               400             12              100             0               Open
```

(b) Datalines from keyword: PIPES.

Figure 3.4: Example: Datalines from .INP file.

The modelling of networks Epanet will be explained in Section 5.5.

Chapter 4

Parameter Identification at Steady State in Various Network Trees

This chapter will discuss the parameter identification of various rooted network trees, without cycles, in steady state. All the trees will be rooted in node 1. The chapter begins with a single section and builds upon that to find a generalized set of parameter equations. The chapter ends with a general method to find the parameter equations for a general network. This method is able to be implemented as an algorithm.

To minimize the number of available sensors, it is assumed that *internal* measurements, such as the head, are unknown, while measurements of the root and leaves are available (inlet and outlet). The term *internal* refers to not root and non-leaves nodes.

4.1 Base Case: a Single Section

The first case to be evaluated is a single section of pipe. With two nodes ($|V| = 2$) and one edge connecting the nodes ($|E| = 1$). As seen in Section 3.2.5, with the steady state conditions, i.e. the flow and pressure rates are zero, $\frac{\partial Q(z,t)}{\partial t} = 0$ and $\frac{\partial H(z,t)}{\partial t} = 0$, the governing PDEs are reduced to ODEs:

$$\frac{\partial H}{\partial z} + \frac{f\ Q\ |Q|}{2\ g\ D\ A^2} = 0 \tag{4.1a}$$

$$\frac{a^2}{g\ A}\frac{\partial Q}{\partial z} = 0 \tag{4.1b}$$

with distance z as the independent variable. The second term in Equation (4.1a) is replaced with $J(Q(z,t))$ giving:

$$\frac{\partial H(z,t)}{\partial z} + J(Q(z,t)) = 0 \tag{4.2a}$$

$$a_1\frac{\partial Q(z,t)}{\partial z} = 0 \tag{4.2b}$$

where $a_1 = \frac{a^2}{g\ A}$, a is the velocity of the pressure wave, A is the area of the cross-section for the specific of pipe, and g is the gravitational acceleration. As seen in Equation (3.11) the term $J(Q(z,t))$ is the hydraulic gradient, and can be calculated with $\frac{f\ Q\ |Q|}{2\ g\ D\ A^2}$. However, other functions for $J(Q(z,t))$ have been suggested by Rojas *et al.* [25], among other an equivalent quadratic function:

$$J(Q(t),\theta)) = \frac{f\ Q\ |Q|}{2\ g\ D\ A^2} = \theta Q^2(t) \tag{4.3}$$

where θ is the simplified friction parameter. Equation (4.3) is inserted into Equation (4.2a), Equation (4.2b) is equal to zero and disappears. This results in one remaining equation:

$$\theta\ (Q(z,t))^2 + \frac{\partial H(z,t)}{\partial z} = 0 \tag{4.4}$$

The simplified Equation (4.4) will now be used on a single section. The single section is illustrated in Figure 4.1, where L_1 is the total length and θ_1 is the friction parameter for the edge. In terms of graph theory, the network tree has two nodes and one edge, where node 1 is the root and node 2 is a leaf.

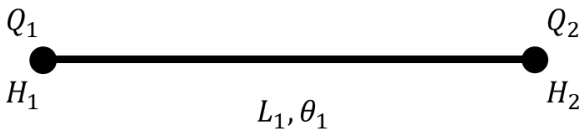


Figure 4.1: Base case: Single section, two nodes, one edge.

The partial derivative of Equation (4.4) is integrated over the full length of the section to remove the partial differential (∂z):

$$\begin{aligned}
&\int_0^{L_1}\left[\theta_1\ Q(t)^2 + \frac{\partial H(z,t)}{\partial z}\right]dz \\
&= \int_0^{L_1}\left[\theta_1\ Q(t)^2\right]dz + \int_0^{L_1}\left[\frac{\partial H(z,t)}{\partial z}\right]dz \\
&= \left[\theta_1\ Q(t)^2\ z\right]_0^{L_1} + \left[H(z,t)\right]_0^{L_1} \\
&= \theta_1\ L_1\ Q(t)^2 + (H(L_1,t) - H(0,t))
\end{aligned} \tag{4.5}$$

where $H(L_1,t) = H_2(t)$ and $H(0,t) = H_1(t)$. Without leaks in the section, the flow at the inlet will be equal to the flow at the outlet, i.e. $Q(t) = Q_1(t) = Q_2(t)$. The (t) is removed for simplicity, which results in one equation:

$$\theta_1\ L_1\ Q^2 + (H_2 - H_1) = 0 \tag{4.6}$$

Equation (4.6) is solved with respect the θ_1.

$$\theta_1 = \frac{1}{L_1}\frac{H_1 - H_2}{(Q_1)^2} \tag{4.7}$$

where length L_1 is typically known, and if the flow and head pressures are measured, the friction parameter for a single section of pipe can be calculated with Equation (4.7).

4.2 Case 1: Two Layers, Three Nodes, Two Edges

The network from the base case is expanded, adding one more layer as seen in Figure 4.2. The resulting network has three nodes ($|V| = 3$) and two edges ($|E| = 2$), where node 3 is a leaf or end node.

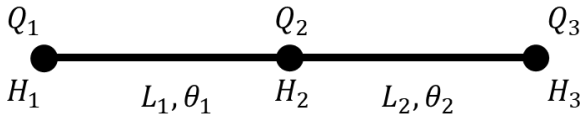


Figure 4.2: Case 1: Three nodes, two edges, one leaf.

The sections are individually solved, and that will result in one equation per section. Similar to the base case in Section 4.1, the equations are integrated over each section to find steady-state equations for the parameters θ_1 and θ_2.

$$\int_0^{L_1}\left[\theta_1\,(Q_1)^2 + \frac{\partial H(z,t)}{\partial z}\right]dz \tag{4.8a}$$

$$\int_0^{L_2}\left[\theta_2\,(Q_2)^2 + \frac{\partial H(z,t)}{\partial z}\right]dz \tag{4.8b}$$

The resulting equations after integration:

$$\theta_1\,L_1\,(Q_1)^2 + H_1(L_1,t) - H_1(0,t) = 0 \tag{4.9a}$$

$$\theta_2\,L_2\,(Q_2)^2 + H_2(L_2,t) - H_2(0,t) = 0 \tag{4.9b}$$

where $H_1(L_1,t) = H_2(t)$, $H_1(0,t) = H_1(t)$, $H_2(L_2,t) = H_3(t)$ and $H_2(0,t) = H_2(t)$. The equations are simplified by substituting the corresponding H's and removing the (t).

$$\theta_1\,L_1\,(Q_1)^2 + H_2 - H_1 = 0 \tag{4.10a}$$

$$\theta_2\,L_2\,(Q_2)^2 + H_3 - H_2 = 0 \tag{4.10b}$$

Equation (4.10a) is solved with respect to H_2:

$$H_2 = H_1 - \theta_1\,L_1\,(Q_1)^2 \tag{4.11}$$

Then, Equation (4.11) is put into Equation (4.10b) to remove H_2. Simultaneously, the head pressures are moved to the right-hand side:

$$\theta_1 L_1 (Q_1)^2 + \theta_2 L_2 (Q_2)^2 = H_1 - H_3 \tag{4.12}$$

One is left with one equation with two unknowns, i.e. the system is undetermined. One possible way to solve the system from Equation (4.12) is to have multiple sets of data. Adding one more set of data, with the index of the data in the superscript:

$$\theta_1 L_1 (Q_1^1)^2 + \theta_2 L_2 (Q_2^1)^2 = H_1^1 - H_3^1 \tag{4.13a}$$
$$\theta_1 L_1 (Q_1^2)^2 + \theta_2 L_2 (Q_2^2)^2 = H_1^2 - H_3^2 \tag{4.13b}$$

The equations can now be put into matrices on the form $A\Theta = H$, with $\Theta = [\theta_1, \theta_2]$:

$$\begin{bmatrix} L_1 (Q_1^1)^2 & L_2 (Q_2^1)^2 \\ L_1 (Q_1^2)^2 & L_2 (Q_2^2)^2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} H_1^1 - H_3^1 \\ H_1^2 - H_3^2 \end{bmatrix} \tag{4.14}$$

the system is solvable, if A is invertible, i.e rank(A) $= 2$. The solution would then be $\Theta = A^{-1}H$. It can be noted the matrix A is invertible if each of the flow measurements is *sufficiently* different from each other. However, in a steady state without leaks, the flow into the section will be equal to the flow out of the section. Updating the A matrix with $Q_1 = Q_2$ and analysing it becomes evident that the rank will never be larger than 1, as there will always be a relationship between a first and second column in the constants $L_{\#}$.

$$\begin{bmatrix} L_1 (Q_1^1)^2 & L_2 (Q_1^1)^2 \\ L_1 (Q_1^2)^2 & L_2 (Q_1^2)^2 \\ \vdots & \vdots \\ L_1 (Q_1^k)^2 & L_2 (Q_1^k)^2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} H_1^1 - H_3^1 \\ H_1^2 - H_3^2 \\ \vdots \\ H_1^k - H_3^k \end{bmatrix} \tag{4.15}$$

That means that the two sections should be treated as one and that the base case can be used to find a common θ for the two sections. The case where the sections shall be treated as one single section can be recognised with the degree of a node being two ($d(V) = 2$).

4.3 Case 2: Two Layers, Four Nodes, Three Edges, Two End Nodes

The next case to be investigated is a network tree with four nodes ($|V| = 4$), three edges ($|E| = 3$) and two leaves. There are two layers in the network. The network can be seen in Figure 4.3. There is a slight modification to the network figure compared to previous network figures, Q is moved from the node to the edge, in contrast with the previous cases (Sections 4.2 and 4.3) where Q was illustrated on the nodes. The movement is based upon the assumptions $\frac{\partial Q}{\partial x} = 0$ (continuity of mass), i.e. the flow at the inlet is equal to the flow at the outlet.

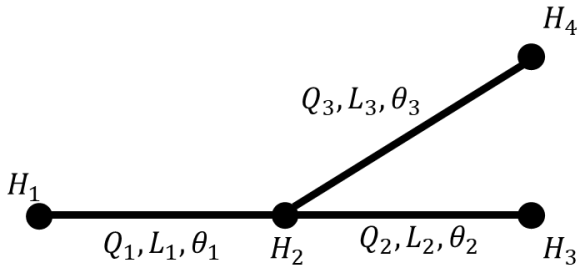


Figure 4.3: Case 2: Four nodes, three edges, two end-nodes, two layers.

The procedure remains similar to the previous cases, the sections are individually integrated:

$$\int_0^{L_1} \left[\theta_1 \, (Q_1(t))^2 + \frac{\partial H(z,t)}{\partial z} \right] dz \tag{4.16a}$$

$$\int_0^{L_2} \left[\theta_2 \, (Q_2(t))^2 + \frac{\partial H(z,t)}{\partial z} \right] dz \tag{4.16b}$$

$$\int_0^{L_3} \left[\theta_3 \, (Q_3(t))^2 + \frac{\partial H(z,t)}{\partial z} \right] dz \tag{4.16c}$$

Solving each equation gives:

$$\theta_1 \, L_1 \, (Q_1(t))^2 + H_1(L_1,t) - H_1(0,t) = 0 \tag{4.17a}$$

$$\theta_2 \, L_2 \, (Q_2(t))^2 + H_2(L_2,t) - H_2(0,t) = 0 \tag{4.17b}$$

$$\theta_3 \, L_3 \, (Q_3(t))^2 + H_3(L_b,t) - H_3(0,t) = 0 \tag{4.17c}$$

Where $H_1(L_1,t) = H_2$, $H_1(0,t) = H_1$, $H_2(L_2,t) = H_3$, $H_2(0,t) = H_2$, $H_3(L_b,t) = H_4$ and $H_3(0,t) = H_2$. For simplicity the heads ($H(L_\#,t)$) in Equation (4.17) are replaced with their respective node heads ($H_\#$), and (t) is omitted:

$$\theta_1 \, L_1 \, (Q_1)^2 + H_2 - H_1 = 0 \tag{4.18a}$$

$$\theta_2 \, L_2 \, (Q_2)^2 + H_3 - H_2 = 0 \tag{4.18b}$$

$$\theta_3 \, L_3 \, (Q_3)^2 + H_4 - H_2 = 0 \tag{4.18c}$$

Equation (4.18a) is solved with respect to H_2:

$$H_2 = H_1 - \theta_1 L_1 \, (Q_1)^2 \tag{4.19}$$

It is noted that the Equation (4.19) for H_2 is equal to H_2 found in Section 4.3. Equation (4.19) is inserted into Equations (4.18b) and (4.18c):

$$\theta_1 L_1 (Q_1)^2 + \theta_2 L_2 Q_2^2 = H_1 - H_3 \tag{4.20a}$$
$$\theta_1 L_1 (Q_1)^2 + \theta_3 L_3 Q_3^2 = H_1 - H_4 \tag{4.20b}$$

The equations can now be put into matrices on the form $A\Theta = H$, with $\Theta = [\theta_1, \theta_2, \theta_3]$:

$$\begin{bmatrix} L_1(Q_1)^2 & L_2(Q_2)^2 & 0 \\ L_1(Q_1))^2 & 0 & L_3(Q_3)^2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} H_1 - H_3 \\ H_1 - H_4 \end{bmatrix} \tag{4.21}$$

As A is (2×3) matrix, it is not invertible, at most rank(A) = 2. Therefore, a minimum of two datasets is required. The superscript is introduced again for indexing the different sets of data. A system with $k \in (0, \infty)$ dataset of measurements would look like:

$$\begin{bmatrix} L_1(Q_1^1)^2 & L_2(Q_2^1)^2 & 0 \\ L_1(Q_1^1))^2 & 0 & L_3(Q_3^1)^2 \\ L_1(Q_1^2)^2 & L_2(Q_2^2)^2 & 0 \\ L_1(Q_1^2))^2 & 0 & L_3(Q_3^2)^2 \\ \vdots & \vdots & \vdots \\ L_1(Q_1^k)^2 & L_2(Q_2^k)^2 & 0 \\ L_1(Q_1^k))^2 & 0 & L_3(Q_3^k)^2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} H_1^1 - H_3^1 \\ H_1^1 - H_4^1 \\ H_1^2 - H_3^2 \\ H_1^2 - H_4^2 \\ \vdots \\ H_1^k - H_3^k \\ H_1^k - H_4^k \end{bmatrix} \tag{4.22}$$

With $k > 1$, the matrix A is rectangular and the solution to Θ may be inconsistent. Therefore the solution to Θ would be the least square solution:

$$\bar{\Theta} = (A^T A)^{-1} A^T H \tag{4.23}$$

Each dataset k must exhibit significant differences to have the full rank of A (rank(A) = 3).

4.4 Case 3: Two layers, Five Nodes, Four Edges, Three end-nodes

From Section 4.2 and Section 4.3, it is evident that multiple sets of data are required to determine the friction parameter Θ. Therefore, the superscription of the dataset index is included from the beginning. The step of initial setup and integration over each section, is omitted from this section. The network tree in case 3 has $|V| = 5$ number of nodes, $|E| = 4$ number of edges and $|V_{leaves}| = 3$ number of leaves, and can be seen in Figure 4.4. Case 3 is an extension of case 2 with one additional edge connected to node 2.

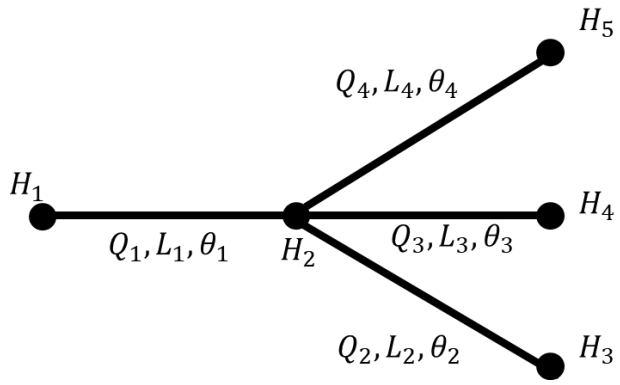


Figure 4.4: Case 3: Five nodes, 4 edges, three end-nodes.

The steady-state system equations are derived from one set of data, $k = 1$:

$$\theta_1 \, L_1 \, (Q_1^1)^2 + H_2^1 - H_1^1 = 0 \tag{4.24a}$$
$$\theta_2 \, L_2 \, (Q_2^1)^2 + H_3^1 - H_2^1 = 0 \tag{4.24b}$$
$$\theta_3 \, L_3 \, (Q_3^1)^2 + H_4^1 - H_2^1 = 0 \tag{4.24c}$$
$$\theta_4 \, L_4 \, (Q_4^1)^2 + H_5^1 - H_2^1 = 0 \tag{4.24d}$$

Equation (4.24a) is expressed for head in node 2, H_2^1, and inserted in Equation (4.24b), Equation (4.24c) and Equation (4.24d). Head pressures $H_\#$ is moved to the right-hand side:

$$\theta_1 \, L_1 \, (Q_1^1)^2 + \theta_2 \, L_2 \, (Q_2^1)^2 = H_1^1 - H_3^1 \tag{4.25a}$$
$$\theta_1 \, L_1 \, (Q_1^1)^2 + \theta_3 \, L_3 \, (Q_3^1)^2 = H_1^1 - H_4^1 \tag{4.25b}$$
$$\theta_1 \, L_1 \, (Q_1^1)^2 + \theta_4 \, L_4 \, (Q_4^1)^2 = H_1^1 - H_5^1 \tag{4.25c}$$

with $k \in (0, \infty)$ number of sets of data, the system matrices for $A\Theta = H$ will be:

$$\underbrace{\begin{bmatrix} L_1\,(Q_1^1)^2 & L_2\,(Q_2^1)^2 & 0 & 0 \\ L_1\,(Q_1^1)^2 & 0 & L_3\,(Q_3^1)^2 & 0 \\ L_1\,(Q_1^1)^2 & 0 & 0 & L_4\,(Q_4^1)^2 \\ \vdots & \vdots & \vdots & \vdots \\ L_1\,(Q_1^k)^2 & L_2\,(Q_2^k)^2 & 0 & 0 \\ L_1\,(Q_1^k)^2 & 0 & L_3\,(Q_3^k)^2 & 0 \\ L_1\,(Q_1^k)^2 & 0 & 0 & L_4\,(Q_4^k)^2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}}_{\Theta} = \underbrace{\begin{bmatrix} H_1^1 - H_3^1 \\ H_1^1 - H_4^1 \\ H_1^1 - H_5^1 \\ \vdots \\ H_1^k - H_3^k \\ H_1^k - H_4^k \\ H_1^k - H_5^k \end{bmatrix}}_{H} \tag{4.26}$$

As A^1 is 3×4 matrix, the matrix is not invertible. With one set of data, the rank is at most: $\text{rank}(A) = 3$. Therefore, a minimum of two datasets is required ($k = 2$). With $k > 1$, the matrix A is rectangular and the solution to Θ may be inconsistent. Therefore the solution to Θ would be the least square solution: Each dataset k must exhibit significant differences to have the full rank of A ($\text{rank}(A) = 4$).

4.5 Case 4: Two Layers, *j* Nodes, *i* Edges

The next case is a generalization of the network with two layers. The purpose is to find a general system matrix for two layered networks. Layer two consists of nodes and edges:

$$V_{\text{layer 2}} = \{3, 4, \ldots, j\} \tag{4.27a}$$
$$E_{\text{layer 2}} = \{2, 3, \ldots, i\} \tag{4.27b}$$

where:

$$j \in [5, (5+m)], \quad m \in (0, \infty)$$
$$i \in [4, (4+m)]$$

The network tree will be explored to find system equations for a network with a growing second layer. The total number of nodes is $|V| = 4 + m$ and the total number of edges is $|E| = 3 + m$, while the number of leaves will is $|V_{leaves}| = 2 + m$.

The steady-state system equations are derived for 1 set of data, $k = 1$ and can be seen in Equation (4.28):

$$\theta_1\ L_1\,(Q_1^1)^2 + H_2^1 - H_1^1 = 0 \tag{4.28a}$$
$$\theta_2\ L_2\,(Q_2^1)^2 + H_3^1 - H_2^1 = 0 \tag{4.28b}$$
$$\theta_3\ L_3\,(Q_3^1)^2 + H_4^1 - H_2^1 = 0 \tag{4.28c}$$
$$\vdots$$
$$\theta_i\ L_i\,(Q_i^1)^2 + H_j^1 - H_2^1 = 0 \tag{4.28d}$$

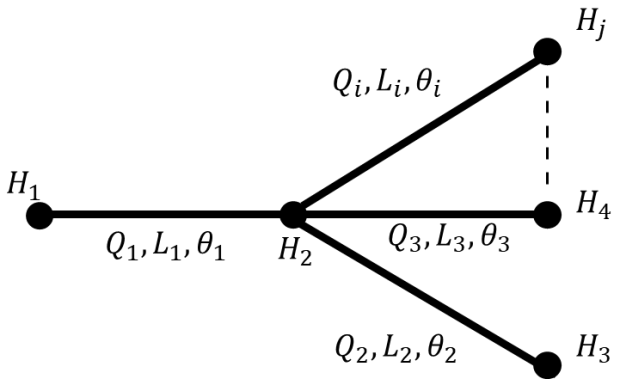


Figure 4.5: Case 4: Two layers, j nodes, i edges.

The head pressures are moved to the right-hand side:

$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_2\ L_2\ (Q_2^1)^2 = H_1^1 - H_3^1 \tag{4.29a}$$
$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_3\ L_3\ (Q_3^1)^2 = H_1^1 - H_4^1 \tag{4.29b}$$
$$\vdots$$
$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_i\ L_i\ (Q_i^1)^2 = H_1^1 - H_j^1 \tag{4.29c}$$

The system matrices for $A\Theta = H$ will be:

$$\underbrace{\begin{bmatrix} L_1\ (Q_1^1)^2 & L_2\ (Q_2^1)^2 & 0 & 0\ \dots & 0 \\ L_1\ (Q_1^1)^2 & 0 & L_3\ (Q_3^1)^2 & 0\ \dots & 0 \\ \vdots & 0 & 0 & \ddots & 0 \\ L_1\ (Q_1^1)^2 & 0 & 0 & 0\ \dots & L_i\ (Q_i^1)^2 \end{bmatrix}}_{A^1} \underbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \\ \theta_i \end{bmatrix}}_{\Theta} = \underbrace{\begin{bmatrix} H_1^1 - H_2^1 \\ H_1^1 - H_3^1 \\ \vdots \\ H_1^1 - H_j^1 \end{bmatrix}}_{H^1} \tag{4.30}$$

where matrix A^1 have data from the first set, $(k = 1)$. The number of rows in A^1 are equal to the number of leaves, and the number of column is equal to the number of edges, making the size of $A^1 = (|V_{leaves}| \times |E|)$. Θ is of size $(|E| \times 1)$ and H^1 is of size $(|V_{leaves}| \times 1)$. When using $k \in (0, \infty)$ datasets, the individual matrices of A^k and H^k will be stacked upon each other. A and H will have the form:

$$A = \begin{bmatrix} A^1 \\ A^2 \\ \vdots \\ A^k \end{bmatrix}, \quad \text{and:} \quad H = \begin{bmatrix} H^1 \\ H^2 \\ \vdots \\ H^k \end{bmatrix} \tag{4.31}$$

With $k > 1$, the matrix A is rectangular and the solution to Θ may be inconsistent. Therefore, the solution to Θ would be the least square solution: Each dataset k must exhibit significant differences for the rank$(A) = |E|$.

4.6 Case 5: Three Layers, Two Edges per Node

For case 5 there is added one more layer to the network tree. Each node of the tree has two edges, with the exception of the root and leaves, which only have one node. For this network there are $|V| = 6$ nodes, $|E| = 5$ edges and $|V_{leaves}| = 3$ leaves. An illustration of the network can be seen in Figure 4.6:

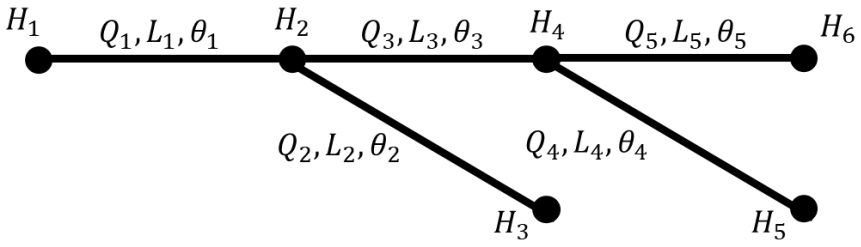


Figure 4.6: Case 5: Three layers, two edges per node.

The network in case 5 can be seen as a combination of two trees from case 2 Section 4.3. The method for deriving the steady-state equations remains the same, and the system of equations will be:

$$\theta_1 \, L_1 \, (Q_1^1)^2 + H_2^1 - H_1^1 = 0 \tag{4.32a}$$
$$\theta_2 \, L_2 \, (Q_2^1)^2 + H_3^1 - H_2^1 = 0 \tag{4.32b}$$
$$\theta_3 \, L_3 \, (Q_3^1)^2 + H_4^1 - H_2^1 = 0 \tag{4.32c}$$
$$\theta_4 \, L_4 \, (Q_4^1)^2 + H_5^1 - H_4^1 = 0 \tag{4.32d}$$
$$\theta_5 \, L_5 \, (Q_5^1)^2 + H_6^1 - H_4^1 = 0 \tag{4.32e}$$

The internal head pressure measurements from node 2, H_2, and node 4, H_4, will be removed as it is assumed unknown. Therefore Equation (4.32a) is solved with respect to H_2:

$$H_2^1 = H_1^1 - \theta_1 \, L_1 \, (Q_1^1)^2 \tag{4.33}$$

Equation (4.32c) is solved with respect to H_4:

$$\begin{aligned} H_4^1 &= H_2^1 - \theta_3 \, L_3 \, (Q_3^1)^2 \\ H_4^1 &= H_1^1 - \theta_1 \, L_1 \, (Q_1^1)^2 - \theta_3 \, L_3 \, (Q_3^1)^2 \end{aligned} \tag{4.34}$$

Inserting Equation (4.33) and Equation (4.34) into Equation (4.32):

$$\theta_1 \, L_1 \, (Q_1^1)^2 + \theta_2 \, L_2 \, (Q_2^1)^2 = H_1^1 - H_3^1 \tag{4.35a}$$
$$\theta_1 \, L_1 \, (Q_1^1)^2 + \theta_3 \, L_3 \, (Q_3^1)^2 + \theta_4 \, L_4 \, (Q_4^1)^2 = H_1^1 - H_5^1 \tag{4.35b}$$
$$\theta_1 \, L_1 \, (Q_1^1)^2 + \theta_3 \, L_3 \, (Q_3^1)^2 + \theta_5 \, L_5 \, (Q_5^1)^2 = H_1^1 - H_6^1 \tag{4.35c}$$

The system matrices $A\Theta = H$ will be:

$$\underbrace{\begin{bmatrix} L_1\,(Q_1^1)^2 & 0 & L_2\,(Q_2^1)^2 & 0 & 0 \\ L_1\,(Q_1^1)^2 & L_3\,(Q_3^1)^2 & 0 & L_4\,(Q_4^1)^2 & 0 \\ L_1\,(Q_1^1)^2 & L_3\,(Q_3^1)^2 & 0 & 0 & L_5\,(Q_5^1)^2 \end{bmatrix}}_{A^1} \underbrace{\begin{bmatrix} \theta_1 \\ \theta_3 \\ \theta_2 \\ \theta_4 \\ \theta_5 \end{bmatrix}}_{\Theta} = \underbrace{\begin{bmatrix} H_1^1 - H_3^1 \\ H_1^1 - H_4^1 \\ H_1^1 - H_5^1 \end{bmatrix}}_{H^1} \tag{4.36}$$

With $k = 1$, the matrix A is rectangular and the solution to Θ may be inconsistent. Therefore the solution to Θ would be the least square solution: Each dataset k must exhibit significant differences for the $\text{rank}(A) = |E|$.

4.7 Case 6: Three Layers, Nodes in Layer Two has *j1* and *j2* Nodes

The next case to be investigated is case 6. The first two layers are similar to Section 4.3. However, there is added one more layer as seen in Figure 4.7. Layer three consists of nodes and edges:

$$V_{\text{layer 3}} = \{5, \ldots, j1, 6, \ldots, j2\} \tag{4.37a}$$

$$E_{\text{layer 3}} = \{4, \ldots, i1, 5, \ldots, i2\} \tag{4.37b}$$

where:

$$j1 \in [7, (7+m)], \quad m \in (0, \infty)$$
$$j2 \in [7+m), ((7+m)+n], \quad n \in (0, \infty)$$
$$i1 \in [6, (6+m)]$$
$$i2 \in [6+m), ((6+m)+n]$$

The network will be explored to find system equations for a network with a growing third layer. The total number of nodes is $|V| = (6+m+n)$, the total number of edges is $|E| = (5+m+n)$, and the total number of leaves is $|V_{leaves}| = (2+m+n)$. It is noted that if $m = 1$ the degree of node 3 will be two ($d(3) = 2$). This will lead to a similar case as seen in Section 4.2, where the columns in the A matrix will be linearly dependent, and the rank will be less than the number of Edges, therefor if $m = 1$, the edges 3 and 5 must be treated as one single edge. The same note is applicable for $n = 1$.

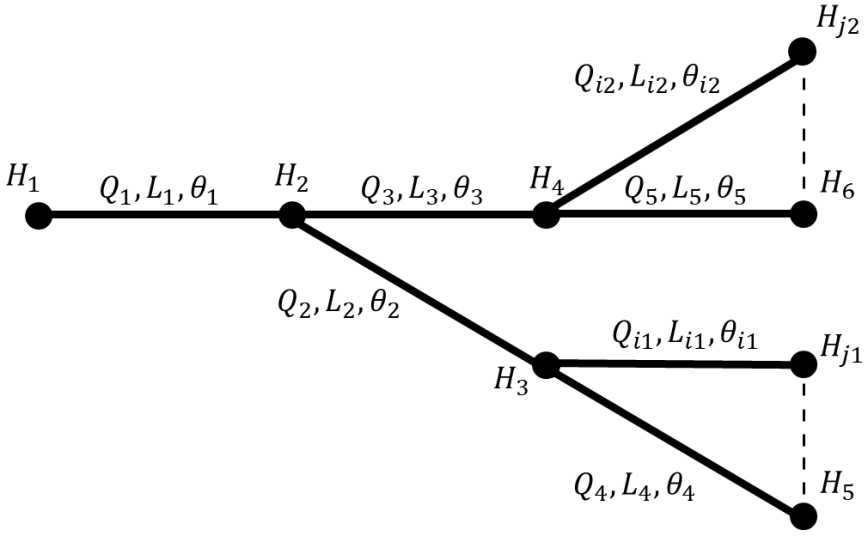


Figure 4.7: Case 6: Three layers, where *j1*, *j2*, *i1* and *i2* are seen in Equation (4.37).

The method for deriving the steady-state equations remains the same, and the system of equations will be:

$$\theta_1\ L_1\ (Q_1^1)^2 + H_2^1 - H_1^1 = 0 \tag{4.38a}$$
$$\theta_2\ L_2\ (Q_2^1)^2 + H_3^1 - H_2^1 = 0 \tag{4.38b}$$
$$\theta_3\ L_3\ (Q_3^1)^2 + H_5^1 - H_4^1 = 0 \tag{4.38c}$$
$$\theta_4\ L_4\ (Q_4^1)^2 + H_5^1 - H_3^1 = 0 \tag{4.38d}$$
$$\vdots$$
$$\theta_{i1}\ L_{i1}\ (Q_{i1}^1)^2 + H_{j1}^1 - H_4^1 = 0 \tag{4.38e}$$
$$\theta_5\ L_5\ (Q_5^1)^2 + H_6^1 - H_4^1 = 0 \tag{4.38f}$$
$$\vdots$$
$$\theta_{i2}\ L_{i2}\ (Q_{i2}^1)^2 + H_{j2}^1 - H_4^1 = 0 \tag{4.38g}$$

The internal pressures heads H_2, H_3 and H_4 are removed from the set of equations. The remaining equations are:

$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_2\ L_2\ (Q_2^1)^2 + \theta_4\ L_4\ (Q_4^1)^2 = H_1^1 - H_5^1 \tag{4.39a}$$
$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_2\ L_2\ (Q_2^1)^2 + \theta_{i1}\ L_{i1}\ (Q_{i1}^1)^2 = H_1^1 - H_{j2}^1 \tag{4.39b}$$
$$\vdots \tag{4.39c}$$
$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_3\ L_3\ (Q_3^1)^2 + \theta_5\ L_5\ (Q_5^1)^2 = H_1^1 - H_6^1 \tag{4.39d}$$
$$\vdots \tag{4.39e}$$
$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_3\ L_3\ (Q_3^1)^2 + \theta_{i2}\ L_{i2}\ (Q_{i2}^1)^2 = H_1^1 - H_{j2}^1 \tag{4.39f}$$

The system equations are now organized into system matrices $A\Theta = H$, due to the large size of the system, the A^1 matrix is split into A_1^1 and A_2^1:

$$A_1 = \begin{bmatrix} L_1\ (Q_1^1)^2 & L_2\ (Q_2^1)^2 & 0 \\ \vdots & \vdots & \vdots \\ L_1\ (Q_1^1)^2 & L_2\ (Q_2^1)^2 & 0 \\ L_1\ (Q_1^1)^2 & 0 & L_3\ (Q_3^1)^2 \\ \vdots & \vdots & \vdots \\ L_1\ (Q_1^1)^2 & 0 & L_3\ (Q_3^1)^2 \end{bmatrix} \tag{4.40}$$

$$A_2 = \begin{bmatrix} L_4\ (Q_4^1)^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & L_{i1}\ (Q_{i1}^1)^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & L_5\ (Q_5^1)^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & L_{i2}\ (Q_{i2}^1)^2 \end{bmatrix} \tag{4.41}$$

where A_1^1 is of size ($|V_{leaves}| \times |V_{internal}|$). By $|V_{internal}|$ is meant number of nodes that are non-root and non-leaves. A_2^1 is square and of size ($|V_{leaves}| \times |V_{leaves}|$). Combined the system matrices are:

$$\underbrace{\begin{bmatrix} A_1^1 & A_2^1 \end{bmatrix}}_{A^1} \underbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \vdots \\ \theta_{i1} \\ \theta_5 \\ \vdots \\ \theta_{i2} \end{bmatrix}}_{\Theta} = \underbrace{\begin{bmatrix} H_1^1 - H_5^1 \\ \vdots \\ H_1^1 - H_{j1}^1 \\ H_1^1 - H_6^1 \\ \vdots \\ H_1^1 - H_{j2}^1 \end{bmatrix}}_{H^1} \tag{4.42}$$

The combined matrix A^1 is rectangular and of size ($|V_{leaves}| \times |E|$). The least-square solution $\bar{\Theta}$ exist if rank(A) = $|E|$, which requires $k \geq 2$.

4.8 Case 7: Four Layers, Two Edges per Node

The last network tree to be explored is based on the network tree in Section 4.6 and extended with one more layer, making it four layers in total. There are two edges per internal node. The total number of nodes is $|V| = 8$, the total number of edges $|E| = 7$, the number of internal nodes $|V_{internal}| = 4$, and the total number of end nodes is $|V_{leaves}| = 4$. The network can be seen in Figure 4.8:

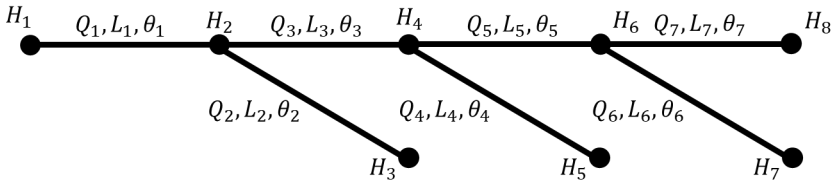


Figure 4.8: Four layers, two edges per node.

The method for deriving the steady-state equations remains the same, and the system of equa-

tions will be:

$$\theta_1\ L_1\ (Q_1^1)^2 + H_2^1 - H_1^1 = 0 \tag{4.43a}$$
$$\theta_2\ L_2\ (Q_2^1)^2 + H_3^1 - H_2^1 = 0 \tag{4.43b}$$
$$\theta_3\ L_3\ (Q_3^1)^2 + H_4^1 - H_2^1 = 0 \tag{4.43c}$$
$$\theta_4\ L_4\ (Q_4^1)^2 + H_5^1 - H_4^1 = 0 \tag{4.43d}$$
$$\theta_5\ L_5\ (Q_5^1)^2 + H_6^1 - H_4^1 = 0 \tag{4.43e}$$
$$\theta_6\ L_6\ (Q_6^1)^2 + H_7^1 - H_6^1 = 0 \tag{4.43f}$$
$$\theta_7\ L_7\ (Q_7^1)^2 + H_8^1 - H_6^1 = 0 \tag{4.43g}$$

The internal pressure H_2, H_3 and H_4 is removed from the set Equation (4.43), and we are left with equations:

$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_2\ L_2\ (Q_2^1)^2 = H_1^1 - H_3^1 \tag{4.44a}$$
$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_3\ L_3\ (Q_3^1)^2 + \theta_4\ L_4\ (Q_4^1)^2 = H_1^1 - H_5^1 \tag{4.44b}$$
$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_3\ L_3\ (Q_3^1)^2 + \theta_5\ L_5\ (Q_5^1)^2 + \theta_6\ L_6\ (Q_6^1)^2 = H_1^1 - H_7^1 \tag{4.44c}$$
$$\theta_1\ L_1\ (Q_1^1)^2 + \theta_3\ L_3\ (Q_3^1)^2 + \theta_5\ L_5\ (Q_5^1)^2 + \theta_7\ L_7\ (Q_7^1)^2 = H_1^1 - H_8^1 \tag{4.44d}$$

Due to the large size of the system, the A matrix i split into A_1 and A_2

$$A_1^1 = \begin{bmatrix} L_1\ (Q_1^1)^2 & 0 & 0 \\ L_1\ (Q_1^1)^2 & L_3\ (Q_3^1)^2 & 0 \\ L_1\ (Q_1^1)^2 & L_3\ (Q_3^1)^2 & L_5\ (Q_5^1)^2 \\ L_1\ (Q_1^1)^2 & L_3\ (Q_3^1)^2 & L_5\ (Q_5^1)^2 \end{bmatrix} \tag{4.45}$$

$$A_2^1 = \begin{bmatrix} L_2\ (Q_2^1)^2 & 0 & 0 & 0 \\ 0 & L_4\ (Q_4^1)^2 & 0 & 0 \\ 0 & 0 & L_6\ (Q_6^1)^2 & 0 \\ 0 & 0 & 0 & L_7\ (Q_7^1)^2 \end{bmatrix} \tag{4.46}$$

where A_1^1 is of size $(|V_{leaves}| \times |V_{internal}|)$ and A_2^1 is square and of size $(|V_{leaves}| \times |V_{leaves}|)$. Combined the system is:

$$\underbrace{\begin{bmatrix} A_1^1 & A_2^1 \end{bmatrix}}_{A^1} \underbrace{\begin{bmatrix} \theta_1 \\ \theta_3 \\ \theta_5 \\ \theta_2 \\ \theta_4 \\ \theta_6 \\ \theta_7 \end{bmatrix}}_{\Theta} = \underbrace{\begin{bmatrix} H_1^1 - H_3^1 \\ H_1^1 - H_5^1 \\ H_1^1 - H_7^1 \\ H_1^1 - H_8^1 \end{bmatrix}}_{H^1} \tag{4.47}$$

The combined matrix A^1 is rectangular and of size $(|V_{leaves}| \times |E|)$. The least-square solution $\bar{\Theta}$ exist if $\text{rank}(A) = |E|$, which requires $k \geq 2$.

4.9 Generalization of Network Trees

The previous sections in Chapter 4 have shown how to derive network equations of many variants of network trees. This has led to a proposal for a generalization of the system matrices ($A\Theta = H$) for a network of arbitrary size, which will be introduced in this section.

There will be one equation per leaf, where each equation holds the terms of the edges and nodes from the root to the leaves. It is explained with an example from Section 4.8, looking at Equation (4.44d):

$$\theta_1 \; L_1 \; (Q_1^1)^2 + \theta_3 \; L_3 \; (Q_3^1)^2 + \theta_5 \; L_5 \; (Q_5^1)^2 + \theta_7 \; L_7 \; (Q_7^1)^2 = H_1^1 - H_8^1 \tag{4.48}$$

Here, it can be seen that; the left-hand side of the equal sign contains terms related to the edgepath from node 1 (root) to node 8 (leaf). Each term is the friction parameter from the edge multiplied with the length of the edge, and multiplied with the squared of the flow through the edge. On the right-hand side of the equation is the difference in head pressure from node 1 (root) to node 8 (leaf).

The matrix A is to be built up by two matrices A_1 and A_2, such that $A = [A_1, A_2]$. Where the first column in A_1 consists of the terms of the friction parameter (θ) related to the edge of the root. The other columns of a will consist of the terms related to the friction parameter (θ) for the *internal* nodes of the network, i.e., not a leaf or end node. As seen when deriving equations, the head measurements for the inner nodes can be removed. While the flow measurements are kept for the edges, they can also be described with the sum of the root flow and the flow from leaves. The size of A_1 is $(|V_{leaves}| \times |V_{internal}|)$.

A_2 is to be square and only with elements on the diagonal. The elements of the diagonal are the terms of the friction parameters (θ) of the edges connected to the leaves. The size of A_2 is $(|V_{leaves}| \times |V_{leaves}|)$. Combining the two matrices into A will give a matrix of size $(|V_{leaves}| \times |E|)$. Each row of the system matrices will consist of elements from the shortest path from the root to the leaf, or end node.

The Θ will consist of all friction parameters θ and is of size $(|E| \times 1)$. The matrix H will be the head pressure of the root, minus the corresponding head pressure of the end node. H has the size $(|V_{leaves}| \times 1)$.

For $k \in (0, \infty)$ sets of data, the matrices would be stacked upon each other:

$$A = \begin{bmatrix} A_1^1 & A_2^1 \\ A_1^2 & A_2^2 \\ \vdots & \vdots \\ A_1^k & A_2^k \end{bmatrix} \quad \text{, and:} \quad H = \begin{bmatrix} H^1 \\ H^2 \\ \vdots \\ H^k \end{bmatrix} \tag{4.49}$$

The least-square solution $\bar{\Theta}$ exist if $\text{rank}(A) = E$, which may require $k \geq 2$.

Chapter 5

Automated System Matrix Generation of Water Network Trees

The setup of system matrices $A\Theta = H$ is automated with an algorithm using the generalization of network trees uncovered in Section 4.9, and with the use of graph theory from Section 3.6. The system matrices can be further processed in Matlab by first modelling the water network in Epanet, exporting this model, and running the algorithm. The junctions will be represented as nodes V, and pipes will be represented as edges E. The algorithm will use a combination of available Matlab algorithms for graph theory and customise it to iterate through each node, add symbolic variables to measurements such as head-pressure H and flow Q, and add friction parameters θ to each edge. The algorithm will return system matrices A, Θ and H. The general flow of the algorithm can be seen in Figure 5.1.

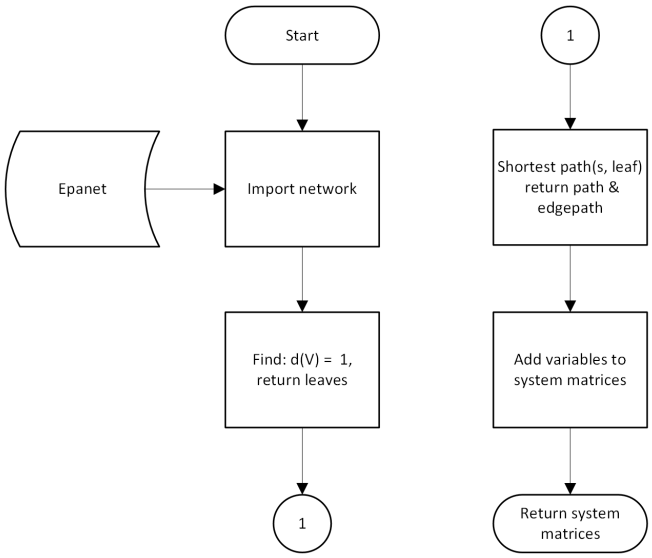


Figure 5.1: Flow chart: Algorithm to import and analyse network trees.

5.1 Step 1: Import Network Tree

The first step is to import the network tree as .INP file to Matlab. The "import network" process reads the .INP file of the network system and creates a Matlab structured array that groups data into data containers, such as junctions and pipes. The pipe data contains each pipe's start node, end node, and length and will form the basis of the network *graph*-object. Not all of the keywords available in the .INP file structure (Table 3.1) are imported; at minimum, the keywords of junctions and pipes must be imported.

5.2 Step 2: Find Leaves

The next step is to find the *leaves* or *end nodes*. The leaves are found by iterating through the graph's nodes and finding nodes with degree one ($d(V) = 1$). All of the nodes are checked except the root, or start node. If the degree of a node is 1, the node is stored in a list. A counter is initialized at zero and incremented for every leaf that is found. Pseudocode of step 2 can be seen in Algorithm 2:

Algorithm 2: Find leaves

```
Data: Graph G
Result: Number of end-nodes, and list of end-node ID
start_node ← 1;
end_nodes ← 0;
End_nodes_ID ← [];
for i ← 1 to height(G.Nodes) do
    if i == start_node then
        continue;
    end
    if degree(G, i) == 1 then
        end_nodes ← end_nodes + 1;
        End_nodes_ID.append(G.Nodes.Name[i]);
    end
end
```

5.3 Step 3: Shortest Path

As the system matrices are strongly dependent on both the path and edgepath from the root to the leaves, the next step is to use the shortest path [36] algorithm to find and store paths and edgepaths from start node, to all leaves. The list of leaves is passed from step 2 to step 3. The pseudocode of step 3 can be seen in Algorithm 3.

Algorithm 3: Finding Paths, Distances, and Edgepaths.

```
Data: Graph G, start_node, End_nodes_ID
Result: Paths_struct, Edgepaths_struct
Paths ← struct();
Edgepaths ← struct();
Distances ← [];
for i ← 1 to length(end_nodes_ID) do
    end_node ← end_nodes_ID[i];
    [Path, D, Edgepath] ← shortestpath(G, start_node, end_node);
    Paths_struct(i).Path ← Path;
    Edgepaths_struct(i).Edgepath ← Edgepath;
    Distances(i) ← D;
end
```

The distance is stored, however it is not further processed as the information of the length of each edge is already stored in the *graph*-object.

5.4 Step 4: Generate System Matrices

When all paths and edgepaths for all leaves are found, the last step is to create the system matrices. As discussed in Section 4.9, the A matrix is built from two matrices, A_1 and A_2. The two matrices is constructed step by step. The final system equations are extracted from the information gathered in the previous steps and with the knowledge from Section 4.9. As an example, the Equation (4.44d) is highlighted in red in the matrices below, and in Figure 5.2 the system equation would be:

$$\theta_1 \, L_1 \, Q_1^2 + \theta_2 \, L_2 \, Q_2^2 + \theta_3 \, L_3 \, Q_3^2 + \theta_4 \, L_4 \, Q_4^2 = H_1 - H_8 \tag{5.1}$$

$$A_1 = \begin{bmatrix} L_1 \, Q_1^2 & 0 & 0 \\ L_1 \, Q_1^2 & L_2 \, Q_2^2 & 0 \\ L_1 \, Q_1^2 & L_2 \, Q_2^2 & L_3 \, Q_3^2 \\ L_1 \, Q_1^2 & L_2 \, Q_2^2 & L_3 \, Q_3^2 \end{bmatrix}, \quad A_2 = \begin{bmatrix} L_4 \, Q_4^2 & 0 & 0 & 0 \\ 0 & L_5 \, Q_5^2 & 0 & 0 \\ 0 & 0 & L_6 \, Q_6^2 & 0 \\ 0 & 0 & 0 & L_7 \, Q_7^2 \end{bmatrix}$$

$$\Theta^T = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6 & \theta_7 \end{bmatrix} \quad \text{and} \quad H = \begin{bmatrix} H_1 - H_5 \\ H_1 - H_6 \\ H_1 - H_7 \\ H_1 - H_8 \end{bmatrix}$$

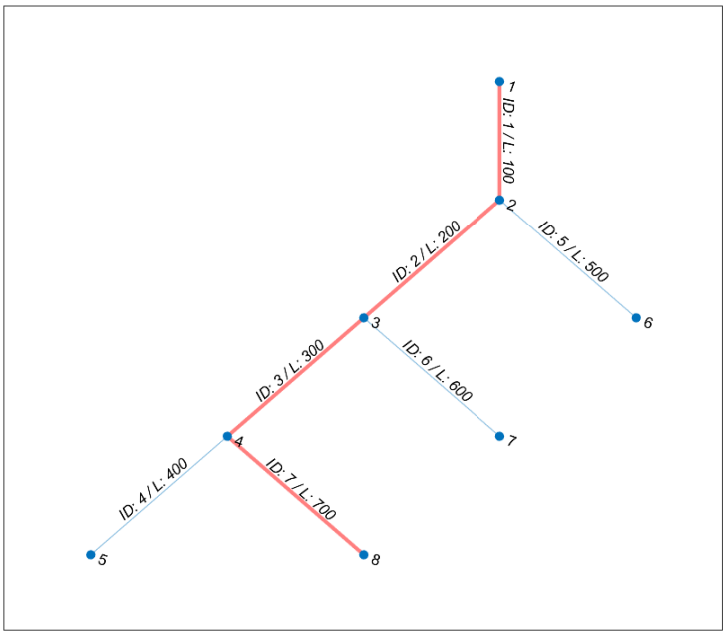


Figure 5.2: Shortest path node 1 - node 8.

5.5 Epanet Network Tree Analysis Setup

Epanet is used to generate to .INP files and to verify the results of the solution to friction parameter steady state analysis. Therefore, this section will give an introduction to the modelling and simulation in Epanet.

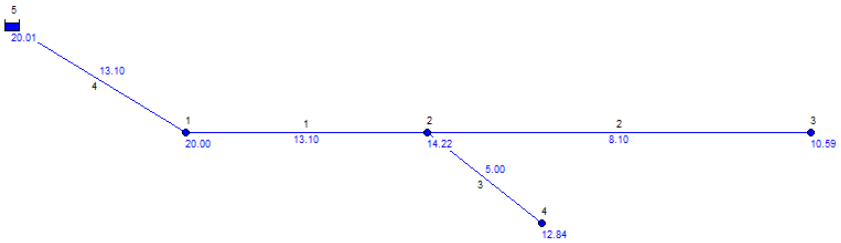


Figure 5.3: Example Epanet network analysis. Head pressure can be seen at each node, and flow on the edges.

The hydraulic head loss in Epanet can be calculated with three formulas: Hazen-Williams, Darcy-Weisbach or Chezy-Manning [40]. As the steady-state analysis in this paper is similar to the Darcy Weisbach equation, this will be chosen as a reference. More information on the hy-

draulic models in Epanet can be found in [41]. When a network is modelled in Epanet, several parameters must be adjusted. Epanet assumes that all pipes are full at all times. There has to be a reservoir, i.e., a source of water to supply the network. A pump can be connected to the reservoir, or the reservoir can be raised above the network to provide head pressure. The latter will be used in this project.

Below are the parameters for edges and nodes (pipes and junctions). *The list does not fully cover all of Epanet's parameters*.

- **Node Input**
  - Base demand. This is a parameter for the flow demand of the junction. $[l/s]$.
  - Elevation. To set the height of each node, the default is 0. $[m]$
  - Demand Pattern. The demand at each node for every hour of the simulation can be pre-configured, enabling extended time-based analysis
- **Node Output**
  - Actual demand. Actual flow demand to node. $[l/s]$
  - Total head. $[m]$
  - Pressure. If elevation is set to zero; head and pressure will be equal. $[m]$
- **Edge Input**
  - Length. Length of each edge. $[m]$
  - Diameter. Inner diameter of pipe. $[mm]$
  - Roughness. Parameter to determine headloss in pipes. $[mm]$
- **Edge Output**
  - Flow. Flow through the section. $[l/s]$
  - Unit Headloss. Headloss through the Edge. $[m]$
  - Friction factor. $[-]$
- **Reservoir Input**
  - Total head. Parameter to adjust the head to network. $[m]$
- **Reservoir Output**
  - Net Inflow. The sum of flows into the reservoir. Negative if water flows out of the reservoir. $[l/s]$
  - Elevation. Reservoir elevation is calculated based on the total head. $[m]$

5.5.1 Comparing Friction Factor f from Epanet with Friction Parameter θ from Steady State Analysis

Epanet can return the friction factor of each node, while the developed algorithm will return the friction parameter θ. Therefore, a formula to compare the calculated friction parameter θ and the simulated friction factor f is established. The Darcy Weisbach equation Equation (3.11) and θ for a single section of pipe Equation (4.7) is used, and the two equations can be seen below.

$$\Delta H = \frac{f\ \Delta x\ Q^2}{2\ g\ D\ A^2} \tag{5.2a}$$

$$\Delta H = \theta\ L\ (Q)^2 \tag{5.2b}$$

where $\Delta x = L$ and $a_1 = gA$. The two equations are put together and solved for f:

$$\begin{aligned} \frac{f\ L\ Q^2}{2\ g\ D\ A^2} &= \theta\ L\ (Q_1)^2 \\ \frac{f}{2\ g\ D\ A^2} &= \theta \\ f &= 2\ g\ D\ A^2\ \theta \end{aligned} \tag{5.3}$$

where D is the diameter of each pipe and A is the area of the cross-section.

5.6 Orkanger Municipality Network Analysis

As part of the project, Orkanger Municipality has sent the water distribution network as *.INP* file. The network file is to be analysed with the purpose of finding opportunities for the algorithm introduced in Chapter 5. As previously seen in Section 3.7, the network file contains information about junctions and pipes stored in keywords. Junctions will be represented as nodes V, and pipes will be represented as edges E. After the network file is imported to Matlab, the networks will be stored as *graph-objects* for further analysis. The result is presented in Section 7.1.

Chapter 6

Recreation of Leak Detection Algorithm

This chapter consist of the methodology of the leak detection algorithm from Carrera and Verde [7]. The leak detection algorithm is recreated for the purpose of testing data from Orkanger Municipality. The main ideas and models are presented and implemented in this chapter. In addition, a section is dedicated to introduce the test data from Orkanger Municipality.

6.1 Transient Flow Model

The network tree for the system model can be seen in Figure 6.1, L_e is the unknown distance from node 1 to the point of the leak.

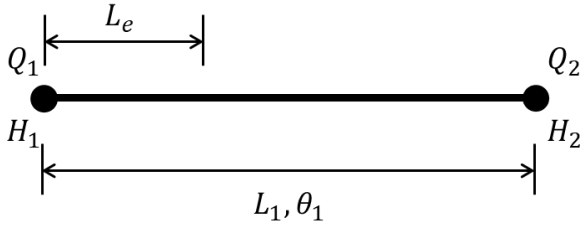


Figure 6.1: One section of pipe, unknown leak location L_e.

The system model is based on the two PDEs from Section 3.2.4, Equation (3.9):

$$\frac{1}{a_1}\frac{\partial Q(z,t)}{\partial t} + \frac{\partial H(z,t)}{\partial z} + J(Q(z,t)),\theta) = 0 \tag{6.1a}$$

$$\frac{\partial H(z,t)}{\partial t} + a_2\frac{\partial Q(z,t)}{\partial z} = 0 \tag{6.1b}$$

where $a_1 = g\ A_r$ and $a_2 = \frac{a}{a_1}$. The hydraulic gradient is chosen to be:

$$J(Q(z,t)),\theta) = \theta\ Q(z,t)^2 \tag{6.2}$$

where θ is the friction parameter for the section of pipe. It is assumed that measurements of head pressure and flow rates at the inlet and outlet of the section (H_1, H_2, Q_1, Q_2) are known or available. The pipe section is broken into two sections, first from node 1 to L_e, and from node 2 to V_2. The partial derivatives of each section are numerically approximated with theory from (Section 3.1.2) over the length of the two sections, and three equations remain:

$$\frac{1}{a_1}\frac{\partial Q_1(t)}{\partial t} + \frac{H_{L_e}(t) - H_1(t)}{L_e} + J(Q_1(t)), \theta) = 0 \tag{6.3a}$$

$$\frac{\partial H_e(t)}{\partial t} + a_2 \frac{Q_2(t) + Q_{L_e}(t) - Q_1(t)}{L_e} = 0 \tag{6.3b}$$

$$\frac{1}{a_1}\frac{\partial Q_2(t)}{\partial t} + \frac{H_2(t) - H_{L_e}(t)}{L - L_e} + J(Q_2(t)), \theta) = 0 \tag{6.3c}$$

The time-derivatives are isolated at the left-hand side and inserting the hydraulic gradient from Equation (6.2) into the system model from Equation (6.3):

$$\dot{Q}_1(t) = \frac{a_1}{L_e}\left(H_1(t) - H_{L_e}(t)\right) - a_1\theta\ Q_1(t)^2 \tag{6.4a}$$

$$\dot{H}_{L_e}(t) = \frac{a_2}{L_e}\left(Q_1(t) - Q_2(t) - Q_{l_e}(t)\right) \tag{6.4b}$$

$$\dot{Q}_2(t) = \frac{a_1}{L - L_e}\left(H_{L_e}(t) - H_2(t)\right) - a_1\theta\ Q_2(t)^2 \tag{6.4c}$$

6.2 Estimating Friction Factor Parameter θ

An adaptive observer is used to estimate the friction parameter θ for the section. The system model in Equation (6.4) is rewritten to the canonical form (Equation (3.21)), defining input u, output y and unknown parameters to be estimated Θ as:

$$y = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}, \quad u = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}, \text{and:} \quad \Theta = \begin{bmatrix} \theta \\ H_{L_e} \end{bmatrix} \tag{6.5}$$

returning the system model for the adaptive observer:

$$\dot{y} = \underbrace{\begin{bmatrix} \frac{a_1}{L_e}u_1 \\ -\frac{a_1}{L-L_e}u_2 \end{bmatrix}}_{\alpha(y,u,t)} + \underbrace{\begin{bmatrix} -a_1 y_1^2 & -\frac{a_1}{L_e} \\ -a_1 y_1^2 & \frac{a_1}{LL_e} \end{bmatrix}}_{\beta(y,t)}\Theta \tag{6.6}$$

The friction parameter θ is then estimated with adaptive observer from Equation (3.22):

$$\begin{aligned} \dot{\hat{y}} &= \alpha(y,u,t) + \beta(y,t)\hat{\Theta} - k_y(\hat{y} - y) \\ \dot{\hat{\Theta}} &= -k_\Theta \beta^T(y,u,t)(\hat{y} - y)^T \end{aligned} \tag{6.7}$$

where $k_y > 0$, $k_\Theta > 0$ and where $||\hat{\Theta} - \Theta|| \to 0$ if β is persistently exciting and $\dot{\beta}$ is bounded.

6.3 Estimating Leak Location L_e

The purpose of this section is to introduce an EKF (Section 3.5.2) for estimation of the leak location L_e given a constant friction parameter θ. The state x, inputs u and measurements y are defined as:

$$x = \begin{bmatrix} Q_1 \\ Q_2 \\ H_{L_e} \\ L_e \end{bmatrix}, \quad u = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}, \text{and:} \quad y = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \tag{6.8}$$

Further, it is assumed that the dynamic of the pressure at leak location, H_{L_e}, is slower than the dynamic of the flow measurements Q_1 and Q_2 ($\dot{H}_{L_e=0}$) Carrera and Verde [7]. Giving the augmented system model based on Equation (6.4) and state $L_e = 0$:

$$\dot{x} = \begin{bmatrix} \frac{a_1}{x_4}(u_1 - x_3)) - a_1\theta x_1 \\ \frac{a_1}{L-x_4}(x_3 - u_2)) - a_1\theta x_2 \\ 0 \\ 0 \end{bmatrix}, \text{and:} \quad y = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{C} x \tag{6.9}$$

This model is the basis for the EKF:

$$\dot{\hat{x}} = f(\hat{x}(t), u(t)) + K(t)[y(t) - C\hat{x}(t)] \tag{6.10}$$

where K is calculated from the solution of Equation (3.32a) as described in Section 3.5.2.

6.4 Leak Detection Algorithm

The previous sections give the basis for the leak detection algorithm from Carrera and Verde [7]; this section will introduce how the algorithm for estimating the location of leak L_e is organized. A flowchart of the algorithm can be seen in Figure 6.2. Conversion of analogue data in discrete time is neglected in this flowchart. However, more information is found in the report Carrera and Verde [7].

The algorithm will estimate the friction parameter with the adaptive observer introduced in Section 6.2. Every iteration, the leak algorithm will determine if there is a leak detected based on the difference in flow measurements:

$$||Q_1 - Q_2|| < \varepsilon_1 \tag{6.11}$$

where ε_1 is a tunable threshold parameter. If a leak is detected, the friction parameter θ is locked, and the EKF is started to estimate the leak location L_e. When the error of the measured states and the estimated states is below a threshold parameter ε_2 the leak location is returned:

$$\begin{aligned} e_1 &= ||Q_1 - \hat{Q}_1|| \\ e_2 &= ||Q_2 - \hat{Q}_2|| \\ e_1 + e_2 &< \varepsilon_2 \end{aligned} \tag{6.12}$$

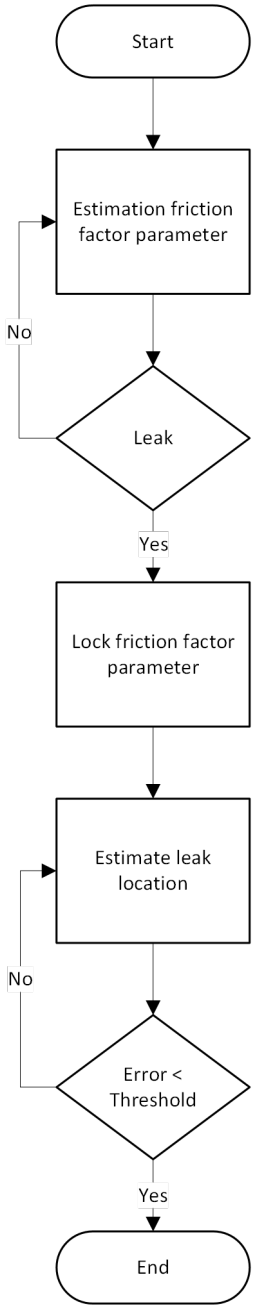


Figure 6.2: Flow chart leak detection algorithm.

6.4.1 Initialization and Tuning Parameters

The adaptive observer and the EKF must be tuned for optimal performance. Regarding the adaptive observer, k_y and k_θ are tunable parameters. In addition, the states have to be initialized.

The EKF also has parameters that need tuning. Covariance matrices Q and R must be tuned after performance. In addition, the parameter $\eta > 0$, and states initialized.

The remaining algorithm parameters can be seen in Table 6.1. Matlabs built-in solver for differential equations, ode45 [42], is used.

Parameter	Value	Unit	Description
L	163.15	m	Overall length
D	0.076	m	Diameter of inner pipe
Δ H	2.25	m	Height difference H_1 and H_2
g	9.81	m/s^2	Gravitational acceleration
a	1330	m/s	Pressure wave velocity

Table 6.1: Parameter for experiment data.

6.5 Preparation of Data from Orkanger Municipality for Leak Detection Algorithm

Orkanger municipality has agreed to test the leak detection algorithm with data from a water network section stretching from Orkanger sentrum (Nyhavna) to Kjøra. This section will be considered as one section. A simplified map of the network can be seen in Figure 6.3.

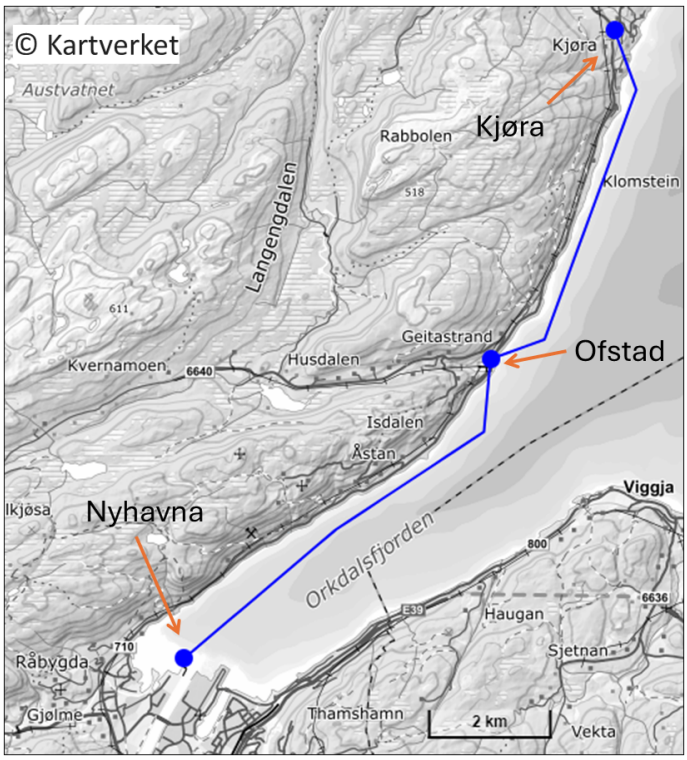


Figure 6.3: Simplified network map; Nyhavna - Ofstad - Kjøra. Blue lines illustrates pipes. Map from Kartverket [43].

The section starts at Nyhavna, where head pressure and flow measurements are measured (Q_1 and H_1). From Nyhavna, the pipe is submerged until it reaches Ofstad; here, the line goes onshore, and a branch is led off to supply the residents living there. Both head pressure and flow are measurements at Ofstad. The main pipe continues submerged from Ofstad until it reaches Kjøra, with a new set of sensors measuring head pressure and flow (Q_2 and H_2). All available measurements can be seen in the simplified Figure 6.4.

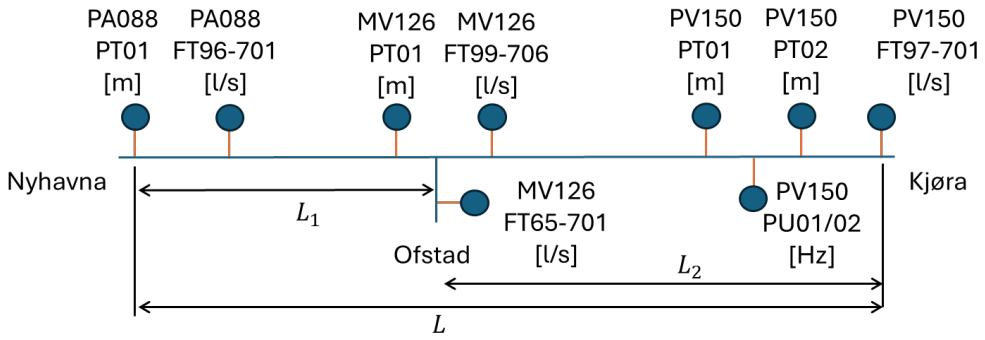


Figure 6.4: Nyhavna - Kjøra, instrumentation diagram.

Not all measurements seen in Figure 6.4, are available, and will not be used in the leak detection algorithm. The branch at Ofsted will be treated as a leak (Q_{L_e}), with a known distance L_1 from Orkanger. Table 6.2a shows what sensors from Orkanger Municipality that are used in the algorithm and their respective variable names. Δh is the height difference between Nyhavna sensors and Kjøra sensors.

Sensor name	Variable name	Location
PA088 PT01	H_1	Nyhavna
PA088 FT96-701	Q_1	Nyhavna
PV150 PT01	H_2	Kjøra
PV150 FT97-701	Q_2	Kjøra
MV126 FT65-701	Q_{L_e}	Ofstad

(a) Orkanger measurements.

Parameter	Value	Unit
L_1	8 196	m
L_2	6 567	m
L	14 764	m
D	140	mm
Δh	8	m

(b) Parameters Orkanger.

Table 6.2: Variables and parameters Orkanger.

Chapter 7

Results

The result chapter is divided into three sections; The first section will focus on the algorithm developed in Chapter 5, where different water network trees will be used as input, simulation data generated and compared to the least square estimates. The second section is devoted to the results of the investigation of the Orkanger water network files. The third section will demonstrate tuning and testing of the dynamic leak detection algorithm discussed in Chapter 6.

7.1 Parameter Identification Matrices from Network Tree Algorithm

The algorithm for the automated setup of water network trees is tested on cases solved theoretically in Chapter 4, and the data is compared to simulated values from Epanet. The simulated flow and head values is used to populate the matrices and calculate a least square estimate of the friction parameter. Three cases will be discussed, case 1 (with two data from two sources), case 2 and case 5.

7.1.1 Case 1: Numerical Values to Calculate Friction Parameter Based on Experiment Data

As seen in Chapter 6, the Verde test experiment and the Orkanger network trees may be characterised as case 1, with three nodes and two edges. The deduction of the network in Section 4.2 showed that there cannot be one unique friction parameter pr edge. The system matrices, in this case, will be one single equation. The friction parameter θ is calculated with said Equation (4.7); thereafter, a model of case 1 is constructed in Epanet with matching data from the experiment.

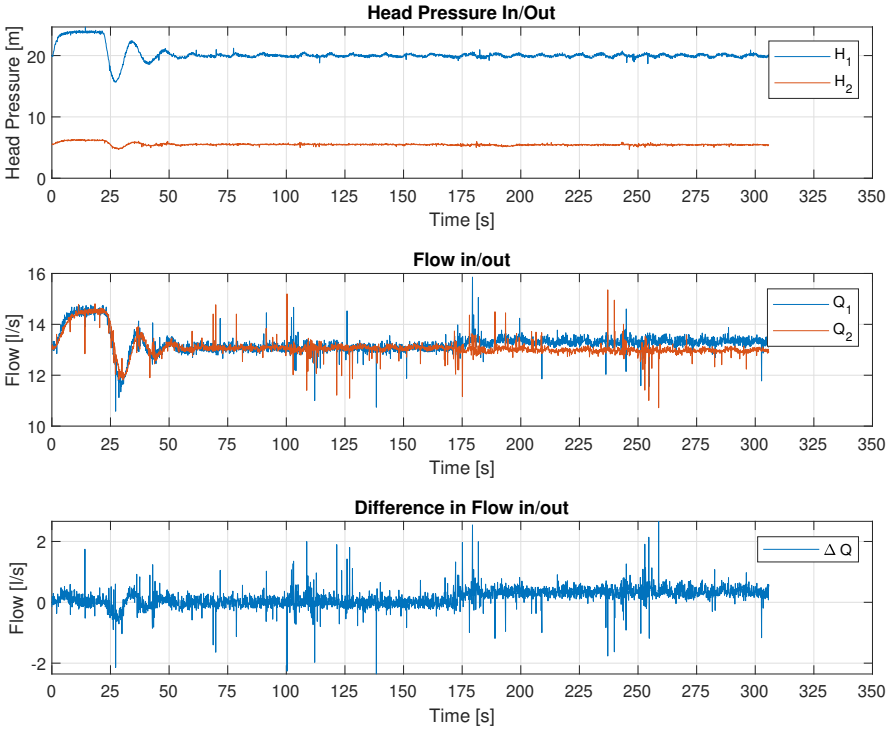


Figure 7.1: Experiment flow and head pressure data: Q_1 and H_1 measured at inlet, Q_2 and H_2 measured at outlet.

The friction parameter is calculated using average flow ($\overline{Q}_1$) and head ($\overline{H}_1$ and $\overline{H}_2$) from $t = 75$ to $t = 100$ where the flow is reasonably stable. The experiment data was collected at 10 Hz, making it 250 measuring points that were used in the calculation. To have comparable data between manual calculations and Epanet, the friction factor f is calculated with Equation (5.3). A plot of the experiment data can be seen in Figure 7.1.

$$\overline{Q}_1 = 13.0861 \cdot 10^{-3} \quad [m^3/s]$$
$$\overline{H}_1 = 20.0143 \quad [m]$$
$$\overline{H}_2 = 5.5116 \quad [m]$$
$$\theta = 517.2955 \quad [s^2/m^6] \tag{7.1a}$$
$$f = 0.0159 \quad [-] \tag{7.1b}$$

The network is modelled in Epanet with similar values for flow and head to calculate the friction factor. The simulated network can be seen in Figure 7.2. A reservoir is added in front of the section and head pressure at node 1 is adjusted to $\overline{H}_1$. The demand on node 3 is set to $\overline{Q}_1$. Other parameters, such as the lengths and inner diameter of the pipe, are also replicated with the actual values from Table 6.1. The head pressure at node 3 is lastly adjusted with the *roughness* parameter. Both edges 1 and 2 are set with the same value of roughness. The head loss over edge 4, the section from the reservoir to node 1, is minimized by setting roughness to zero and having a large diameter pipe.

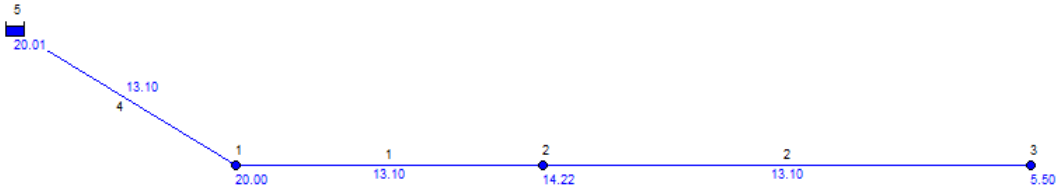


Figure 7.2: Case 1: Epanet model of Verde experiment.

With these parameters set, Epanet calculates a friction factor $f = 0.016$, which is similar to the manual calculation in Equation (7.1). The result will be discussed in Chapter 8.

7.1.2 Case 1: Numerical Values to Calculate Friction Parameter Based on Orkanger Data

The data from Orkanger can be seen in Figure 7.3. Due to data quality issues, the overall setup explained in Chapter 6, could unfortunately not be used. This section will shortly explain the issues with the data.

The data received for head H_2 (MV126 PT01) is right before the branch to Ofstad, and not at the Kjøra location. The data from flow sensor Q_2 is placed right after the branch to Ofstad (MV126 FT99-706); the actual water flow at Kjøra is unknown. Looking at the plot of the data Figure 7.3, H_2 is about 10 m larger than H_1. That is not correct, as the pressure should drop, not increase, over the section from Nyhavna to Ofstad. The finding was shared with Orkanger to see if the sensors had been calibrated correctly. A bias was discussed as the reason for the higher head at Ofstad, however, while the head sensor at Nyhavna (PA088 PT01) and the head sensor at Kjøra (PV150 PT01) are located at $\Delta h = 8$, the head sensor at Ofstad (MV126 PT01) is approximately at $\Delta h = 0$. As the sensors are both placed at the same height it is not the

Sensor name	Variable name	Location
PA088 PT01	H_1	Nyhavna
PA088 FT96-701	Q_1	Nyhavna
MV126 PT01	H_2	Ofstad
MV126 FT99-706	Q_2	Ofstad
MV126 FT65-701	Q_{L_e}	Ofstad

Table 7.1: Orkanger measurements updated.

reason for the bias.

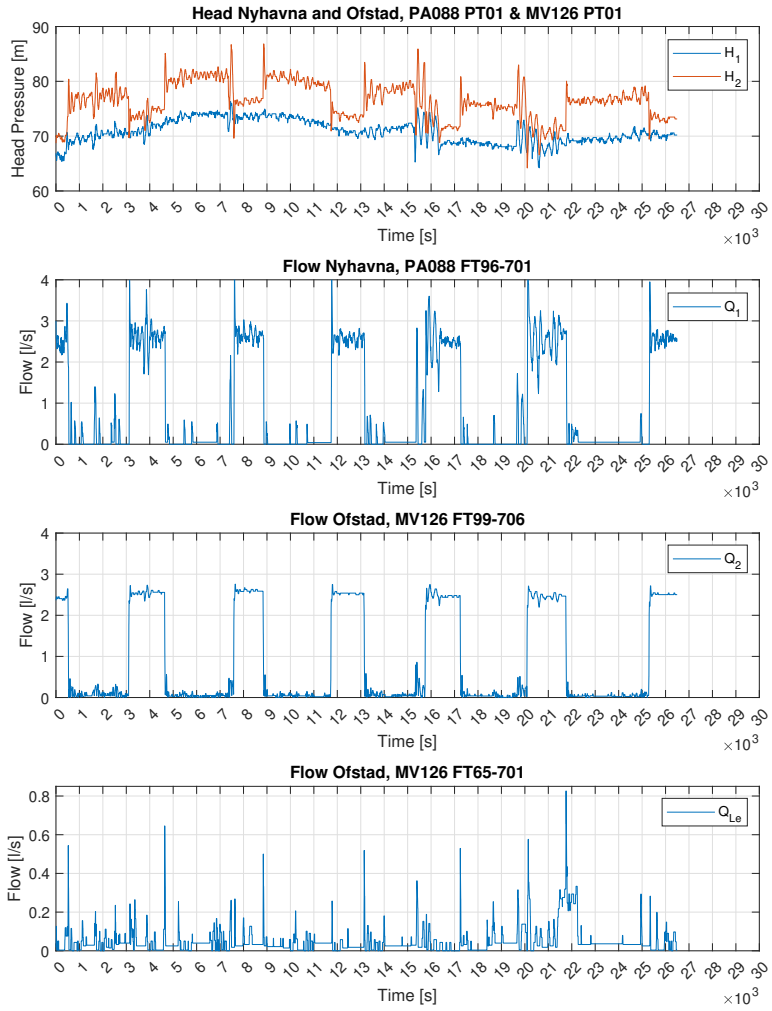


Figure 7.3: Data Orkanger: Flow and head pressure.

Looking at Figure 7.3, it can be seen that the flow Q_1 is changing from around zero l/s to between $2.5 - 3l/s$ with a recursive interval of approximately 2500 seconds lasting about 1000 seconds each. This behaviour was discussed with Orkanger representatives and they explained that it was due to a filling schedule for a height reservoir used to keep pressure and supply for consumers at Kjøra. When the reservoir is filled, the head H_2 at Ofstad drops with approximately 5m. This is reasonable behaviour and comperable to the behaviour in the experiment data in Figure 7.1. The head H_1 at Orkanger, on the other hand, is without a head drop. This indicates that H_1 and H_2 are decoupled, while there should, in reality, have been a significant head drop also at H_1 when the filling of the reservoir starts. This was also discussed with representatives from Orkanger, and it was investigated at the location; There is installed a check valve right after the sensor, before the pump that supplies the network. Therefore, H_1 does not read the correct head. A check valve is a valve where the flow can only pass one way.

Other alternative head transmitters are considered at the time this thesis was written.

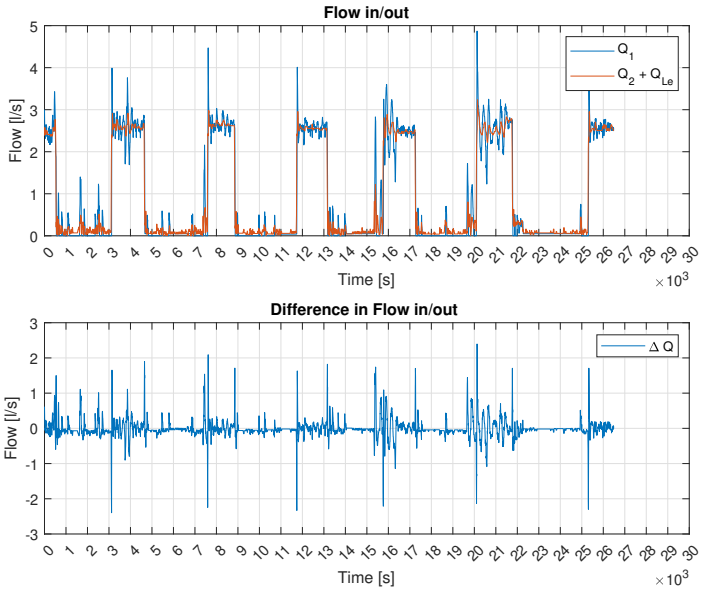


Figure 7.4: Data Orkanger: Flow in/out.

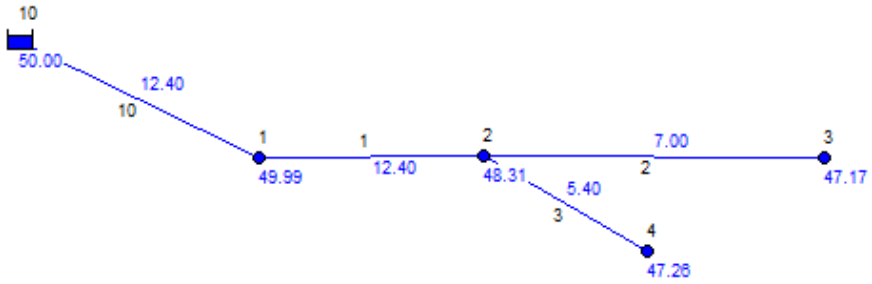
7.1.3 Case 2 Verification

Case 5 introduced in Section 4.3 is modelled in Epanet and exported as a network file. The network file is imported to Matlab. The roughness on each edge is set to 0.1, and the diameter of the sections is set to 110mm. The demand on end nodes is set according to Table 7.2:

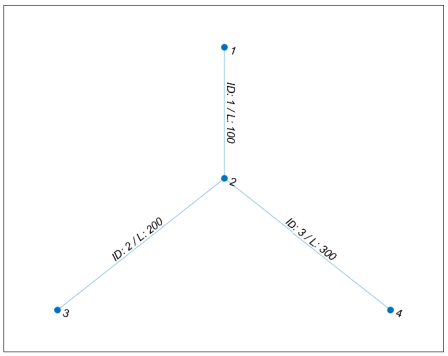
Node ID	Demand $[l/s]$
3	7.0
4	6.0

Table 7.2: Initial flow demand case 2.

After running the algorithm the graph and network matrices in Figure 7.5 are produced.



(a) Epanet model, with flow on edges, and head on nodes.



(b) Network tree graph model Matlab.

```
A =
[C_1*Q_1^2, C_2*Q_2^2,          0]
[C_1*Q_1^2,          0, C_3*Q_3^2]

Theta =
    {[theta_1]}
    {[theta_2]}
    {[theta_3]}

H =
    {[H_1 - H_3]}
    {[H_1 - H_4]}
```

(c) Resulting Network Matrices.

Figure 7.5: Case 2: Epanet model, parameter matrices and graph plot.

The values of $C_\#$ in Figure 7.5c are equivalent to $L_\#$ for each edge; the numerical values are stored in the graph object for each of the edges so that they can be retrieved when needed. As seen in Section 4.3, with only one set of data, the matrice A will be rank-deficient. That is also confirmed by numerical calculations with flow figures from Table 7.2; The rank of $A = 2$ with only one dataset, A has size (2×3). It is evident that the maximal rank is 2. Therefore,

the least square solution of Θ must be used. A^1 and A^2 matrices can be seen below:

$$A^1 = \begin{bmatrix} 0.0154 & 0.0098 & 0 \\ 0.0154 & 0 & 0.0087 \end{bmatrix}, \quad A^2 = \begin{bmatrix} 0.0154 & 0.0098 & 0 \\ 0.0154 & 0 & 0.0087 \\ 0.0156 & 0.0092 & 0 \\ 0.0156 & 0 & 0.0097 \end{bmatrix} \tag{7.2}$$

To have varying data from Epanet to be used in the parameter calculations, demand patterns are constructed. The patterns used in Epanet for simulating case 2 can be seen in Table 7.3. The base demand of the node is multiplied by the multiplier at each step. Node 1 gets multiplier pattern 1, and node 2 gets multiplier pattern 2.

Time Period	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Multiplier pattern 1:	1.0	0.97	0.95	0.92	0.90	0.95	1.02	1.05	1.1
Multiplier pattern 2:	0.9	0.95	1.02	1.05	1.1	1.0	0.97	0.95	0.92

Table 7.3: Demand pattern table for nodes.

The resulting friction parameter $\bar{\Theta}$ and friction factor f with 9 sets of data ($k = 9$) are:

$$\bar{\Theta} = \begin{bmatrix} 113.5238 \\ 108.5263 \\ 111.3286 \end{bmatrix}, \quad f = \begin{bmatrix} 0.0221 \\ 0.0212 \\ 0.0217 \end{bmatrix} \tag{7.3}$$

Figure 7.6 shows the plot of $\bar{\Theta}$ and friction factor f with growing number of data set. The last plot in the figure shows the friction factor derived from Epanet. As seen on the last plot the friction factor from Epanet is changing with flow and head. The result will be discussed in Chapter 8.

$$f_{Epanet} = \begin{bmatrix} 0.0213 \\ 0.0223 \\ 0.0232 \end{bmatrix} \tag{7.4}$$

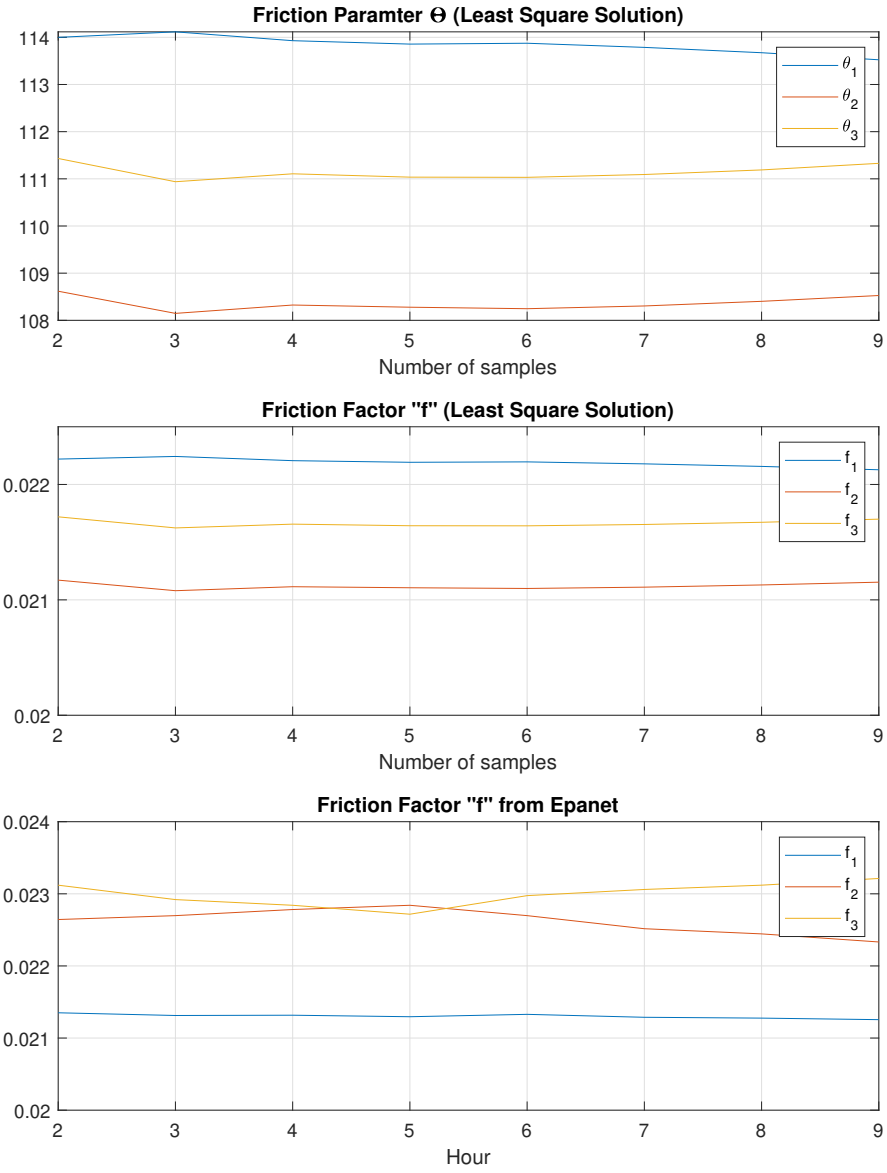


Figure 7.6: Case 2: Friction parameter and friction factor.

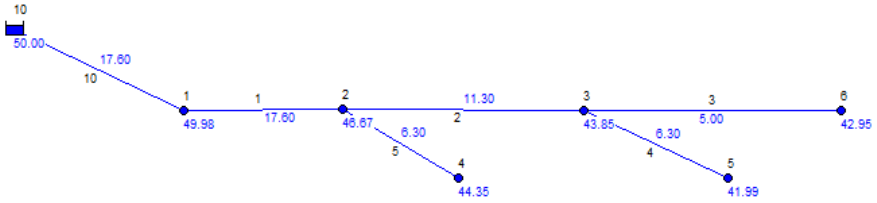
7.1.4 Case 5 Verification

Case 5 introduced in Section 4.6 is also modelled in Epanet and exported as a network file. The network file is imported to Matlab. The roughness on each edge is set to 0.1, and the diameter of the sections is set to 110mm. The demand on end nodes is set according to Table 7.4:

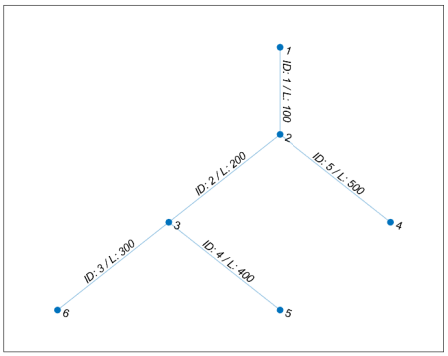
Node ID	Demand $[l/s]$
4	7.0
5	6.0
6	5.0

Table 7.4: Initial flow demand.

After running the algorithm the graph and network matrices in Figure 7.7 are produced.



(a) Epanet model, with flow on edges, and head on nodes.



(b) Network tree graph model Matlab.

```
A =
[C_1*Q_1^2, C_2*Q_2^2, C_3*Q_3^2,          0,          0]
[C_1*Q_1^2, C_2*Q_2^2,          0, C_4*Q_4^2,          0]
[C_1*Q_1^2,          0,          0,          0, C_5*Q_5^2]

Theta =
    {[theta_1]}
    {[theta_2]}
    {[theta_3]}
    {[theta_4]}
    {[theta_5]}

H =
    {[H_1 - H_6]}
    {[H_1 - H_5]}
    {[H_1 - H_4]}
```

(c) Resulting network matrices.

Figure 7.7: Case 5: Epanet model, parameter matrices and graph plot.

The values of $C_{\#}$ in Figure 7.7c are equivalent to $L_{\#}$ for each edge; the numerical values are stored in the graph object for each of the edges so that they can be called upon if needed. As seen in Section 4.6, with only one set of data, the matrice A will be rank-deficient. That is also confirmed by numerical calculations with flow figures from Table 7.4; The rank of $A = 3$ with only one dataset, A has size (3×5). Therefore, the least square solution of Θ must be used.

$$A^1 = \begin{bmatrix} 0.0310 & 0.0255 & 0.0075 & 0 & 0 \\ 0.0310 & 0.0255 & 0 & 0.0159 & 0 \\ 0.0310 & 0 & 0 & 0 & 0.0198 \end{bmatrix} \tag{7.5}$$

To have varying data over to get more datasets, demand patterns are constructed. The patterns used in Epanet for simulating case 5 can be seen in Table 7.5. The base demand of the node is multiplied by the multiplier at each step. Node 1 gets multiplier pattern 1, node 2 gets multiplier pattern 2 and node 3 gets multiplier pattern 3.

Time Period	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Multiplier pattern 1:	1.0	0.97	0.95	0.92	0.90	0.95	1.02	1.05	1.1
Multiplier pattern 2:	0.9	0.95	1.02	1.05	1.1	1.0	0.97	0.95	0.92
Multiplier pattern 3:	1.05	1.1	1.0	0.97	0.95	0.92	0.9	0.95	1.02

Table 7.5: Demand pattern table for nodes.

The resulting friction parameter $\bar{\Theta}$ and friction factor f with 9 sets of data ($k = 9$) are:

$$\bar{\Theta} = \begin{bmatrix} 112.4049 \\ 106.3710 \\ 110.7014 \\ 112.6397 \\ 108.4459 \end{bmatrix}, \quad f = \begin{bmatrix} 0.0219 \\ 0.0207 \\ 0.0216 \\ 0.0220 \\ 0.0211 \end{bmatrix} \tag{7.6}$$

Figure 7.8 shows the plot of $\bar{\Theta}$ and friction factor f with growing number of data set. The last plot of the figure shows the friction factor derived from Epanet. As seen on the last plot the friction factor from Epanet is changing with flow and head. The result will be discussed in Chapter 8.

$$f_{Epanet} = \begin{bmatrix} 0.0208 \\ 0.0215 \\ 0.0232 \\ 0.0229 \\ 0.0228 \end{bmatrix} \tag{7.7}$$

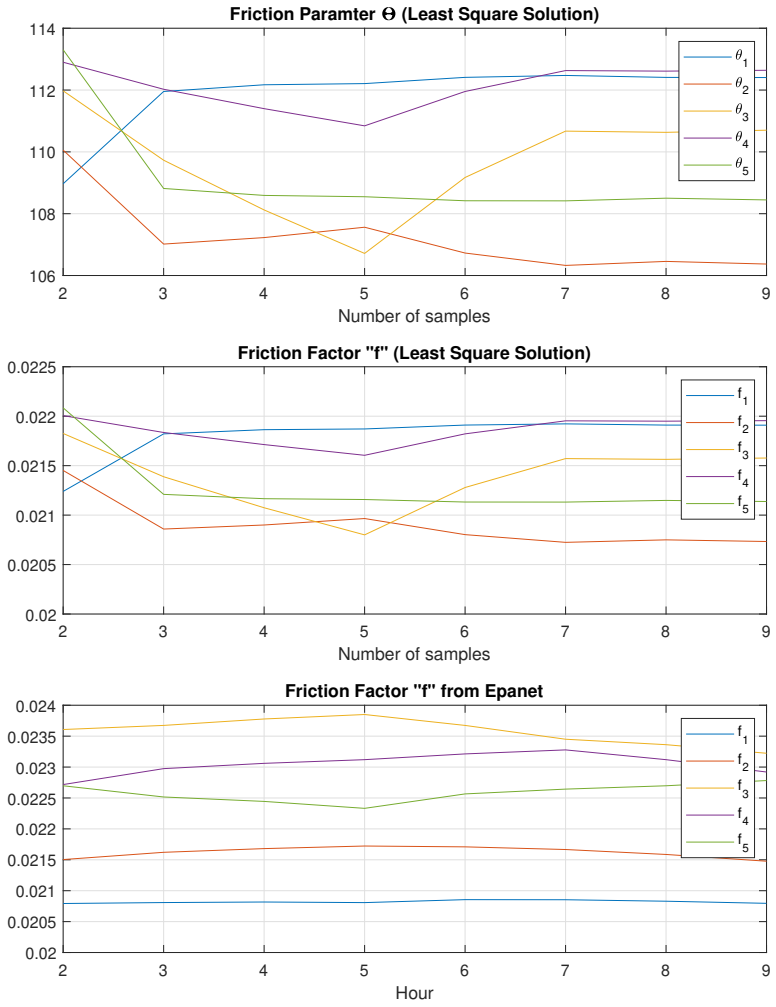
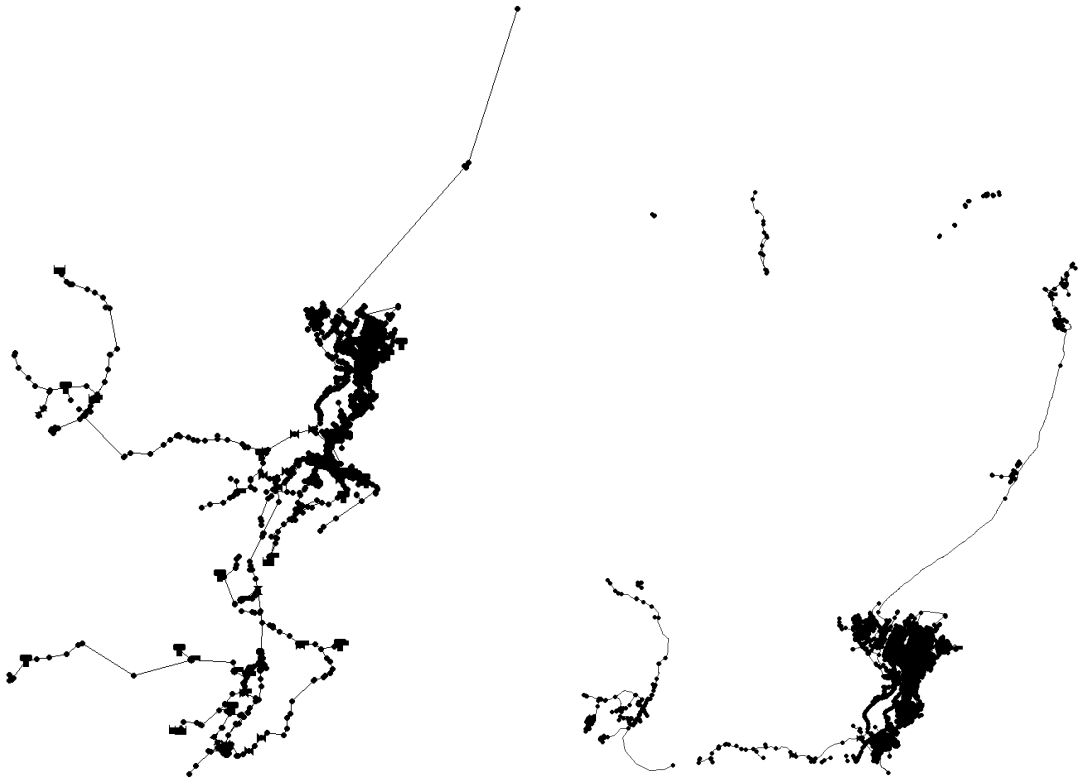


Figure 7.8: Case 5: Friction parameter and friction factor.

7.2 Orkanger Municipality Water Network

The Orkanger Municipal water network is imported to Matlab and plotted as a *graph object*. Two network files are received and investigated; the two files will be referred to as File 1 and File 2, respectively.

There were significant differences between the two networks from the two files. File 1 contained $|V_1| = 2233$ nodes and $|E_1| = 2435$ edges. File 2 contained $|V_2| = 16522$ nodes and $|E_2| = 15868$ edges. Both files were opened in Epanet. The plots of both File 1 and File 2 water networks can be seen in Figure 7.9. The black circles are junctions and the black lines are edges.

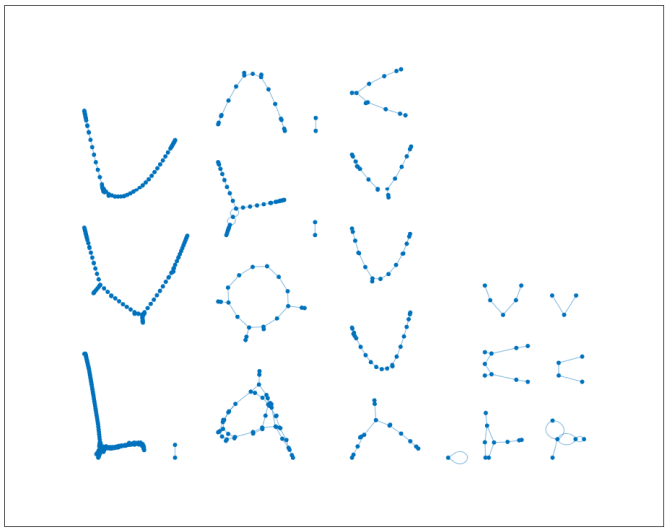


(a) File 1: Epanet water network plot.

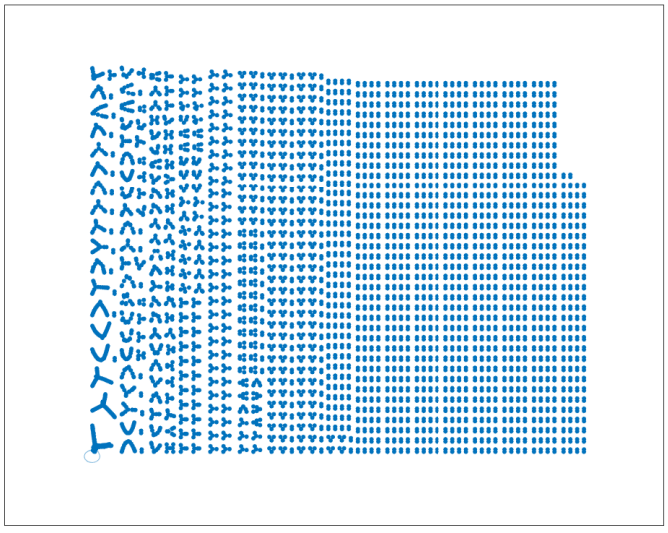
(b) File 2: Epanet water network plot.

Figure 7.9: Epanet plot of File 1 and File 2. There are similarities between the two water networks. The edge going out and upwards from the cluster is the section from Nyhavna to Ofstad and onwards to Kjøra.

The graph plot of the two figures can be seen in Figure 7.10a and Figure 7.10b. An observation when plotting File 2 was that it contained several more disconnected graphs compared to File 1. The graphs will be discussed in Chapter 8.



(a) File 1: The water network consists of a few non-connected graphs. The graph in the bottom left comprises most of the nodes. Some of the graphs are trees; some are graphs with cycles.



(b) File 2: The water network consists of *many* non-connected graphs. The graph in the bottom left comprises most of the nodes. Many of the graphs are trees that comprise only a few nodes.

Figure 7.10: Graph-object plots of the two water network files from Orkanger.

The two network files contained different *keywords* (Table 7.6). The keywords are in the correct order relative to the two network files.

File 1	File 2
Title	Title
Junctions	Options
Reservoirs	Junctions
Tanks	Pipes
Pipes	Pumps
Coordinates	Valves
Valves	Curves
Pumps	Coordinates
VSD_pumps	Vertices
Demands	
Pattern	
Status	
Controls	
Mixing	
Reactions	
Energy	
Times	
Report	
Options	

Table 7.6: File 1 and File 2 available keywords.

The available information found in the keywords for the two network files will be discussed in Chapter 8.

7.3 Dynamic Leak Algorithm

The leak detection algorithm was deployed using experiment data provided by Carrera and Verde [7]. The purpose is to tune and verify its basic performance. After the initial tuning, it was to be deployed using the data from Orkanger.

7.3.1 Verification and Tuning on Verde-experiment Data

The overall setup is explained in Chapter 6. As seen in Table 6.1, there is a height difference between the two head pressure sensors. This difference is compensated for by adding the height difference to the second measurement of head pressure: $H_2 = H_2 + \Delta H$. The data from the experiment is passed through a lowpass filter to remove high-frequency noise. After tuning, the gain for the adaptive observer was set to:

$$\begin{aligned} k_y &= 7 \\ k_\Theta &= 5.5 \cdot 10^9 \end{aligned} \tag{7.8}$$

The following covariance matrices and η were chosen for the EKF:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 \cdot 10^8 & 0 \\ 0 & 0 & 0 & 1 \cdot 10^8 \end{bmatrix}, \qquad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{and:} \quad \eta = 0.1 \tag{7.9}$$

Figures 7.11 to 7.13 shows the plots of the resulting data. The performance of the leak detection algorithm will be discussed in Chapter 8. The threshold parameter to decide if there was a leak (ε_2) is set to $1.5 \cdot 10^{-6}$.

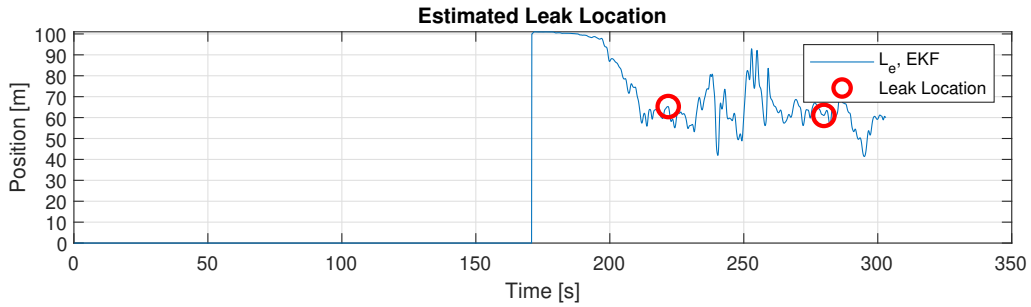


Figure 7.11: Leak detection algorithm: adaptive observer leak location

As seen on the plot in Figure 7.11, there are two occasions where the threshold is met and leak location is reported. That is, at $t = 221.8$ and $t = 279.9$. The respective location of the leaks are reported as $L_e = 65.275m$ and $L_e = 61.030m$. The actual location of the leak is at $65m$. As seen in Figure 7.13, at approximately $t = 170s$, the adaptive observer state values are frozen and passed to the EKF. At that time, the friction parameter was estimated to be: $\theta = 519$.

The EKF was zero until the leak was detected. The actual flow and head measurements are presented standalone in Figure 7.1. It can be seen that the leak was initiated at approximately $t \approx 170\ s$, as $\Delta Q > 0$. The leak detection was manually triggered by analysing the flow plots. The method suggested in Carrera and Verde [7] was not used.

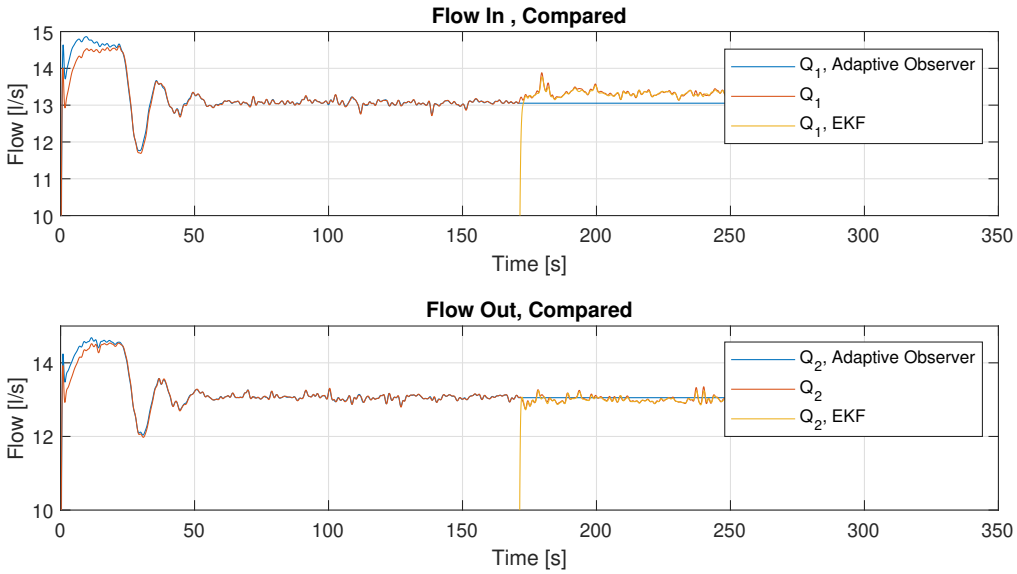


Figure 7.12: Leak Detection Algorithm: Flow data from adaptive observer and EKF compared in the same plot.

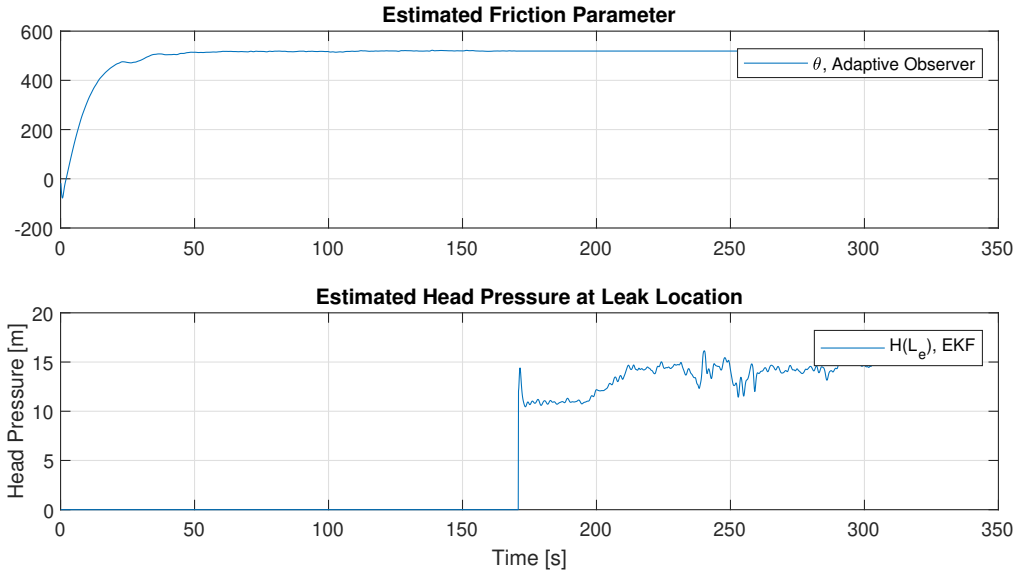


Figure 7.13: Leak detection algorithm: Estimated friction parameter θ from the adaptive observer and estimated head pressure at leak location from EKF.

7.3.2 Orkanger Data

Due to bad data quality, the leak detection algorithm could not be deployed using data from Orkanger at this point in time. The data is explained in Section 7.1.2.

Chapter 8

Discussion

This chapter will discuss the method and the results presented in previous chapters.

8.1 Friction Parameter Identification of Network Trees in Steady State

As seen in the Section 7.1, the network tree algorithm returns matrices for parameter identification for an imported network file. The matrices is built upon equations based on the simplified flow model, with the assumptions seen in Section 3.2.4. There are some requirements for the models that are imported that need to be complied with when modeling a network; All internal nodes must be of higher degree than two ($d(V) > 2$). If the degree is one, it is either a leaf or a root. If the degree is two the produced A matrix will not be able to achieve rank equal to number of edges. A proposal to avoid the issue is for the algorithm to recognize such nodes and combine edges to form a single section. The implication of combining sections has not been investigated further, one would expect that the sections shall have equal inner diameter, as the inner diameter of the sections is used to calculate friction factor f. That is something that needs investigated in further work.

No cycles in the graph is another important requirement. As the algorithm is designed for rooted network trees, it is a prerequisite for the algorithm. However, this version of the network tree algorithm does not check for cycles. Therefor it is important to be made aware of that the results with a network file with cycles would be invalid. In addition to not having cycles, the network tree must only contain the nodes and edges to be parameter identified. For example, reservoirs and pumps must be removed from the network file or added after the matrices are produced.

As seen the resulting matrices is generated based on the network file. Therefor the naming convention (ID's) is important for the nodes and edges. If the ID's of the network model change, the resulting matrices will also change correspondingly. Therefore, it is advised that the network modelling must be done accurately and with the end product in mind. The matrices may, of course, be manipulated after they are produced. As the algorithm uses the shortest path from a *start node* to all leaves, it is important to declare the correct start node, other start nodes may result in invalid system matrices.

After the matrices are produced, sets of data, either measured in a real network or simulated values, should be used to verify that the A matrix's rank is bigger than the number of edges to ensure that the least square solution exists. The current algorithm requires that head and flow measurements exist for the root, and the leaves. As it stands, the A matrices have flow measurements from all edges implemented; these flow measurements are sums of flow measurements of the root and the leaves of the network and can be simplified. However, this must now be done manually by the user.

The implementation of the algorithm could most likely have been improved. A observation made during initial programming was that the edges in the graph-object produced by Matlab does not always have the same ID as the nodes and edge from the imported network file. Therefore, several lines of code are used to keep the IDs consistent from the imported network file, to the resulting matrices.

8.1.1 Verification of Friction Parameter Θ Case 1

The steady-state friction parameter was verified with three different network scenarios. The first case, showed in Section 7.1.1, had a similar friction factor, $f_{\text{Epanet}} = 0.016$, for the Epanet model, and $f_{\text{ss}} = 0.0159 \approx 0.016$ for the *steady-state* parameter model with input of average flow $\bar{Q}$ and head $\bar{H}$ from the experiment data. The friction parameter was $\theta_{\text{ss}} = 517.3\ [s^2/m^6]$. The friction parameter from the Adaptive Observer (θ_{dyn}) is slightly higher, with $\theta_{\text{dyn}} = 519\ [s^2/m^6]$. The corresponding friction factor f_{dyn} is calculated with Equation (5.3):

$$f_{\text{dyn}} = \theta_{\text{dyn}} \cdot 2gDA^2 = 0.0159 \tag{8.1}$$

corresponding to $f_{\text{dyn}} \approx 0.016$, which equals the friction factor of both steady-state calculated and Epanet.

Unfortunately, the data from Orkanger could not be used to calculate friction parameters for case 1. If there were data, it is expected that the friction parameter would not be equal to the one calculated with the Verde data as the parameters such as length, diameter, flow, and head are significantly different.

8.1.2 Verification of Friction Parameter Θ Case 2 and Case 5

The steady-state friction parameter calculation of case 2 and case 5 was also compared to friction factors from Epanet. However, unlike in case 1, there was no experiment data to compare.

Case 2 friction factor calculations from Epanet and the network algorithm did not give the exact same results; there were some differences. The size of the fault due to the difference is verified with Equation (3.11) by calculating head loss. Edge 3 of case 2 is chosen, as it had the largest difference in friction factor.

The same parameters were used on the edge, as was used in Epanet: L = 300m, D = 110mm, and Q = 5.7 l/s (with multiplier 2, step 1). With friction factor from edge 3, $f_{3,\text{ss}} = 0.217$, the corresponding head loss ($\Delta H_{3,\ \text{ss}}$) is:

$$\Delta H_{3,\ \text{ss}} = \frac{LfQ^2}{2gDA^2} = 1.0851\ m \tag{8.2}$$

and with friction factor from epanet beeing $f_{3,\ \text{Epanet}} = 0.0232$, the corresponding head loss ($\Delta H_{3,\ \text{Epanet}}$) is:

$$\Delta H_{3,\ \text{Epanet}} = \frac{LfQ^2}{2gDA^2} = 1.1602\ m \tag{8.3}$$

The difference between the two head losses is 0.0750 m, which is minimal. However, it will produce the wrong head pressure, which should be considered when modelling networks.

Similar comparisons for the friction factors in case 5 are not calculated in this thesis. However, as seen in Section 7.1.4, none of the calculated friction factors of the edges was equal to the friction factor modelled in Epanet; they were equal to the second decimal.

8.1.3 Magnitude of Chosen Data Used in Verification

The magnitude of the measurement used as flow and head to verify the network model in Epanet was based on the experiment data combined with the data from Orkanger. If the data from Orkanger was usable, it should be used to verify different network scenarios. Alternatively, research in the form of a literature review could be performed to find typical flow and head values for industrial water consumers or private households. The multiplier table used to simulate values in Epanet was choosen not to be larger then 1 ± 10 %. The reason was to keep the flow values relatively stable. However, that did result in some variation in friction factors that may been avoided if less variation in multiplier was chosen. The requirement for the flow values are for the rank of A to be equal to number of edges.

8.2 Orkanger Municipality Network files

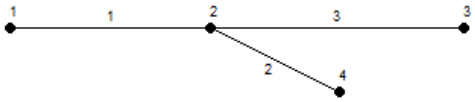
As seen in Section 7.2 the two network files received from Orkanger produced significantly different graphs. One reason to the significant difference in the number of nodes and edges between the two files is how the pipes have been modelled. For File 2, there are more curves to the edges; every curve is approximated by multiple straight segments, leading to more nodes and edges. Another difference between the two network files is the naming conventions. The node and edge names were different, which makes it difficult to compare sections. And it will make it hard to compare parameters if matrices were to be generated from the same sections. The keyword *Coordinates* contains the north-east coordinates of each node. The keyword exists in both File 1 and File 2. It may be used to compare the nodes between the network files. However, it has not been tested in this project.

The network files contained very different keywords, which, in turn, means that they contained different information. The steady-state network algorithm needs flow and head measurements from the tree's root and leaves to calculate the friction parameter Θ for the different edges in the sub-tree. As seen in Table 7.6, neither File 1 nor File 2 contained specific information about where/if sensors are placed. It is unsure if the files may be used to search for areas where the steady-state network algorithm may be used; another approach would be to use the experience of the representatives from Orkanger Municipality to find sections that may be used. Similar to the approach of the Orkanger-Kjøra section.

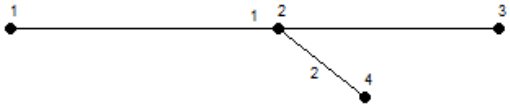
8.2.1 File 2: A large Number of Disconnected Graphs

The core issue of File 2 was the large number of disconnected graphs. This issue was investigated to find a cause. One of the disconnected trees in the graph produced by File 2 was used to find examples of node and edge IDs. File 2 was opened in Epanet to search for those IDs. It was discovered that adding a new node to an edge requires a specific process; simply adding the node to an existing edge to expand the network will produce a disconnected graph.

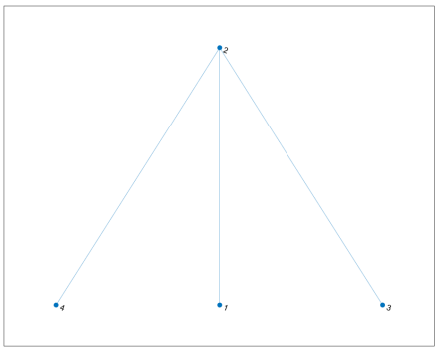
A simple network tree was created to verify the finding. When a node is added after the completion of an edge, it does not automatically join the edges to the new node. To connect the new node to the old edge, the edge has to be removed, and then the two new edges must be added. Figure 8.1 shows the simplified network trees and the graphs produced by the two trees using two different processes to add nodes.



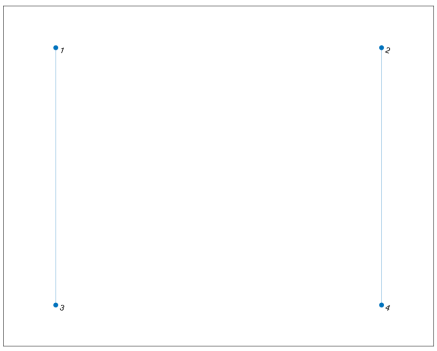
(a) Correct process of adding nodes Epanet.



(b) Incorrect process of adding nodes Epanet.



(c) Correct process of adding nodes graph. All nodes in one tree



(d) Incorrect process of adding nodes graph. The nodes are split into two trees

Figure 8.1: Two networks in Epanet produce different graphs.

8.3 Leak Detection Algorithm

The leak detection algorithm performs well when tuned properly. However, slightly changing parameters, such as changing the friction parameter from the one estimated by the adaptive observer (θ_{dyn}), to the friction parameter estimated with the steady state analysis (θ_{ss}), produced very different estimated leak positions.

Running the leak detection algorithm with the exact same tuning parameters for the covariance matrices Q and R in the Extended Kalman Filter, and not changing the threshold value ϵ_2, but with the friction parameter from the steady state calculations, the leak location algorithm gives an estimated leak location at $L_e = 75.38$ at $t = 221.8$.

Running the leak detection algorithm again with the friction parameter from steady state calculations ($\theta_{\text{dyn}} = \theta_{\text{ss}}$), but increasing Q_{44} from $1 \cdot 10^8$ to $1 \cdot 10^{10}$ gives an estimated leak location $L_e = 63.544$ at $t = 203.6$.

Divergence was also encountered when tuning both the adaptive observer and the extended Kalman filter. It is suggested to filter the data before using the leak detection algorithm, to remove high frequency noise. As the test data was limited to a single time series, the extent of the robustness of the tuning is unknown. It would be most interesting to test the leak detection algorithm on actual data from Orkanger.

Although the friction parameter calculated with the steady state simplified formula (θ_{ss}) did not perform adequatly when used in the EKF on the test data, it should be investigated further with more tests and different tuning; to possibly use the friction parameters calculated from steady state data on larger trees, without using the adaptive observer to estimate the friction parameters.

The threshold to detect leaks and transit from adaptive observer to the Extended Kalman filter for the leak detection algorithm was not thoroughly tested in this master project and will not be discussed.

8.3.1 Data from Orkanger

Unfortunately, for several reasons, no data from Orkanger could be used to test the dynamic leak detection algorithm. The data itself was explained in Section 7.1.2. It was attempted to use the data; one idea was to sum the flow from Kjøra and Ofstad and run the steady-state analysis and the Adaptive Observer to compare friction parameters. However, with head H_1 being lower than H_2, it would produce a negative friction parameter. Alternatively it was attempted to use the section from Ofstad to Kjøra to calculate the friction parameter. However, the outlet head H_2 at Kjøra was not available.

Chapter 9

Conclusion

This master's project aimed to identify friction parameters in tree-shaped water distribution networks with minimal measurements available. An algorithm was developed with basis in theoretical analysis of network trees and a simplified flow model. A prerequisite for the algorithm is that head and flow measurements are available at the root and the leaves. The algorithm successfully imports network files and generates system matrices where the friction parameters can be calculated. Verification of the calculated friction parameters with simulated friction factors were also conducted successfully. Suggestions for further development of the algorithm for automated parameter identification of network trees is to investigate how graphs with cycles can be included. Secondly it is suggested to include automated detection of nodes with degree 2, $(d(V)) = 2)$, such that the edges connected can be treated as one edge, and a friction parameter for that edge can be calculated. The last suggestion is to make general improvements to the implementation in Matlab.

Secondly, the master's project has conducted an analysis of Orkanger Municipal water network files to identify opportunities for friction parameters identification. As part of the investigation, the network files were imported into Matlab and processed as *graph-objects*. It is concluded that the network files need further processing before an algorithm can be deployed and used. Suggestions for further processing of the water network files include reducing the number of disconnected graphs in file 2. The next suggestion is to develop algorithms to identify and compare the IDs of nodes and edges in the two network files. It is suggested to collaborate with Orkanger Municipal to include sensor information in the network files so it is easily accessible.

Finally, the master's project recreated the leak detection algorithm from Carrera and Verde [7] and tuned it on experiment data. The results are promising, and the leak location was identified with acceptable performance. It is noted that the leak detection algorithm is sensitive to tuning parameters. The data from Orkanger was unfortunately unavailable. Suggestions for further work related with the leak detection algorithm include deploying the leak detection algorithm on data from Orkanger Municipal when it is available. Further it is suggested to collaborate with Orkanger Municipal to investigate if other sections of their water network meet the requirements for the leak detection algorithm.

Bibliography

[1] 'Lekkasjer,' Norsk Vann. (25th Aug. 2021), [Online]. Available: https://norskvann.no/ledningsnett-og-teknologi/lekkasjer/ (visited on 28/05/2024).

[2] M. Steinberg, C. F. Nordheim, T. M. Lyngstad and K. Janak, 'Rapportering av data for vannforsyningssystemer i Norge for 2019,' *Folkehelseinstituttet*, 2019. [Online]. Available: https://www.fhi.no/globalassets/dokumenterfiler/rapporter/2020/rapport-om-vannforsyning-2019/vannverksrapport-for-2019.pdf.

[3] A. Flatin, A. Unhjem and K. J. Sola, 'Experiences with leak detection and control,' *Norsk Vann BA*, no. 171, p. 43, 30th Jun. 2009, ISSN: 1890-8802. [Online]. Available: norskvann.no.

[4] Martin. 'Water and sanitation,' United Nations Sustainable Development. (), [Online]. Available: https://www.un.org/sustainabledevelopment/water-and-sanitation/ (visited on 28/05/2024).

[5] European Commission. Directorate General for the Environment., *EU reference document good practices on leakage management WFD CIS WG PoM: main report*. LU: Publications Office, 2015. [Online]. Available: https://data.europa.eu/doi/10.2779/102151 (visited on 28/05/2024).

[6] M. A. Adegboye, W.-K. Fung and A. Karnik, 'Recent advances in pipeline monitoring and oil leakage detection technologies: Principles and approaches,' *Sensors*, vol. 19, no. 11, p. 2548, Jan. 2019, Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1424-8220. DOI: 10.3390/s19112548. [Online]. Available: https://www.mdpi.com/1424-8220/19/11/2548 (visited on 28/05/2024).

[7] R. Carrera and C. Verde, 'LabVIEW-based SCADA system for sequential leaks' diagnosis in pipelines,'

[8] L. Billmann and R. Isermann, 'Leak detection methods for pipelines,' *Automatica*, vol. 23, no. 3, pp. 381–385, 1st May 1987, Number: 3, ISSN: 0005-1098. DOI: 10.1016/0005-1098(87)90011-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005109887900112 (visited on 22/02/2024).

[9] L. Torres, G. Besançon and C. Verde, 'Leak detection using parameter identification,' *IFAC Proceedings Volumes*, 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, vol. 45, no. 20, pp. 910–915, 1st Jan. 2012, ISSN: 1474-6670. DOI: 10.3182/20120829-3-MX-2028.00070. [Online]. Available: https://

www.sciencedirect.com/science/article/pii/S1474667016348704 (visited on 29/05/2024).

[10] C. Verde, 'Multi-leak detection and isolation in fluid pipelines,' *Control Engineering Practice*, vol. 9, no. 6, pp. 673–682, 1st Jun. 2001, ISSN: 0967-0661. DOI: 10.1016/S0967-0661(01)00026-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066101000260 (visited on 29/05/2024).

[11] O. M. Aamo, J. Salvesen and B. A. Foss, 'OBSERVER DESIGN USING BOUNDARY INJECTIONS FOR PIPELINE MONITORING AND LEAK DETECTION,' *IFAC Proceedings Volumes*, 6th IFAC Symposium on Advanced Control of Chemical Processes, vol. 39, no. 2, pp. 53–58, 1st Jan. 2006, Number: 2, ISSN: 1474-6670. DOI: 10.3182/20060402-4-BR-2902.00053. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S147466701635296X (visited on 22/02/2024).

[12] O. M. Aamo, 'Leak detection, size estimation and localization in pipe flows,' *IEEE Transactions on Automatic Control*, vol. 61, no. 1, pp. 246–251, Jan. 2016, Number: 1 Conference Name: IEEE Transactions on Automatic Control, ISSN: 1558-2523. DOI: 10.1109/TAC.2015.2434031. [Online]. Available: https://ieeexplore.ieee.org/document/7109136 (visited on 26/02/2024).

[13] H. Anfinsen and O. M. Aamo, 'Leak detection, size estimation and localization in branched pipe flows,' *Automatica*, vol. 140, p. 110 213, 1st Jun. 2022, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2022.110213. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109822000589 (visited on 22/02/2024).

[14] N. C. A. Wilhelmsen and O. M. Aamo, 'Leak detection, size estimation and localization in water distribution networks containing loops,' in *2022 IEEE 61st Conference on Decision and Control (CDC)*, ISSN: 2576-2370, Dec. 2022, pp. 5429–5436. DOI: 10.1109/CDC51059.2022.9993208. [Online]. Available: https://ieeexplore.ieee.org/document/9993208 (visited on 22/02/2024).

[15] N. C. A. Wilhelmsen and O. M. Aamo, 'Explicit backstepping kernel solutions for leak detection in branched pipe flows,' *IEEE Control Systems Letters*, vol. 7, pp. 913–918, 2023, Conference Name: IEEE Control Systems Letters, ISSN: 2475-1456. DOI: 10.1109/LCSYS.2022.3228952. [Online]. Available: https://ieeexplore.ieee.org/document/9983797 (visited on 29/05/2024).

[16] N. C. A. Wilhelmsen and O. M. Aamo, 'Explicit backstepping kernel solutions for leak detection in pipe flow networks containing loops,' in *2023 62nd IEEE Conference on Decision and Control (CDC)*, ISSN: 2576-2370, Dec. 2023, pp. 5208–5215. DOI: 10.1109/CDC49753.2023.10383406. [Online]. Available: https://ieeexplore.ieee.org/document/10383406 (visited on 22/02/2024).

[17] R. M. Lesyshen, 'Water transmission line leak detection using extended kalman filtering,' Master thesis, University of Saskatchewan, 2005, 62 pp. [Online]. Available: http://hdl.handle.net/10388/etd-03242005-110841.

[18] S. Oven, 'Leak detection in pipelines by the use of state and parameter estimation,' Master thesis, NTNU, 2014, 62 pp. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/261148.

[19] J. A. Delgado-Aguiñaga, V. Puig and F. I. Becerra-López, 'Leak diagnosis in pipelines based on a kalman filter for linear parameter varying systems,' *Control Engineering Practice*, vol. 115, p. 104 888, 1st Oct. 2021, ISSN: 0967-0661. DOI: 10.1016/j.conengprac.2021.104888. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066121001659 (visited on 29/05/2024).

[20] A. D. Polyanin, W. E. Schiesser and A. I. Zhurov, 'Partial differential equation,' *Scholarpedia*, vol. 3, no. 10, p. 4605, 10th Oct. 2008, ISSN: 1941-6016. DOI: 10.4249/scholarpedia.4605. [Online]. Available: http://www.scholarpedia.org/article/Partial_differential_equation (visited on 09/05/2024).

[21] A. Kværnø, 'Partial differential equations and finite difference methods.,' 11th Apr. 2020. [Online]. Available: https://wiki.math.ntnu.no/_media/tma4130/2020h/pde.pdf (visited on 20/03/2024).

[22] E. W. Weisstein. 'Smooth function.' Publisher: Wolfram Research, Inc. (), [Online]. Available: https://mathworld.wolfram.com/ (visited on 20/03/2024).

[23] C. Verde and L. Torres, Eds., *Modeling and Monitoring of Pipelines and Networks*, vol. 7, Applied Condition Monitoring, Cham: Springer International Publishing, 2017, ISBN: 978-3-319-55944-5. DOI: 10.1007/978-3-319-55944-5. [Online]. Available: http://link.springer.com/10.1007/978-3-319-55944-5 (visited on 28/02/2024).

[24] M. H. Chaudhry, *Applied Hydraulic Transients*. New York, NY: Springer, 2014, ISBN: 978-1-4614-8538-4. DOI: 10.1007/978-1-4614-8538-4. [Online]. Available: https://link.springer.com/10.1007/978-1-4614-8538-4 (visited on 28/02/2024).

[25] J. Rojas, C. Verde and L. Torres, 'Estimation of hydraulic gradient for a transport pipeline,' *Journal of Pressure Vessel Technology*, vol. 143, no. 31801, 7th Oct. 2020, ISSN: 0094-9930. DOI: 10.1115/1.4048322. [Online]. Available: https://doi.org/10.1115/1.4048322 (visited on 10/05/2024).

[26] J. C. Nash, *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*. CRC Press, 1st Jan. 1990, 298 pp., Google-Books-ID: M9hTn3UAheQC, ISBN: 978-0-85274-319-5.

[27] E. Kreyszig, *ADVANCED ENGINEERING MATHEMATICS*, 10th. Columbus, Ohio: John Wiley & Sons, Inc, 2011, 1283 pp., ISBN: 978-0-470-45836-5.

[28] G. Strang, *LINEAR ALGEBRA AND ITS APPLICATIONS*, Third. Massachusetts: Thomson Learning, Inc, 1988, 516 pp., ISBN: 0-15-551005-3.

[29] G. Besançon, *Nonlinear Observers and Applications*. Springer-Verlag Berlin Heidelberg, 2007, 234 pp., ISBN: 978-3-540-73502-1. [Online]. Available: https://link.springer.com/book/10.1007/978-3-540-73503-8 (visited on 09/05/2024).

[30] G. Welch and G. Bishop, 'An introduction to the kalman filter,' 2006.

[31] R. E. Kalman, 'A new approach to linear filtering and prediction problems,' *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1st Mar. 1960, ISSN: 0021-9223. DOI: 10.1115/1.3662552. [Online]. Available: https://doi.org/10.1115/1.3662552 (visited on 09/05/2024).

[32] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control, 2nd Edition*, 2nd ed. Trondheim: John Wiley & Sons, Inc, 2021, ISBN: 978-1-119-57505-4.

[33] K. Reif, F. Sonnemann and R. Unbehauen, 'An EKF-based nonlinear observer with a prescribed degree of stability,' *Autom.*, vol. 34, no. 9, pp. 1119–1123, 1998. DOI: 10.1016/S0005-1098(98)00053-3. [Online]. Available: https://doi.org/10.1016/S0005-1098(98)00053-3.

[34] J. Kleinberg and E. Tardos, *Algorithm Design: Pearson New International Edition*. Pearson Education, 29th Aug. 2013, 828 pp., Google-Books-ID: ayOpBwAAQBAJ, ISBN: 978-1-292-03704-2.

[35] S. G. Williamson and E. A. Bender, *Lists, Decisions and Graphs*. San Diego: University of California, 2010, 261 pp. [Online]. Available: https://books.google.no/books?id=vaXv_yhefG8C.

[36] 'Shortest path between two single nodes - MATLAB shortestpath - MathWorks nordic.' (), [Online]. Available: https://se.mathworks.com/help/matlab/ref/graph.shortestpath.html (visited on 06/05/2024).

[37] 'Breadth-first graph search - MATLAB bfsearch - MathWorks nordic.' (), [Online]. Available: https://se.mathworks.com/help/matlab/ref/graph.bfsearch.html (visited on 06/05/2024).

[38] O. US EPA. 'EPANET.' (24th Jun. 2014), [Online]. Available: https://www.epa.gov/water-research/epanet (visited on 21/03/2024).

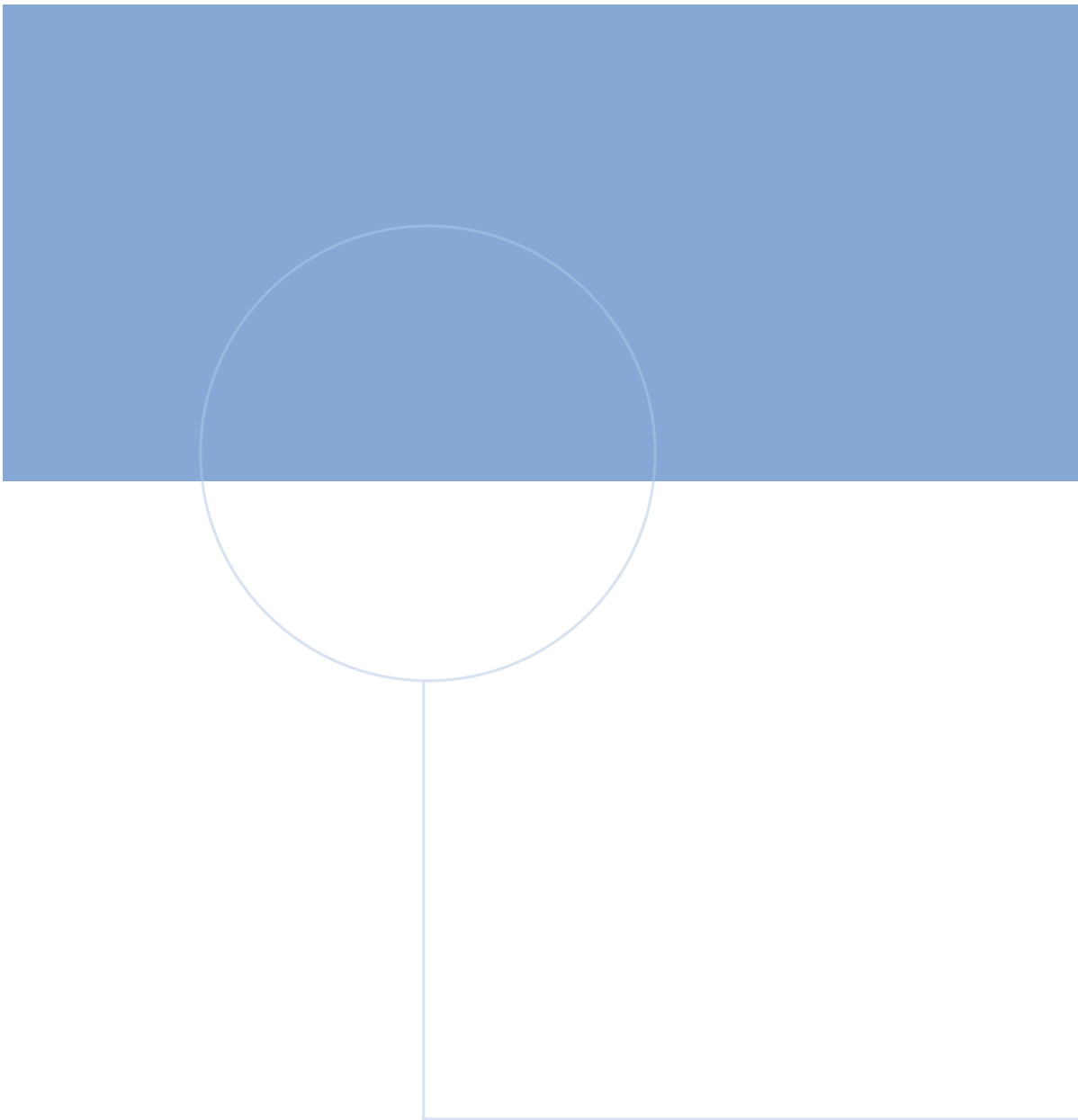
[39] 'Units of measurement — EPANET 2.2 documentation.' (), [Online]. Available: https://epanet22.readthedocs.io/en/latest/back_matter.html#command-line (visited on 07/05/2024).

[40] 'The network model — EPANET 2.2 documentation.' (), [Online]. Available: https://epanet22.readthedocs.io/en/latest/3_network_model.html (visited on 22/05/2024).

[41] 'Analysis algorithms — EPANET 2.2 documentation.' (), [Online]. Available: https://epanet22.readthedocs.io/en/latest/12_analysis_algorithms.html#analysis-algorithms (visited on 24/05/2024).

[42] 'Solve nonstiff differential equations — medium order method - MATLAB ode45 - MathWorks nordic.' (), [Online]. Available: https://se.mathworks.com/help/matlab/ref/ode45.html (visited on 02/06/2024).

[43] 'Norgeskart.' (), [Online]. Available: https://www.norgeskart.no/#!?project=norgeskart&layers=1005&zoom=10&lat=7040734.16&lon=246450.35 (visited on 03/06/2024).



NTNU

Norwegian University of
Science and Technology