

System Identification: theory and applications

Songlei Fang

Technische Universiteit Delft



SYSTEM IDENTIFICATION: THEORY AND APPLICATIONS

by

Songlei FANG

to obtain the degree of Master of Science
in Electrical Engineering
Track Wireless Communication and Sensing
at the Delft University of Technology,
to be defended publicly on Monday August 29, 2022

Student number: 5233038
Project duration: November, 2021 – , August, 2022
Thesis committee: Prof.dr.ir. Rob Kooij, TU Delft, supervisor
Ivan Jokić, TU Delft, daily supervisor
Dr. J.L.A. Dubbeldam, TU Delft, committee



PREFACE

With this thesis project, 'Networked System Identification: theory and applications', I finished the Master of Science degree in Electrical Engineering at the Delft University of Technology. The project has been carried out at the Network Architectures and Services (NAS) group. First of all I would like to thank the Network Architectures and Services (NAS) group for giving me this opportunity to do this project. I would like to express gratitude to my supervisor Professor Rob Kooij for guidance and encouragement throughout the entire duration of this project. I would like to thank Ivan Jokić, my daily supervisor, for his support and suggestions that helped me a lot. I would like to thank Dr. J.L.A. Dubbeldam for being a part of my thesis committee. I am very happy to have completed this interesting research in the last nine months. In addition, I would also like to thank my friends for their company and encouragement during these nine months.

*Songlei Fang
Delft, 2022*

ABSTRACT

A complex network consists of the underlying topology, defined by a graph and the dynamical processes taking place on a network, defined by a set of governing equations. In this thesis, we deploy the discrete-time linear state-space (DLSS) model to identify the dynamical processes taking place on a complex network. Unlike the black-box identification approach, we split the network into dynamical units and identify the dynamics of each dynamical unit independently. Next, we relate input/output vectors of individual dynamical units, based on the underlying topology and provide the model for the dynamics of the entire network. Because we use a linear model, by scaling the model to the entire network, no information is lost about dynamical processes of individual units. In this thesis, we apply this new networked system identification solution to two real-world complex networks, water and road networks, and find this identification approach to successfully improve the identification performance compared to the common black-box identification approach.

CONTENTS

1	Introduction	1
1.1	Objectives	2
1.2	Challenges	2
1.3	Thesis outline	2
2	System and System Identification Theory	5
2.1	Basics of Systems Theory	5
2.1.1	Dynamical systems classification	5
2.1.2	Discrete-time linear state-space model	7
2.1.3	Stability of discrete-time linear systems	7
2.2	System identification	9
2.2.1	Parametric Methods	9
2.2.2	Subspace Methods for LTI Systems	10
3	Networked Linear System	15
3.1	Complex network	15
3.1.1	Network topology	16
3.1.2	Process on the network	16
3.1.3	Network dynamics	18
3.2	Identifying dynamics of a networked system	20
3.2.1	Analytical approach	20
3.2.2	Simple toy example	21
4	Applications and Results	25
4.1	Water networks	25
4.1.1	Basic information about the network	25
4.1.2	Identification approaches and results	27
4.2	Road networks	31
4.2.1	Basic information about the network	31
4.2.2	Identification approaches and results	32
5	Conclusions and Future Research	43
5.1	Conclusions	43
5.2	Future research	44
	Bibliography	45
A	Appendix A	47
B	Appendix B	51
C	Appendix C	53

1

INTRODUCTION

Networks may be found everywhere. Electric power networks, transportation networks, water networks, economic networks, the Internet, the World Wide Web, social networks, and biological networks are some examples of networks. A network is defined by its underlying topology as well as the dynamical processes taking place on the network [1, 2]. The network topology has been deeply investigated in the past two decades [3].

Dynamical processes on complex networks such as synchronization [4], diffusion [5], epidemic spreading [6] and traffic [7] have been intensively researched in the past two decades. Recently, scientific study has focused on the interaction between network topology and dynamics [8]. While many actual networks have comparable (universal) structural qualities, Barzel, Harush, and colleagues [9] demonstrated that there are classes of dynamical processes that display fundamentally diverse flow patterns. The network dynamics are determined by the structure of the network as well as the type of dynamic interactions that exist between the nodes.

Recent research has focused on complex networks with linear dynamics [10], which can be justified in a variety of ways. Firstly, non-linear dynamics on networks may be approximated or bounded by linear dynamics in most circumstances, which can be seen in a paper about epidemic models [6]. Secondly, the concept of controlling complex networks has been intensively researched [11]. Non-linear system control is a complex topic in system theory, and it is based on the well-known linear system control theory [12]. In network control theory, there is a similar sequence of research progress. In the literature, networks of agents (dynamical systems) [13], networked multi-input-multi-output (MIMO) systems [14], and complex networked dynamical systems [10] have all been used to describe complex networks with linear dynamics. From the perspective of system/control theory, the above techniques create network models with linear processes.

A new concept about the linear processes on complex networks has been proposed by Jokić and Van Mieghem in 2020 [15]. In this new approach, we split a complex network into dynamical units and identify their dynamics independently. In the next

step, we relate their input/output vectors based on the underlying topology and provide the model for the dynamics of the entire network.

In this thesis we apply this new approach to identify the dynamics of two real-world networks, namely water networks and road networks, and we provide the estimation accuracy analysis. We find that applying this new identification approach to real-world networks can successfully improve the identification performance of the common black-box identification approach. In other words, this thesis introduces and validates a new identification method that can be successfully applied to analyse the dynamics of real-world networks, and by analysing the estimation accuracy we find that it is more accurate than the common black-box identification approach.

1.1. OBJECTIVES

Based on the background mentioned above, the objectives of our research are as follows:

1. Select the appropriate model to analyse the dynamics.
2. Analyse the dynamics of the entire network by taking into account the dynamics of individual systems and the underlying network topology.
3. Apply the theory to analyse the dynamics of two real-world networks (water networks and road networks)
4. Summarize the identification results of two real-world networked systems and analyse the estimation accuracy of the dynamics

1.2. CHALLENGES

There are several challenges when we focus on this research:

1. Identify the model parameters with the best descriptive/predictive power given input and output measurements from a system.
2. Relate dynamic interactions between different individual systems, based on the underlying topology
3. Given the linear model of a networked system, how to steer the dynamics on a network towards some “desired” regime/state.
4. Select suitable real-world systems and explore the benefits of applying our identification approach.

1.3. THESIS OUTLINE

The structure of this thesis is as follows; Chapter 2 presents an overview of the system theory and system identification. In addition, we introduce the discrete-time linear state-space model used to represent the dynamics of a system and several methods for system identification. In Chapter 3, we introduce networked linear systems and explain

the analytical approach for identifying the network dynamics by combining dynamics of individual systems and taking into account the underlying topology. In Chapter 4, we apply the identification approach on two real-world networks and analyse the estimation accuracy. Finally, we present our conclusions and discuss a possible scope for future research in Chapter 5.

2

SYSTEM AND SYSTEM IDENTIFICATION THEORY

In this chapter we present an overview of system theory. We introduce different types of dynamical systems and their governing equations, which can help a reader to understand the differences between them. Firstly, we introduce the general nonlinear system and gradually introduce assumptions that allow us to define the discrete-time linear state-space model, that is used to identify the dynamics of an individual system. Further we introduce several methods for system identification.

2.1. BASICS OF SYSTEMS THEORY

In mathematics, a dynamical system is a system in which a function describes the time dependence of a point in an ambient space. The swing of a pendulum, the flow of water in a pipe, or the annual spring fish population in a lake are some examples of dynamical systems.

2.1.1. DYNAMICAL SYSTEMS CLASSIFICATION

In real life, most systems perform nonlinear dynamics in time. Dynamical systems can be divided into a number of categories based on how their dynamics depend on time. A discrete dynamical system is a system whose state is only defined at a series of discrete points in time. A continuous dynamical system is a system that defined at any moment in time. Because most real world systems perform nonlinearly in time, we first analyze general nonlinear systems.

General Nonlinear System The most general form of a dynamical system is determined by the following governing equations

$$\begin{aligned}\frac{dx(t)}{dt} &= f(x(t), u(t), v(t)) \\ y(t) &= h(x(t), u(t), w(t))\end{aligned}\tag{2.1}$$

where the $n \times 1$ vector $x(t) \in \mathbb{R}^n$ denotes the state-space variables, the $p \times 1$ vector $u(t) \in \mathbb{R}^p$ indicates the input vector, the $p \times 1$ vector $y(t) \in \mathbb{R}^m$ defines the output vector, while t denotes time. The vectors $v(t)$ and $w(t)$ indicate (possibly multidimensional) noise. State variables contain the dynamics of the system.

2

Discrete Time Systems The classifications of dynamical systems described in this section still apply if the time is discrete rather than continuous,

$$\frac{dx_i(t)}{dt} = \lim_{\Delta t \rightarrow 0} \frac{x_i(t+\Delta t) - x_i(t)}{\Delta t} \rightarrow \left. \frac{x_i(t+\Delta t) - x_i(t)}{\Delta t} \right|_{\Delta t=1} \stackrel{\text{def}}{=} x_i[k+1] - x_i[k]$$

by explicitly replacing $t \in \mathbb{R}$ by $k \in \mathbb{N}$ and $\frac{dx(t)}{dt}$ by $x[k+1]$. A general discrete nonlinear dynamical system is given by

$$\begin{aligned} x[k+1] &= f(x[k], u[k], v[k]) \\ y[k] &= h(x[k], u[k], w[k]) \end{aligned} \quad (2.2)$$

Above we have described the different ways in which the General Nonlinear System can be presented. It will have different governing equations (Eq.(2.1) and Eq.(2.2)) depending on whether the time is continuous or not. Therefore we can choose the corresponding governing equations when focusing on the dynamics of the real-world networks.

We mentioned that non-linear dynamics on networks may be approximated or bounded by linear dynamics in most circumstances in the introduction part. So next we focus on linear dynamical systems.

Linear Parameter Varying (LPV) System The governing equation for a linear state-space system with parameter fluctuations is

$$\begin{aligned} \frac{dx(t)}{dt} &= A(t)x(t) + B(t)u(t) + v(t) \\ y(k) &= C(t)x(t) + D(t)u(t) + w(t) \end{aligned} \quad (2.3)$$

Linear Time-Invariant (LTI) System The system dynamics of a linear time-invariant [state-space] system are given by

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) + v(t) \\ y(k) &= Cx(t) + Du(t) + w(t) \end{aligned} \quad (2.4)$$

where the $n \times n$ state matrix A defines how the $n \times 1$ state vector $x(t)$ depends on its previous value, while the $n \times m$ input matrix B determines the relation between the state vector $x(t)$ and the previous value of the $m \times 1$ input vector $u(t)$. The relation between the $p \times 1$ output vector $y(t)$ and the state vector $x(t)$ is defined by the $p \times n$ output matrix C . Finally, direct relation between the output vector $y(t)$ and the input vector $u(t)$ is defined by the $p \times m$ feedforward matrix D .



Figure 2.1: Discrete-time linear state space (DLSS) model

If only the input-output relation of the system is of interest, the Eq(2.3) can be transformed to the innovation form.

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) + Ke(t) \\ y(k) &= Cx(t) + Du(t) + e(t) \end{aligned} \quad (2.5)$$

$e(t)$ is a white noise sequence and K is the Kalman gain [16].

In order to identify the dynamics of real-world networks, we need to understand the different ways in which dynamics are represented. This allows us to choose the appropriate governing equations to present the dynamics. This is the reason why we first introduced the classification of dynamical systems.

2.1.2. DISCRETE-TIME LINEAR STATE-SPACE MODEL

Discrete-time linear state space (DLSS) model can be used to define the dynamics of linear time-invariant (LTI) systems. In addition, a non-linear system can be approximated by a discrete-time linear state space (DLSS) model with sufficient number of state variables. That is the reason why we use Discrete-time linear state space (DLSS) model to represent the dynamics of the systems in our research.

The dynamics within the i -th node/system obey the DLSS governing equations:

$$\begin{aligned} x_i[k+1] &= A_i x_i[k] + B_i u_i[k] \\ y_i[k] &= C_i x_i[k] + D_i u_i[k] \end{aligned} \quad (2.6)$$

where the discrete time is modelled by k . The $n_i \times n_i$ state matrix A_i defines how the $n_i \times 1$ state vector x_i depends on its previous value, while the $n_i \times m_i$ input matrix B_i determines the relation between the state vector x_i and the previous value of the $m_i \times 1$ input vector u_i . The relation between the $p_i \times 1$ output vector y_i and the state vector x_i is defined by the $p_i \times n_i$ output matrix C_i . Finally, direct relation between the output vector y_i and the input vector u_i is defined by the $p_i \times m_i$ feedforward matrix D_i .

As shown in the Figure 2.1, a simple block diagram of a DLSS model is provided, defining the relation between the input and output vector.

2.1.3. STABILITY OF DISCRETE-TIME LINEAR SYSTEMS

The $n \times n$ state space matrix A defines how the $n \times 1$ state space variables $x[k]$ depend on their values in the previous time instant $x[k-1]$. The eigenvalue decomposition of

the $n \times n$ state space matrix A is defined as follows

$$A = M \cdot \Lambda \cdot M^T,$$

where the $n \times n$ eigenvector matrix M contains the n eigenvectors of A in its columns, where the i -th column of M equals the $n \times 1$ eigenvector m_i , while the $n \times n$ diagonal matrix $\Lambda = \text{diag}(\lambda)$ contains the eigenvalues $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ of A . The eigenvalue relation can be modified further as follows:

$$\lambda_i m_i = A m_i \quad (2.7)$$

The corresponding column vectors m are defined as eigenvectors. Then Eq.(2.7) can be rewritten as

$$(\lambda_i I - A) m_i = 0 \quad (2.8)$$

The condition of such a set of linear equations has a non-trivial solution is that

$$\det(\lambda_i I - A) = 0 \quad (2.9)$$

Then we can rewrite the Eq.(2.8) as

$$\lambda^n + a_{n-1} \lambda^{n-1} + \dots + a_1 \lambda + a_0 = 0 \quad (2.10)$$

we can also factor it according to its roots $\lambda_1, \dots, \lambda_n$

$$(\lambda - \lambda_1)(\lambda - \lambda_2) \dots (\lambda - \lambda_n) = 0 \quad (2.11)$$

We can define

$$M = [m_1 \ m_2 \ \dots \ m_n] \quad (2.12)$$

and

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & \lambda_n \end{bmatrix} \quad (2.13)$$

Then

$$\Lambda^k = \begin{bmatrix} \lambda_1^k & 0 & \dots & 0 \\ 0 & \lambda_2^k & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & \lambda_n^k \end{bmatrix} \quad (2.14)$$

and

$$\Phi(k) = A^k = (M \Lambda M^{-1})^k = M \Lambda^k M^{-1} \quad (2.15)$$

The following is a summary of the prerequisites for system stability:

All eigenvalues must be smaller than one for a linear discrete-time system described by the state equation to be asymptotically stable.

There are some additional independent conditions that should be taken into account:

1. When one or more eigenvalues, or a pair of conjugate eigenvalues, have magnitudes greater than one, the notion of stability is broken since at least one related modal component increases exponentially without bound from any initial state.
2. If an eigenvalue $\lambda = 1$, then it produces a model exponent $\lambda^k = 1^k = 1$ that is a constant. Again, the system is classified as being minimally stable because the response to the system neither grows nor decays.

2.2. SYSTEM IDENTIFICATION

In this section, an overview of system identification approaches is provided, which can help readers understand the meaning of system identification and how to identify the dynamics of systems.

System identification aims at determining dynamical systems equations which produce a similar output given the same input as the original system. We can broadly divide the methods into two categories: parametric methods and subspace methods. For parametric methods, we simply give definitions to help a reader to understand some simple methods of system identification. We focus mainly on subspace methods because we use this method in the application of the water network. A more detailed description will be given in a subsequent section.

2.2.1. PARAMETRIC METHODS

The governing equations of a nonlinear discrete-time system are parametrized by the parameter vector θ .

$$\begin{aligned} x[k+1] &= f(x[k], u[k], v[k], \theta) \\ y[k] &= h(x[k], u[k], w[k], \theta) \end{aligned} \quad (2.16)$$

The unknowns of a system are captured by the parameter vector θ , as explained in [17]. Because $v[k]$, $w[k]$, $x[k]$ and $y[k]$ are all random processes, for given θ , the probability density of $x[k]$ and $y[k]$ are determined as $f(x[k]|\theta)$ and $f(y[k]|\theta)$, respectively. The $k \times 1$ vectors $U[k] = (u[1], \dots, u[k])$ and $Y[k] = (y[1], \dots, y[k])$ contain the input and output values, until time moment k .

Various estimating techniques, including those listed in the following, can be used to obtain an estimate $\hat{\theta}$ of the parameter vector (see e.g. [18] for a more detailed discussion).

Maximum Likelihood Estimation Finding θ in such a way as to maximize the likelihood of the output $Y[n]$ given the input $U[n]$ is a frequent strategy if previous knowledge of the distribution of θ is unavailable.

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}} f(Y[n]|U[n], \theta) \quad (2.17)$$

Maximum-A-Posteriori Estimation Maximizing the likelihood of θ given the input and output is a simple strategy if, on the other hand, a priori knowledge of the distribution of θ is known.

$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}} f(\theta|Y[n], U[n]) = \underset{\theta}{\operatorname{argmax}} f(Y[n]|U[n], \theta)f(\theta) \quad (2.18)$$

In comparison to maximum likelihood, the maximum-a-posteriori strategy is more inclusive. In fact, only when one sets $f(\theta) = \text{const}$, it holds that $\hat{\theta}_{MAP} = \hat{\theta}_{ML}$.

Bayesian Estimation In contrast to the first two methods, a cost criteria rather than a likelihood term is used to assess the estimator's goodness. When the true parameters are stored in θ , the cost function $C(\hat{\theta}, \theta)$ determines how "expensive" or "poor" the estimation output $\hat{\theta}$ is. After that, the Bayesian estimator minimizes the anticipated cost:

$$\hat{\theta}_{BAYES} = \underset{\hat{\theta}}{\operatorname{argmin}} E_{\theta}[C(\hat{\theta}, \theta)|U[n], Y[n]] \quad (2.19)$$

A commonly used cost criterion is the Mean Square Error (MSE)

$$C_{MSE}(\hat{\theta}, \theta) = \|\hat{\theta} - \theta\|^2 \quad (2.20)$$

which leads in the Minimum Mean Square Error (MMSE) estimate

$$\hat{\theta}_{MMSE} = \underset{\hat{\theta}}{\operatorname{argmin}} E_{\theta}[C_{MSE}(\hat{\theta}, \theta)|U[n], Y[n]] \quad (2.21)$$

It can be shown when weak regularity assumptions are used [19], that

$$\hat{\theta}_{MMSE} = E_{\theta}[\theta|Y[n], U[n]] \quad (2.22)$$

$\hat{\theta}_{MMSE} = \hat{\theta}_{MAP}$ if $p(\theta|U[n], Y[n])$ is unimodal and symmetric (such as Gaussian). For Bayesian estimation, $C = C_{MSE}$ is typically assumed in the absence of a specifically specified cost function.

2.2.2. SUBSPACE METHODS FOR LTI SYSTEMS

Subspace methods can be classified as "semi-parametric" methods, because only an upper bound for the system order i in *LTI* systems needs to be chosen, but the system matrices do not need to be parametrised by a vector θ as in the previous sections.

Basic Idea

We take into account a linear, discrete-time, multivariable, time-invariant system with m inputs and l outputs, obeying the following governing equations:

$$\begin{aligned} x[k+1]_{n \times 1} &= A_{n \times n}x[k]_{n \times 1} + B_{n \times m}u[k]_{m \times 1} \\ y[k]_{l \times 1} &= C_{l \times n}x[k]_{n \times 1} + D_{l \times m}u[k]_{m \times 1} \end{aligned} \quad (2.23)$$

Furthermore, we will also frequently use the set of Markov parameters H_i of the linear system, defined by $H_0 = D$ $H_i = C.A^{i-1}.B$ ($i > 0$) The idea is to stack shifted version of y , u and x into Hankel matrices. The following equation for the state space system can be easily obtained by manipulating the description of the system in state space [20]:

$$Y_h(k, i, j) = \Gamma(i)X(k, j) + H_{tl}(i)U_h(k, i, j) \quad (2.24)$$

The subscript h indicates that the matrix has a block Hankel structure and the matrix is structured as follows:

$$Y_h(k, i, j) = \begin{bmatrix} y[k] & y[k+1] & \cdots & y[k+j-1] \\ y[k+1] & y[k+2] & \cdots & y[k+j] \\ y[k+2] & y[k+3] & \cdots & y[k+j+1] \\ \cdots & \cdots & \cdots & \cdots \\ y[k+i-1] & y[k+i] & \cdots & y[k+j+i-2] \end{bmatrix} \quad (2.25)$$

This matrix is a $li \times j$ block Hankel matrix created from $i+j-1$ output vectors. The $li \times j$ matrix $U_h(k, i, j)$ has the same structure with input vectors. The $li \times mi$ lower triangular block Toeplitz matrix $H_{tl}(i)$ consists of the i first markov parameters:

$$H_{tl}(i) = \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ CA^{i-2}B & CA^{i-3}B & CA^{i-4}B & \cdots & D \end{bmatrix} \quad (2.26)$$

while the $il \times n$ observability matrix $\Gamma(i)$ is defined as follow:

$$\Gamma(i) = \begin{bmatrix} C \\ CA \\ CA^2 \\ \cdots \\ CA^{i-1} \end{bmatrix} \quad (2.27)$$

The singular value decomposition (SVD) is the primary tool in the explicit numerical solution of this identification scheme, even though the geometrical visualization is crucial from a conceptual standpoint.

Every real $m \times n$ matrix A can be decomposed in three real matrices according to the Autonne-Ectart-Young theorem [21]:

$$A_{m \times n} = U_{m \times m} \Sigma_{m \times n} V_{n \times n}^t \quad (2.28)$$

Matrices U and V are orthogonal, while Σ is real, pseudo-diagonal with non-negative diagonal elements:

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \quad (2.29)$$

with $\Sigma_1 = \text{diag}(\sigma_i)$, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, and r is the algebraic rank of the matrix A . The elements σ_i are the singular values of the matrix A , the columns of U (V) are

the left (right) singular vectors, respectively. They generate an orthonormal basis for the columnspace (rowspace) of the matrix A .

The most accurate method for estimating the rank of a matrix is the singular value decomposition. It is extremely resilient to disturbances, and the majority of software packages nowadays include numerically dependable algorithms and effective software.

2

Identification Algorithm

If U is a $m \times n$ matrix, by leaving out the last(first) blockrow, we can create matrix $\underline{U}(U)$ from U with fewer rows. The context will make clear how many rows specifically need to be skipped. The projection of the row space of Y_h onto the orthogonal complement of the row space of U_h is generally a space that possesses the shift property of the observability matrix, as can be seen from the input-output matrix equation. Starting with the input-output matrix Eq.(2.23), this observation is used in the construction of the matrices A and C :

1. If i is an overestimation of i_0 , the observability index of the system to be identified, and $(j - mi) > n$, then choose the number of columns j of Y_h and U_h bigger than $\max(mi, li)$.
2. Denoted by U_h^\perp the orthonormal matrix whose columnspace is the orthogonal rowcomplement of U_h

$$Y_h U_h^\perp (U_h^\perp)^t = \Gamma(i) X(k, j) U_h^\perp (U_h^\perp)^t \quad (2.30)$$

The rank of $Y_h U_h^\perp (U_h^\perp)^t$ can be computed from the singular value decomposition:

$$Y_h U_h^\perp (U_h^\perp)^t = \begin{bmatrix} P_1 & P_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Q_1^t \\ Q_2^t \end{bmatrix} \quad (2.31)$$

where P_1 is $li \times n$, P_2 is $li \times (li - n)$, $S_1 n \times n$, $Q_1 j \times n$ and $Q_2 j \times (j - n)$. Then we can compute matrices A and C of the model from the SVD equation. For A_t and C_t , they satisfy $\underline{P}_1 A_t = \overline{P}_1$ and C_t = first $l \times n$ blockrow of P_1 .

We define the pseudo-inverse U^+ of U_h , it can be computed from the second input-output relation that

$$P_2^t Y_h U^+ = P_2^t H_{tl} \quad (2.32)$$

Then define $K = P_2^t H_{tl}$, and K can be computed from i blocks of dimension $(li - n) \times m$:

$$\begin{bmatrix} K_1 & \cdots & K_i \end{bmatrix} = \begin{bmatrix} P_1 & \cdots & P_i \end{bmatrix} \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CA^2 B & CAB & D & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ CA^{i-2} B & CA^{i-3} B & CA^{i-4} B & \cdots & D \end{bmatrix} \quad (2.33)$$

where P_k are the $(li - n) \times (li)$ subblocks of P_2^t . Then we can rewrite the equation :

$$\begin{bmatrix} K_1 \\ K_2 \\ K_3 \\ \vdots \\ K_i \end{bmatrix} = \begin{bmatrix} P_1 & P_2 & P_3 & \cdots & P_i \\ P_2 & P_3 & P_4 & \cdots & 0 \\ P_3 & P_4 & P_5 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ P_i & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \Gamma(i-1) \end{bmatrix} \begin{bmatrix} D \\ B \end{bmatrix} \quad (2.34)$$

Finally, we can compute the $n \times m$ matrix B and $l \times m$ matrix D . Thus, we can identify all matrices in state space representation, which means we successfully identify the system. The code for this algorithm is provided in [appendix A](#).

3

NETWORKED LINEAR SYSTEM

3.1. COMPLEX NETWORK

A complex network consists of the underlying topology, defined by the graph, and the process taking place on the network, defined by a set of governing equations. From Table 3.1, we can see the topology and dynamics of some simple real- world networks.


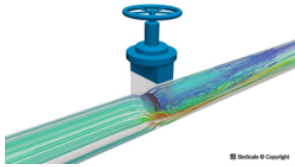


Network	Topology	Dynamic process
Water network		
Road network		

Table 3.1: Examples of complex networks

In this chapter, we mainly focus on the theory about network topology and the process taking place on the network. In addition, we introduce a new approach unlike the black-box identification approach and present how it can help with identifying the dynamics of a networked system.

3.1.1. NETWORK TOPOLOGY

The underlying structure (topology) of a network is assumed to be time-invariant and is represented by a graph $G(N, L)$, which means the graph G have N nodes and L links. The $N \times N$ matrix W defines how nodes are connected in the graph G . If $w_{ij} = 1$, there exists a link between node i and node j , otherwise $w_{ij} = 0$.

There will be some external nodes connected to the graph. We divide these nodes into two categories : input nodes and output nodes. The input nodes point to the nodes of graph G , while the output nodes are pointed out from the nodes of graph G .

There are r input nodes, and we use the $r \times N$ matrix Φ to define how input nodes are connected to the nodes in graph G

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1N} \\ \phi_{21} & \phi_{22} & \dots & \phi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{r1} & \phi_{r2} & \dots & \phi_{rN} \end{bmatrix} \quad (3.1)$$

If $\phi_{ij} = 1$, there exists an input link between input node i and internal node j , otherwise $\phi_{ij} = 0$.

There are q output nodes, and we use the $N \times q$ matrix Ψ to show how they connect to the graph G through output links.

$$\Psi = \begin{bmatrix} \psi_{11} & \psi_{12} & \dots & \psi_{1q} \\ \psi_{21} & \psi_{22} & \dots & \psi_{2q} \\ \vdots & \vdots & \vdots & \vdots \\ \psi_{N1} & \psi_{N2} & \dots & \psi_{Nq} \end{bmatrix} \quad (3.2)$$

If $\psi_{ij} = 1$, there exists an output link between internal node i and output node j , otherwise $\psi_{ij} = 0$.

There may also exist external links between input nodes and output nodes. And we use the $r \times q$ matrix Z to represent this kind of links.

$$Z = \begin{bmatrix} z_{11} & z_{12} & \dots & z_{1q} \\ z_{21} & z_{22} & \dots & z_{2q} \\ \vdots & \vdots & \vdots & \vdots \\ z_{r1} & z_{r2} & \dots & z_{rq} \end{bmatrix} \quad (3.3)$$

If $z_{ij} = 1$, there exists an external link between input node i and output node j , otherwise $z_{ij} = 0$.

Figure 3.1 illustrates a network graph G of 10 nodes, with additional $r = 5$ and $q = 4$.

3.1.2. PROCESS ON THE NETWORK

Every node in the network represents a linear time-invariant (LTI) system (defined in Eq.(2.26)), whose dynamics is captured by the discrete-time linear state space (DLSS) model [22]. The model and the meaning of each variable are explained in section

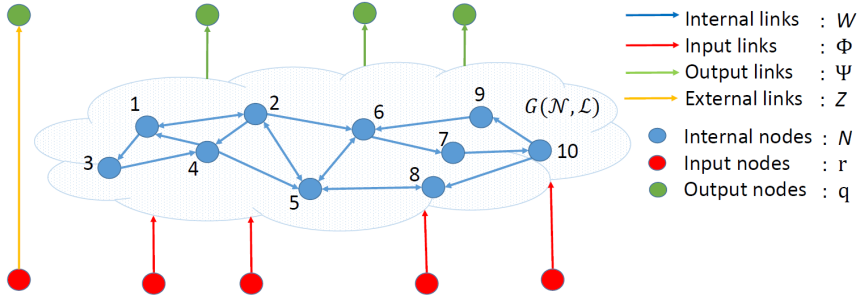


Figure 3.1: a network of 10 nodes [15]

2.1.2. The dynamics within the i -th node/system is determined by DLSS governing equations:

$$\begin{cases} x_i[k+1] = A_i \cdot x_i[k] + B_i \cdot u_i[k] \\ y_i[k] = C_i \cdot x_i[k] + D_i \cdot u_i[k] \end{cases} \quad (3.4)$$

For example, Figure 3.2 shows the topology (lower part) and the dynamics (block diagram in the upper part) of an interconnected networked system, composed of three nodes/systems.

We define the $N \times 1$ vector n to represent the number of states for each node/system of the network:

$$n = [n_1 \quad n_2 \quad \dots \quad n_i \quad \dots \quad n_N]^T \quad (3.5)$$

We also define the $N \times 1$ vector m to represent the dimension of the input vector u_i for each system.

$$m = [m_1 \quad m_2 \quad \dots \quad m_i \quad \dots \quad m_N]^T \quad (3.6)$$

Similarly, the $N \times 1$ vector p represent the dimension of the output vector y_i for each system.

$$p = [p_1 \quad p_2 \quad \dots \quad p_i \quad \dots \quad p_N]^T \quad (3.7)$$

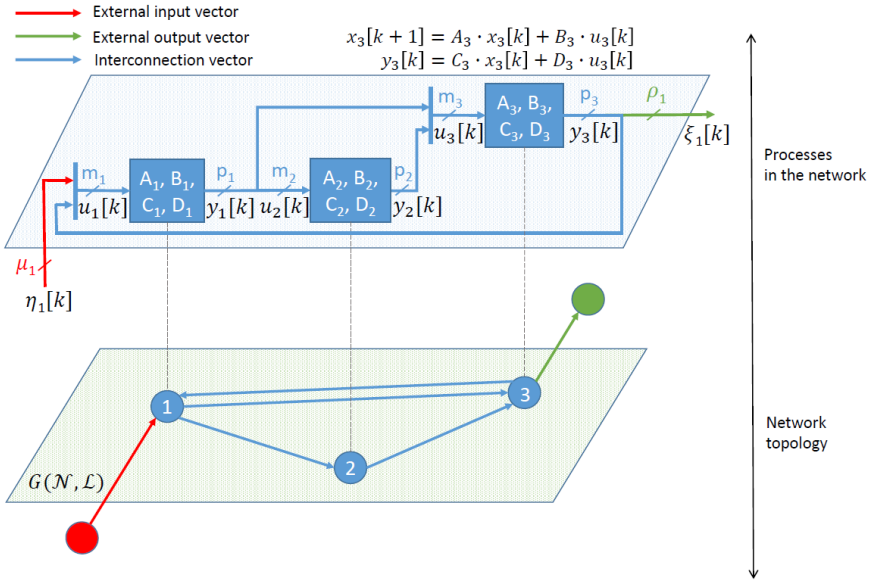
We also use the $r \times 1$ vector μ defined in Eq.(3.8) to present the dimension of each external input vector, while the $M \times 1$ vector η defined in equation Eq.(3.9) concatenating r external input vectors, where

$$M = \sum_{j=1}^r \mu_j$$

$$\mu = [\mu_1 \quad \mu_2 \quad \dots \quad \mu_i \quad \dots \quad \mu_r]^T \quad (3.8)$$

$$\eta = [\eta_1 \quad \eta_2 \quad \dots \quad \eta_i \quad \dots \quad \eta_r]^T \quad (3.9)$$

Similarly, we use the $q \times 1$ vector ρ defined in Eq.(3.10) to store the dimension of each external output vector ξ_i , while the $P \times 1$ vector ξ defined in Eq.(3.11) concatenates q

Figure 3.2: DLSS dynamics of a simple network with $N = 3$ nodes/systems [15]

external input vectors, where

$$P = \sum_{j=1}^q \rho_j$$

$$\rho = [\rho_1 \quad \rho_2 \quad \dots \quad \rho_i \quad \dots \quad \rho_q]^T \quad (3.10)$$

$$\xi = [\xi_1 \quad \xi_2 \quad \dots \quad \xi_i \quad \dots \quad \xi_q]^T \quad (3.11)$$

The dimension of input vectors stored in the $N \times 1$ vector m and dimensions of the external output vectors stored in the $q \times 1$ vector ρ must obey [15]

$$\begin{bmatrix} m_{N \times 1} \\ \rho_{q \times 1} \end{bmatrix} = \begin{bmatrix} W_{N \times N}^T & \Phi_{N \times r}^T \\ \Psi_{q \times N}^T & Z_{q \times r}^T \end{bmatrix} \cdot \begin{bmatrix} p_{N \times 1} \\ \mu_{r \times 1} \end{bmatrix} \quad (3.12)$$

because the input vector u_i of the i -th system consists of the output vectors of those systems connected to the system i and the external input vectors connected to the system, stored in the $N \times N$ adjacency matrix W and the $N \times r$ matrix Φ . An equivalent reasoning holds for the dimension of each external input vector, stored in the $q \times 1$ vector ρ .

3.1.3. NETWORK DYNAMICS

A complex system network $G(N, L)$ is shown in Figure 3.3. The internal dynamics of each individual system is captured by the DLSS governing equations provided in Eq.(3.4). We construct a DLSS model that captures the dynamics between the $P \times 1$ external output vector ξ shown in Eq.(3.11) and the $q \times 1$ external input vector ρ defined in Eq.(3.10),

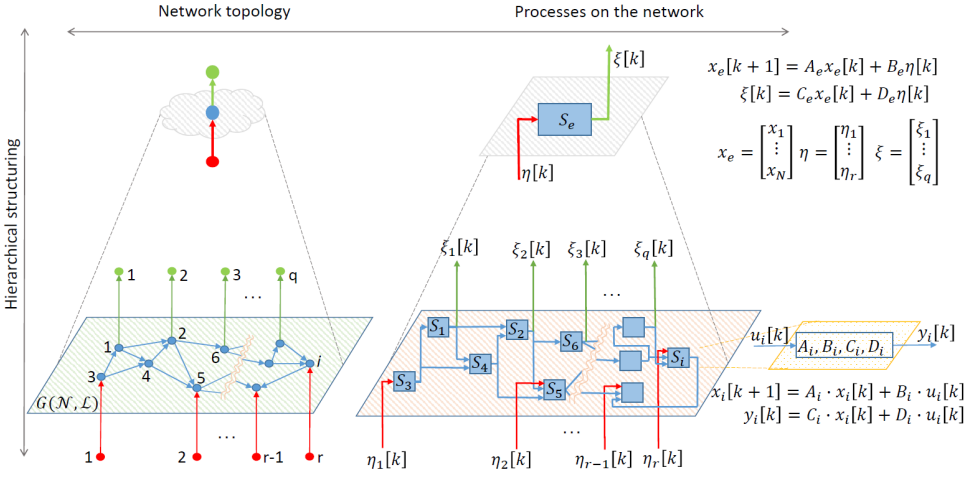


Figure 3.3: Underlying topology vs linear processes on the complex network [15]

without loosing any information from individual system level, as is shown in Eq.(3.15). The dynamics of the entire network obey the following governing equations

$$\begin{cases} x_e[k+1] = A_e \cdot x_e[k] + B_e \cdot \eta[k] \\ \xi[k] = C_e \cdot x_e[k] + D_e \cdot \eta[k] \end{cases} \quad (3.13)$$

where the $\sum_{j=1}^N n_j \times 1 x_e$ state vector x_e stores state variables of each individual system in the network. The vector $\sum_{j=1}^N n_j \times 1 x_e$ represent the states of each system in the network:

$$x_e[k] = \begin{bmatrix} x_1[k] \\ x_2[k] \\ \vdots \\ x_N[k] \end{bmatrix} \quad (3.14)$$

We obtain the matrices A_e , B_e , C_e and D_e according to the network topology and the dynamics of individual nodes/systems. Here we only show the final results and the details can be found in Chapter 4 in [15].

$$\begin{cases} A_e = (B_d \cdot F_w) \cdot (I - D_d \cdot F_w)^{-1} \cdot C_d + A_d \\ B_e = (B_d \cdot F_w) \cdot (I - D_d \cdot F_w)^{-1} \cdot (D_d \cdot F_\phi) + B_d \cdot F_\phi \\ C_e = F_\psi \cdot (I - D_d \cdot F_w)^{-1} \cdot C_d \\ D_e = F_\psi \cdot (I - D_d \cdot F_w)^{-1} \cdot (D_d \cdot F_\phi) + F_z \end{cases} \quad (3.15)$$

When there is no direct interaction between the input vector u_i and the output vector y_i of each system in the network, the results are:

$$\begin{cases} A_e = B_d \cdot F_w \cdot C_d + A_d \\ B_e = B_d \cdot F_\phi \\ C_e = F_\psi \cdot C_d \\ D_e = F_z \end{cases} \quad (3.16)$$

The meaning of variables can be seen in Table B.1 and Table B.2.

3

3.2. IDENTIFYING DYNAMICS OF A NETWORKED SYSTEM

In this part, we introduce how to identify the dynamics of a networked system using the model introduced in the previous subsection.

In addition, we present a simple example to help the reader to understand this new approach.

3.2.1. ANALYTICAL APPROACH

We divided how to identify dynamics of a networked system into the following steps:

1. Split the network into a set of inter-connected systems.

The first step is to recognize dynamical units of a network that are interconnected. For example, in a road network different road segments can be defined as dynamical units, while the intersections reveal interdependence between traffic flows of different road segments.

2. Define the input vector $u[k]$ and the output vector $y[k]$ of each individual system.

In this step, we examine available measurements of the process taking place in each dynamical unit. Following the direction of the flow, we determine the input vector $u[k]$ and the output vector $y[k]$, as a subset of the measured variables that sufficiently capture the dynamical process. The dimension of the input and the output vector depend on available measurements. For example, in a road network, traffic flow and speed at the beginning of a road segment compose the input vector $u[k]$. Similarly, the output vector $y[k]$ consists of the traffic flow and the speed at the end of a road segment. In general, it is important that vectors $u[k]$ and $y[k]$ define variables of the same type because the input vector $u[k]$ of one system may also be the output vector $y[k]$ of an adjacent dynamical unit, see Eq.(3.12).

3. Identify the dynamics with the discrete-time linear state space model (DLSS).

Since we have defined the input vector $u[k]$ and the output vector $y[k]$ in the previous step, the dynamics of each individual system are obtained through the discrete-time linear state space model (DLSS) defined in Eq.(2.6).

4. Create a networked system based on the underlying topology

In this step, we create a networked system, which consists of different interconnected dynamical units defined in the previous steps. Depending on how

the different dynamical units are connected, we distinguish internal links, external input links and external output links to present a diagram of the networked system. For example, in a road network, a networked system consists of different road segments and junctions.

Based on the diagram of the networked system, we define the number of nodes/systems in the network N , number of external inputs r and number of external outputs q . Also the $N \times N$ adjacency matrix W is defined to illustrate how nodes are connected in the networked system. Similarly, we obtain the $r \times N$ matrix Φ (Eq.(3.1)) and the $N \times q$ matrix Ψ (Eq.(3.2)) to present how the external input nodes and external output nodes connect to the networked system respectively. Finally, we define the $r \times q$ matrix Z (equation 3.3) to illustrate the external links between input nodes and output nodes.

According to the input vector $u[k]$ and the output vector $y[k]$ of different dynamical units in the networked system, we obtain the $N \times 1$ vector n (Eq.(3.5)) to represent the number of states for each node/system of the networked system. Similarly, we define the $N \times 1$ vector m (Eq.(3.6)) and $N \times 1$ vector p (Eq.(3.7)) to represent the dimension of the input vector u_i and the output vector y_i for each system. For example, since we choose traffic flow and speed to capture the dynamical process, the dimension of them are all 2. We also define the $r \times 1$ vector μ (Eq.(3.8)) and the $q \times 1$ vector ρ (Eq.(3.10)) to store the dimension of each external input vector and output vector respectively.

5. Relate the input vector $u[k]$ and the output vector $y[k]$ of each individual system, based on network topology.

According to the diagram of the networked system defined in the previous step, we relate the input vector $u[k]$ and the output vector $y[k]$ of each individual system. The input vector $u[k]$ of one dynamical unit may be the output vector $y[k]$ of another units.

6. Combine the inter-connected DLSS models into a single DLSS model.

In this step, since we have identified the DLSS models of individual dynamical units and created the networked system in the previous steps, the single DLSS model of the networked system is obtained from the Eq.(3.15) and Eq.(3.16). And the code for this part is provided in Appendix C.

7. Analyze the estimation accuracy of the obtained single DLSS model.

In this step, we obtain the mean squared error to illustrate the estimation accuracy of the identification approach. Compared with the common black-box identification approach, this new identification approach improves the estimation accuracy because it consider the interconnected dynamical units.

3.2.2. SIMPLE TOY EXAMPLE

In order to help the reader to understand how to apply this new identification approach, we present a simple example, see Figure 3.4.

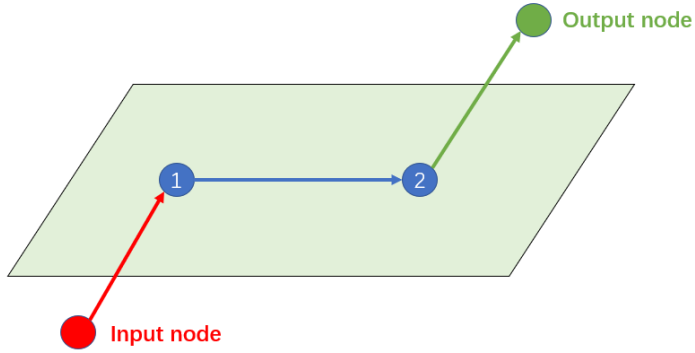


Figure 3.4: A toy example with two internal nodes

This network consist of $N = 2$ nodes/systems, $r = 1$ external input and $q = 1$ external output. Then we define the following information:

2×1 vector n to represent the number of states for each node/system of the network

$$n = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \quad (3.17)$$

2×1 vector m to represent the dimension of the input vector u_i for each system

$$m = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \quad (3.18)$$

2×1 vector p represent the dimension of the output vector y_i for each system

$$p = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \quad (3.19)$$

1×1 vector μ represent the dimension of each external input vector

$$\mu = [2] \quad (3.20)$$

1×1 vector ρ represent the dimension of each external output vector

$$\rho = [2] \quad (3.21)$$

$(N \times N)$ Adjacency matrix of the network W

$$W = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad (3.22)$$

$(r \times N)$ Topology matrix of how external input nodes are connected to internal nodes Φ

$$\Phi = [1 \ 0] \quad (3.23)$$

$(N \times q)$ Topology matrix of how internal nodes are connected to external input nodes Ψ

$$\Psi = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (3.24)$$

Because no links between external input node and external input node, so Z are all zeros.

First we can identify the two internal DLSS dynamics. Then we can use the theory in section 3.1.3 to get the the matrices A_e , B_e , C_e and D_e . The code can be found in the Appendix C.

$$A_e = \begin{bmatrix} 0.0243 & 0.06076 & -0.6214 & 0 & 0 & 0 \\ 0.0257 & -0.3672 & -0.6372 & 0 & 0 & 0 \\ 0.8687 & 0.1935 & 0.1142 & 0 & 0 & 0 \\ -0.7219 & 0.1378 & 0.2239 & 0.7650 & 0.0965 & 0.0534 \\ 0.0156 & -0.5714 & 0 & 0.0965 & 0.5147 & -0.1154 \\ 0.0116 & -0.4249 & 0 & 0.0534 & -0.1154 & 0.7397 \end{bmatrix} \quad (3.25)$$

$$B_e = \begin{bmatrix} 0.3919 & -0.7411 \\ 0 & -0.5078 \\ -0.9480 & -0.3206 \\ 0 & -0.3752 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (3.26)$$

$$C_e = \begin{bmatrix} 7.5314 & -3.0891 & -2.3215 & 1.1921 & -0.0245 & 0 \\ -0.0009 & 0.0324 & 0 & -1.6118 & -1.9488 & 0.8617 \end{bmatrix} \quad (3.27)$$

$$D_e = \begin{bmatrix} 0 & 3.8912 \\ 0 & 0 \end{bmatrix} \quad (3.28)$$

4

APPLICATIONS AND RESULTS

In this chapter, we apply our new approach to identify the dynamics of two real-world networks and analyse the estimation accuracy. First we split the real-world network into a set of inter-connected systems and identify the dynamics of each individual system. Then we combine the inter-connected DLSS models based on the underlying connections of individual systems into a single DLSS model capturing the dynamics of the entire network.

4.1. WATER NETWORKS

Cities and towns receive their water from distribution networks. Water is first drawn from natural sources (such as lakes, rivers, aquifers, etc.), and it is sent to water-treatment plants to be purified. Then, it is distributed to the population [23]. Therefore, it is of great practical importance to study the dynamics of water networks. The analysis is performed with time series of a water network, simulated by the software EPANET [24].

4.1.1. BASIC INFORMATION ABOUT THE NETWORK

EPANET models a water distribution system as a collection of links connecting nodes. The links represent pipes, pumps, and control valves. The nodes represent junctions, tanks, and reservoirs. Figure 4.1 below illustrates how these objects can be connected to one another to form a network. Then we will briefly define each of the mentioned elements.

Junctions

Junctions are points in the network where links join together and where water enters or leaves the network. The basic input data requirements for junctions are:

1. Elevation above some reference (usually mean sea level)
2. Water demand (rate of withdrawal from the network)
3. Initial water quality

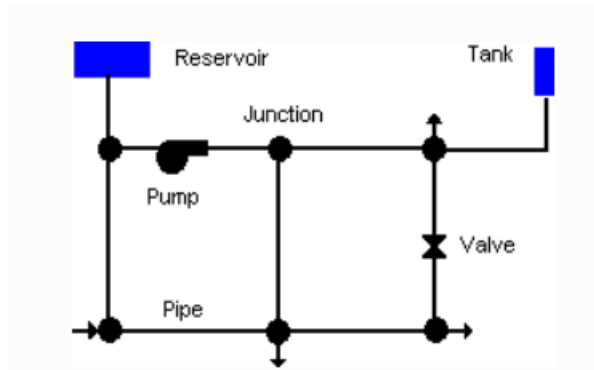


Figure 4.1: Underlying topology of water network

4

The output results computed for junctions at all time periods of a simulation are:

1. Hydraulic head (internal energy per unit weight of fluid)
2. Pressure
3. Water quality

Reservoirs

Reservoirs are modeled as nodes in the water network and represent an endless external source or drain of water. Lakes, rivers, groundwater aquifers, and connections to other systems can be modelled as reservoirs in the EPANET. Water quality source sites can also be reservoirs.

The hydraulic head of a reservoir, which is equivalent to the water surface elevation if the reservoir is not under pressure, and its starting quality for water quality analysis are the two main input properties.

A reservoir's head and water quality cannot be impacted by network activity since it serves as a network's boundary point. It lacks computed output attributes as a result. However, by giving it a time pattern, its head may be made to change over time.

Tanks

Tanks are nodes with storage capacity, and throughout a simulation, the amount of water they can hold can change over time. The following are the main tank input properties:

1. Bottom elevation (where water level is zero)
2. Diameter (or shape if non-cylindrical)
3. Initial, minimum and maximum water levels
4. Initial water quality

Pipes

Water is transported by pipes from one point in the network to another. According to EPANET, all pipelines are always filled. The direction of flow is from the end with higher internal hydraulic head (energy per weight of water) to that with lower internal hydraulic head. The following are the main hydraulic input parameters for pipes:

1. Start and end nodes
2. Diameter
3. Length
4. Roughness coefficient (for determining headloss)
5. Status (open, closed, or contains a check valve)

Computed outputs for pipes include:

1. Flow rate
2. Velocity
3. Headloss
4. Darcy-Weisbach friction factor
5. Average reaction rate (over the pipe length)
6. Average water quality (over the pipe length)

Pumps

Pumps are components that generate a fluid energy, increasing its hydraulic head. The start and end nodes as well as the pump curve are the main input parameters for a pump (the combination of heads and flows that the pump can produce). Instead of a pump curve, a constant energy device that provides the fluid with a constant quantity of energy (horsepower or kilowatts) for all combinations of flow and head might be used to depict the pump.

Simulations

We can obtain the simulation data of some variables over time such as flow and velocity for pipes, pressure, demand and head for junctions. We can export them to construct the data set to analyze the dynamics of the water network based the underlying topology. For example Figure 4.2 illustrates simulation data for flow of one pipe over one day and Figure 4.3 provides the simulation data for pressure of one junction over one day.

4.1.2. IDENTIFICATION APPROACHES AND RESULTS

Firstly, we split the water network into a set of inter-connected systems and focus on the identification approach of each individual system.

Identification approach 1

In this identification approach, we model the flow of each pipe as a DLSS model. Figure 4.4 illustrates the underlying topology formed by three pipes. The flow of a pipe

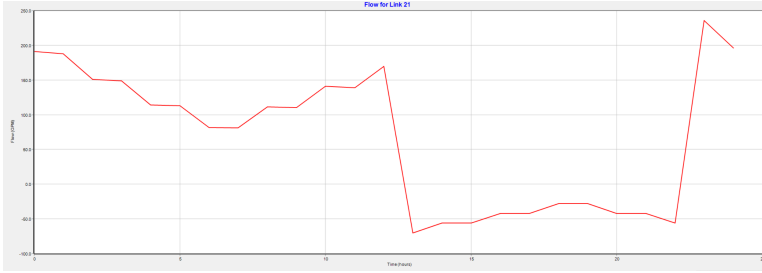


Figure 4.2: Flow simulation over one day

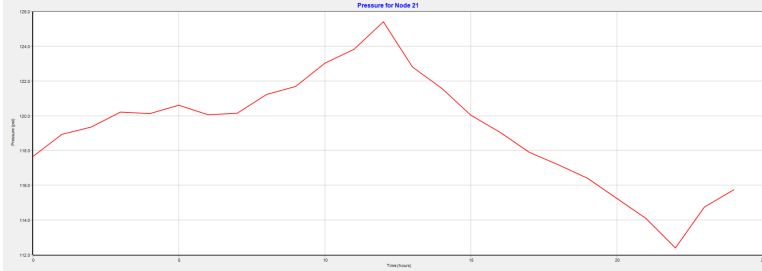


Figure 4.3: Pressure simulation over one day

is denoted as $f[k]$, while the pressure inside a junction is denoted as $p[k]$. We define the flow $f_i[k]$ out of pipe i as the output vector $y_i[k]$ of the i -th system. From Figure 4.4, we conclude that the flow $f_1[k]$ of pipe 1 and flow $f_2[k]$ of pipe 2 together compose the flow $f_3[k]$ of pipe 3. In addition, we define the pressure $p_1[k]$ of junction 1 and the pressure $p_2[k]$ of junction 2 as the external input vector for system i . Therefore, the input vector $u_3[k]$ is defined as follows

$$u_3[k] = [p_1[k] \quad p_2[k] \quad f_1[k] \quad f_2[k]]^T$$

Then we can construct a DLSS model to estimate the flow of pipe 3 as output vector $y[k]$ like Figure 4.5.

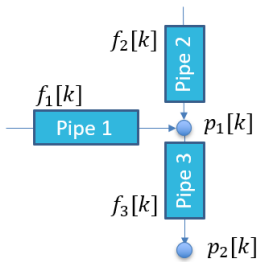


Figure 4.4: Identification approach 1 of water network



Figure 4.5: DLSS model of identification approach 1

We compare the estimated flow $\hat{f}_3[k]$ of the identified DLSS model with the measured

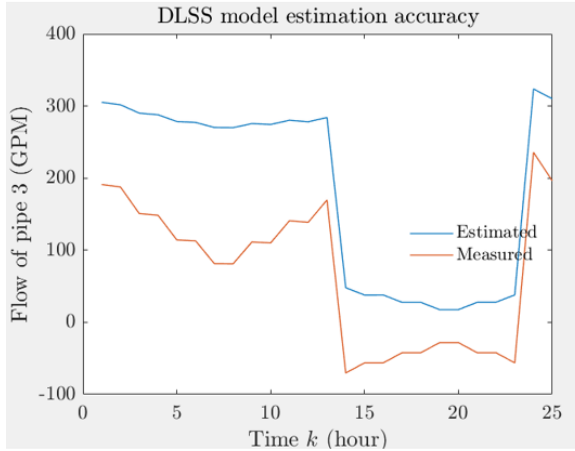


Figure 4.6: Estimation accuracy of approach 1

flow $f_3[k]$ and provide results in Figure 4.6. The estimated accuracy is surprisingly low. Therefore, in the following part we considered adding additional variables to the DLSS model, in order to improve estimated accuracy.

Identification approach 2

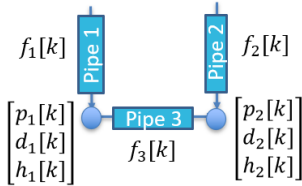


Figure 4.7: Identification approach 2 of water network

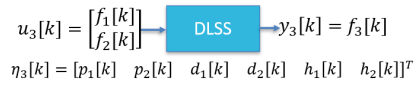


Figure 4.8: DLSS model of identification approach 2

In this identification approach, we model the pipe as a node, while the water flow within the pipe we model as a DLSS model like figure 4.7. We conclude that the flow $f_1[k]$ of pipe 1 and flow $f_2[k]$ of pipe 2 together compose the flow $f_3[k]$ of pipe 3. But we redefine the external input vector $\eta[k]$, by adding the following variables:

1. $d[k]$: Water demand (rate of withdrawal from the network)
2. $h[k]$: Hydraulic head (internal energy per unit weight of fluid)

Therefore, the input vector $u_3[k]$ is defined as follows

$$u_3[k] = [p_1[k] \quad p_2[k] \quad d_1[k] \quad d_2[k] \quad h_1[k] \quad h_2[k] \quad f_1[k] \quad f_2[k]]^T$$

Then we can construct a DLSS model to estimate the flow of pipe 3 as output vector $y[k]$ like Figure 4.8. Then we use the identification algorithm talked in Section 2.2.2 to compare the estimated flow result of pipe 3 with the measured flow from EPANET.

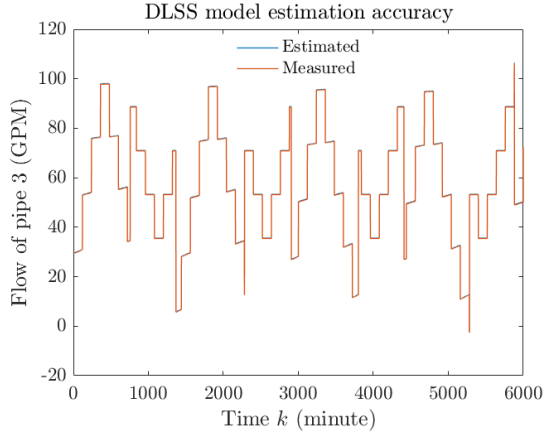


Figure 4.9: Estimation accuracy of approach 2

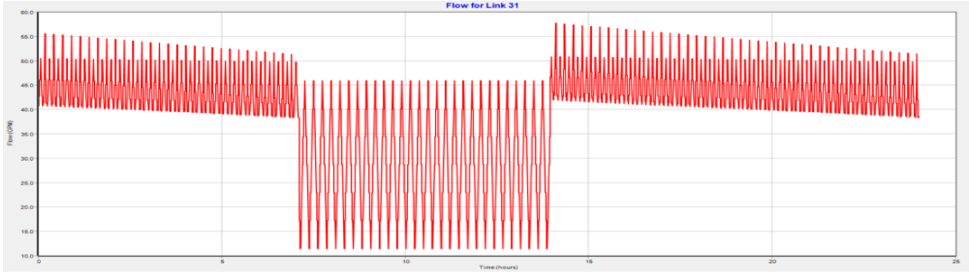


Figure 4.10: Flow simulation reducing time interval

From figure 4.9, we can see the estimation accuracy of second approach is very high. But the values of the $p_i \times m_i$ feed-forward matrix D_i inform us that the flow $y_3[k] = f_3[k]$ is estimated as the linear combination of the flows $f_1[k]$ and $f_2[k]$ of two adjacent pipes. In other words, no internal linear dynamics are captured by the DLSS model.

$$D = \begin{bmatrix} 0.004 & -0.003 & 0.001 & 0.000 & 0.067 & 0.0970 & 0.5159 & 0.4841 \end{bmatrix} \quad (4.1)$$

Conclusion

Figure 4.10 illustrates the flow simulation from EPANET when reducing the time interval to 1 minute. The flow change is immediate, which means there are no transient processes in the time series. In other words, EPANET at each time step k provides the estimated steady state as the current state of the water network. Consequently, transient processes are not present. Therefore, the identification approach is not able to capture internal dynamics of individual systems.

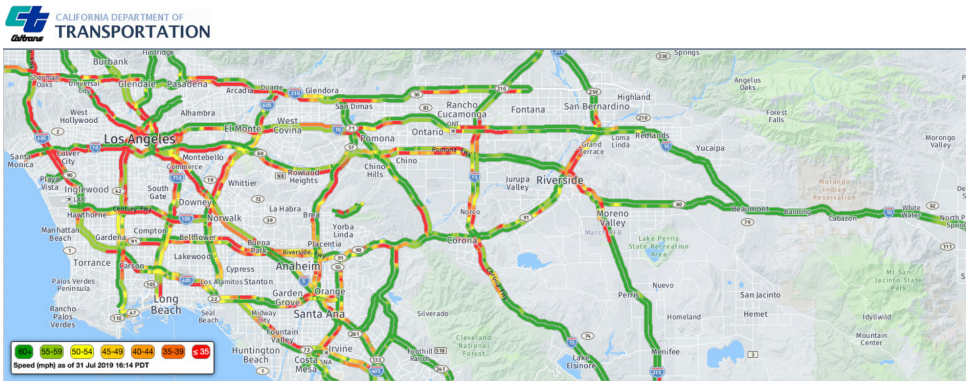


Figure 4.11: Caltrans Performance Measurement System

4.2. ROAD NETWORKS

An urban road network consists of a variety of different functions within the jurisdiction of the town and regional roads, which form the skeleton of the overall urban planning layout and can provide safe, rapid, economic, and comfortable driving conditions for all kinds of transportation. The development of urban road networks around the world is related to the development of industry, the concentration of the urban population, traffic and transport development.

Nowadays congestion problems caused by cars in urban areas have made life uncomfortable and inconvenient. A traffic congestion deteriorates life quality. In more detail, a traffic congestion causes travel delays, increased fuel consumption and overall costs [25]. There was a 17% increase in the volume of traffic jams on Dutch roads in 2020, according to the annual summary by motoring organisation ANWB [26]. So it is important to explore and analyze the dynamics of the road network.

4.2.1. BASIC INFORMATION ABOUT THE NETWORK

In this thesis, we use the data set from Caltrans Performance Measurement System (PeMS). The traffic data displayed on the map depicted in Figure 4.11 is collected in real-time from over 39,000 individual detectors. From different sensors (red points in Figure 4.12), we obtain the measurements of the traffic flow, speed, both per each lane and in total. An example of measured flow over a road segment composed of four lanes, during a period of 7 days is presented in Figure 4.13.

We consider a road segment as a dynamical unit. By defining relations between different road segments, we model the entire road network as a networked system. For an individual road segment, we collect the time series of the flow and speed from the beginning of the segment and at the end of it. These time series are considered as the input and output vectors. We identify the dynamics between the input and the output vector as the discrete-time linear state-space model.

In our research, we adopt the smallest available time step $T = 5$ minutes. Therefore, the flow $f[k]$ is described in units [Veh/5min], while the speed $v[k]$ is measured in [mph].

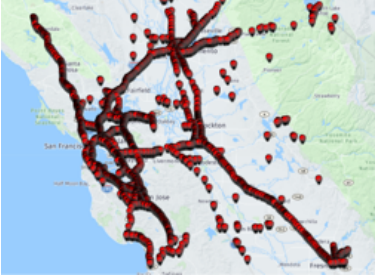


Figure 4.12: Sensors in the road network

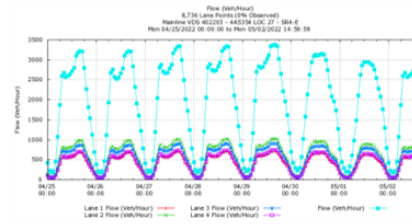


Figure 4.13: Simulation from the road map

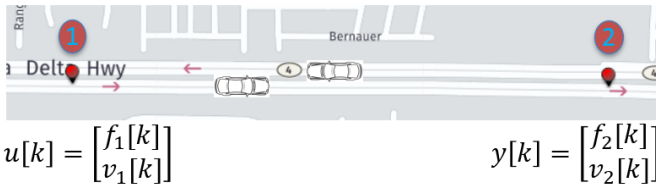


Figure 4.14: Modeling a single road segment as a dynamical unit.



Figure 4.15: Block-diagram of a single road segment.

4.2.2. IDENTIFICATION APPROACHES AND RESULTS

In this subsection, we develop the identification approach that combines the dynamics of individual road segments and the underlying topology of the road network, to provide a model for the dynamics of the entire network.

Identification approach 1

In the first approach, we analyse the dynamics of a single road segment. We model a single road segment as a dynamical unit, as depicted in Figure 4.14. In the measured part, vehicles enter from point 1, then exit from point 2, and we measure the flow at this two points as $f_1[k]$ and $f_2[k]$, also the speed at the two points as $v_1[k]$ and $v_2[k]$. The measured vehicle flow $f_1[k]$ and speed $v_1[k]$ at the beginning of the segment compose the 2×1 input vector $u[k] = [f_1[k], v_1[k]]^T$, while the measured variables $f_2[k]$ and $v_2[k]$ from the end of the segment constitute the 2×1 output vector $y[k] = [f_2[k], v_2[k]]^T$. Next, we identify the dynamics between the input vector $u[k]$ and the output vector $y[k]$ with the DLSS model in Eq.(3.4). The output of the DLSS model, denoted as $\hat{y}[k] = [\hat{f}_2[k], \hat{v}_2[k]]^T$, provides an estimation of the flow $f_2[k]$ and the speed $v_2[k]$ at the end of the road segment. The block diagram of a single road segment is provided in Figure 4.15.

We present the identification results for this single road segment mode in Figure 4.16.

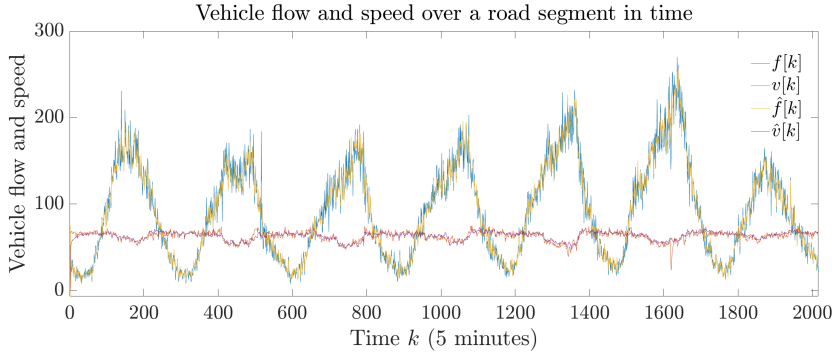


Figure 4.16: Vehicle flow and speed in a road segment

The estimation accuracy is high, informing us that a DLSS model successfully captures the dynamics of vehicles propagating over a road segment in time.

In Figure 4.17 and Figure 4.18, we depict the estimation error separately for the vehicle speed $f_2[k]$ and speed $v_2[k]$, respectively. Both the flow $f_2[k]$ and the speed $v_2[k]$ are estimated with comparable accuracy. From the estimation error of speed, a larger negative error indicates a possible congestion. It is useful for us to monitor road conditions for a single road segment.

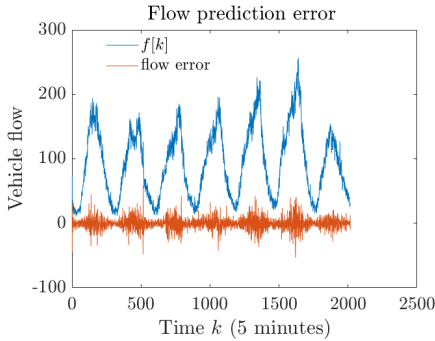


Figure 4.17: Flow estimated error

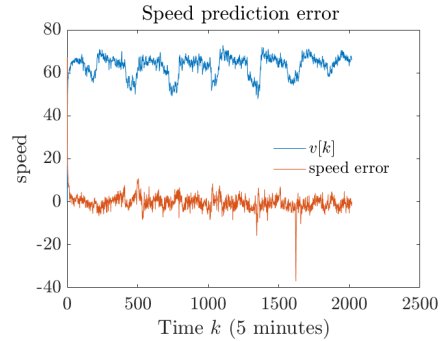


Figure 4.18: Speed estimated error

Identification approach 2

For the second approach, we model a road junction with m road segments with m different DLSS models, as depicted in Figure 4.20. Vehicles arrive at a junction from different road segments, turn in different directions and leave the junction.

When we focus on the dynamics of the junction, we analyze multiple road segments as a whole system, unlike in identification approach 1. The vehicle flow and speed of the incoming traffic from all road segments of the junction constitute the $2m \times 1$ input vector $u[k]$ for each DLSS model, while each DLSS model estimates the flow and speed of the outgoing traffic flow per each road segment.

Vector $g_m[k]$ and $w_m[k]$ present the measurements of vehicle flow and speed

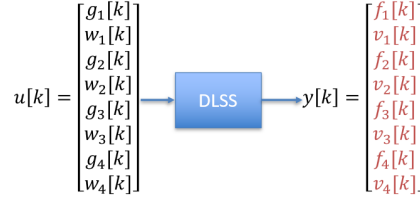


Figure 4.19: Road junction DLSS model

that enter into the junction from road segment m respectively. Similarly, the measurements of traffic flow and speed that drive out from road segment m are obtained as $f_m[k]$ and $v_m[k]$. The vehicle flow and speed of the incoming traffic from this four road segments of the junction constitute the 8×1 input vector $u[k] = [g_1[k], w_1[k], g_2[k], w_2[k], g_3[k], w_3[k], g_4[k], w_4[k]]^T$ for this junction DLSS model, while this DLSS model estimates the flow and speed of the outgoing traffic flow per each road segment as output vector $y[k] = [f_1[k], v_1[k], f_2[k], v_2[k], f_3[k], v_3[k], f_4[k], v_4[k]]^T$, as depicted in Figure 4.19.

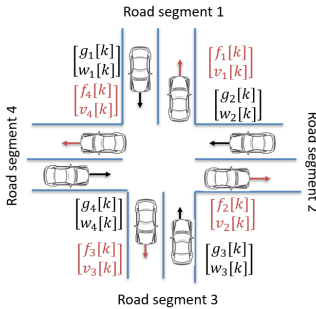


Figure 4.20: A road junction

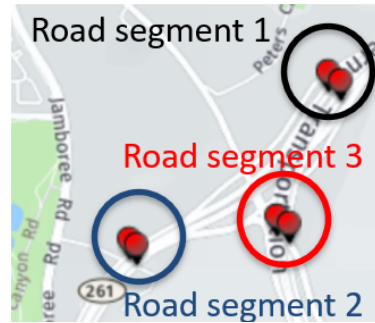


Figure 4.21: A real junction in the road map

Then we apply this model to a real junction in the map which consists of three road segments illustrated as Figure 4.21. In order to present the estimation results clearly, we represent three different road segments separately in Figure 4.22, Figure 4.23 and Figure 4.24. The three road segments all present reasonable estimated result, which means we successfully model the junction as a single DLSS model to identify the dynamics of this three road segments.

From analysing the dynamics of the junction as a single DLSS model, we summarise the following conclusions:

1. Analyze the dynamics in the intersection for different road segments. For example we monitor the road conditions by analyzing the estimated error of speed of different road segments.
2. If we only obtain incoming flows of one road segment, it is impossible to predict

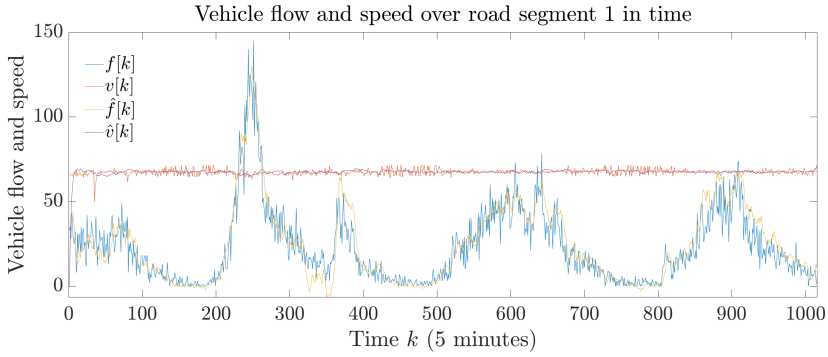


Figure 4.22: Vehicle flow and speed in road segment 1

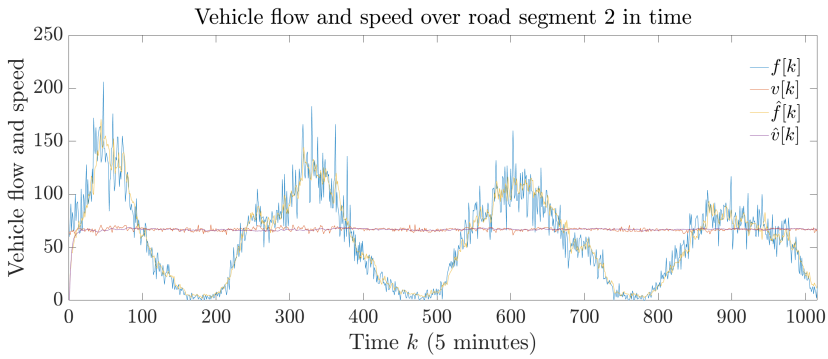


Figure 4.23: Vehicle flow and speed in road segment 2

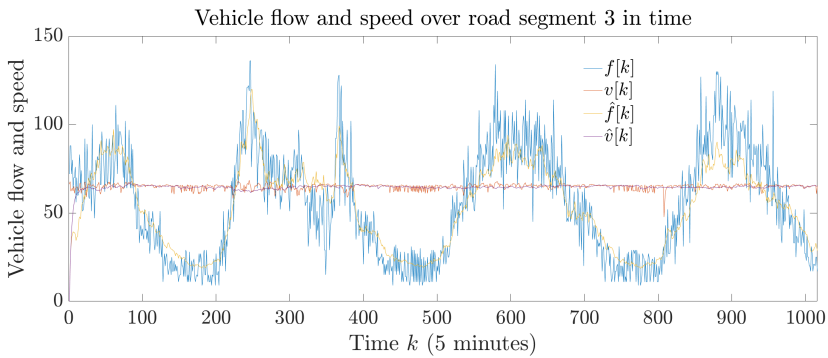


Figure 4.24: Vehicle flow and speed in road segment 3

the outgoing flow of that segment, by analyzing single road segment. Using the second approach, we can do this, by taking into account each road segment of a junction.

Hierarchical Structuring of the California road network

The entire California road network represents an interconnected network of systems of different junctions and road segments. So this two introduced identification approaches of dynamical units allow us to scale the model.

We present a networked road system like Figure 4.25, which consists of one junction and three different road segments. Depending on how this dynamical units are connected, we distinguish internal links, external input links and external output links to present a diagram of this networked road system as Figure 4.26.

4

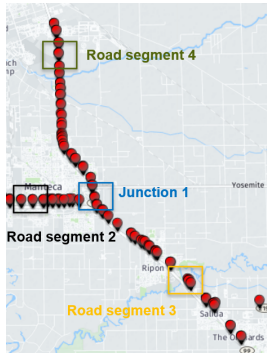


Figure 4.25: A networked road system

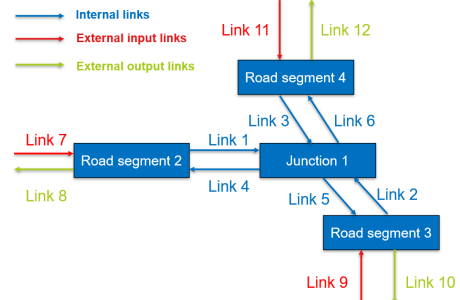


Figure 4.26: block diagram for this networked road system

We divide this networked system to 9 interconnected systems and they are presented as follows. The flow and speed of link m in the networked road system are defined as $f_m[k]$ and $v_m[k]$. The junction 1 is divided to system 1, system 2 and system 3. The road segment 2 is divided to system 4 and system 5. The road segment 3 is divided to system 6 and system 7. Finally road segment 4 is divided to system 8 and system 9.

$$u[k] = \begin{bmatrix} f_1[k] \\ v_1[k] \\ f_2[k] \\ v_2[k] \\ f_3[k] \\ v_3[k] \end{bmatrix} \rightarrow \text{DLSS} \rightarrow y[k] = \begin{bmatrix} f_4[k] \\ v_4[k] \end{bmatrix}$$

(a) System 1 of junction

$$u[k] = \begin{bmatrix} f_1[k] \\ v_1[k] \\ f_2[k] \\ v_2[k] \\ f_3[k] \\ v_3[k] \end{bmatrix} \rightarrow \text{DLSS} \rightarrow y[k] = \begin{bmatrix} f_5[k] \\ v_5[k] \end{bmatrix}$$

(b) System 2 of junction

$$u[k] = \begin{bmatrix} f_1[k] \\ v_1[k] \\ f_2[k] \\ v_2[k] \\ f_3[k] \\ v_3[k] \end{bmatrix} \rightarrow \text{DLSS} \rightarrow y[k] = \begin{bmatrix} f_6[k] \\ v_6[k] \end{bmatrix}$$

(c) System 3 of junction

Firstly, we construct 9 DLSS models to analyze the dynamics for these 9 interconnected dynamical units individually according to the two identification approaches presented before.

Based on the diagram of the networked road system, we define the relate matrices and vectors, which are illustrated in step 4 of our identification approach. Next, we relate the input vector $u[k]$ and the output vector $y[k]$ of each individual system, based on



Figure 4.27: System 4 of road segment 2



Figure 4.28: System 5 of road segment 2



Figure 4.29: System 6 of road segment 3



Figure 4.30: System 7 of road segment 3



Figure 4.31: System 8 of road segment 4



Figure 4.32: System 9 of road segment 4

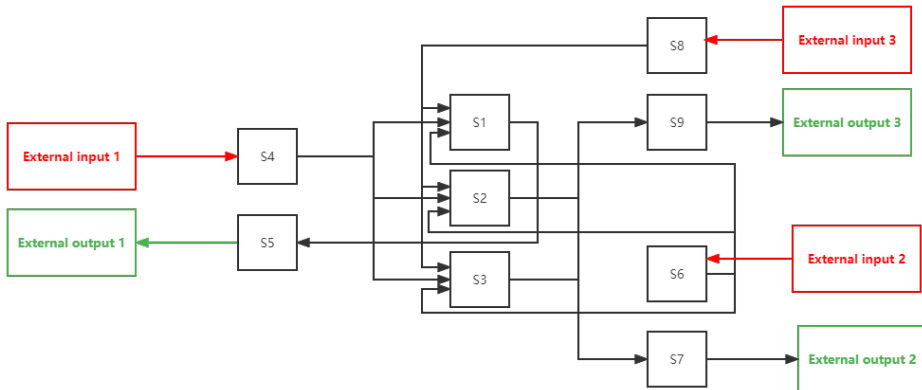


Figure 4.33: Interconnected 9 systems

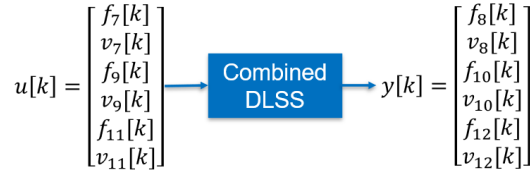


Figure 4.34: Combined DLSS model for the networked road system

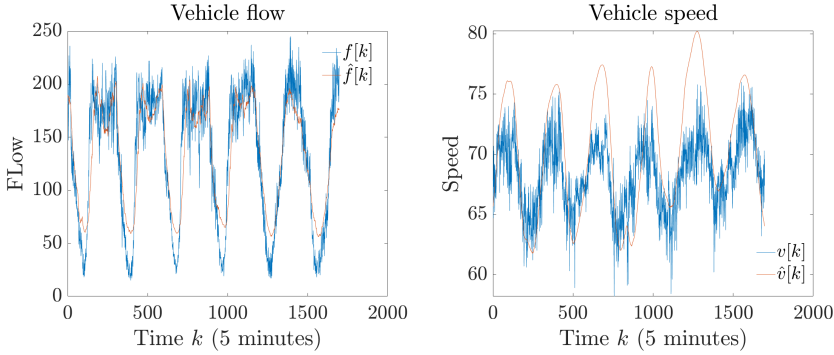


Figure 4.35: External output 1

topology of this networked road network, so that we construct a diagram to illustrate this 9 interconnected systems in Figure 4.33.

Then we combine this 9 DLSS models into a single DLSS model according to the step 6 of our identification approach. The vehicle flow and speed of the external input links constitute the 6×1 input vector $u[k] = [f_7[k], v_7[k], f_9[k], v_9[k], f_{11}[k], v_{11}[k]]^T$, and this single DLSS model estimates the flow and speed of external output links as 6×1 output vector $y[k] = [f_8[k], v_8[k], f_{10}[k], v_{10}[k], f_{12}[k], v_{12}[k]]^T$, as depicted in Figure 4.34.

In order to illustrate the estimated results clearly, we divided the external outputs as three individual Figures depicted as 4.35, 4.36 and 4.37. From this Figures we successfully combined the nine interconnected DLSS models to a single DLSS model to present the dynamics of the external output vectors of this networked road system within an acceptable error.

Error comparison

In order to compare the performance with the common black-box identification approach, we present the mean squared error (MSE). In common black-box identification approach, we ignore the internal topology and the different individual dynamical unit of the networked system, as depicted in Figure 4.25. From the table 4.1, the MSE reduce if using the combined DLSS model. Therefore using our new identification approach in the real-world road network improves the estimation performance. That means we provides a more efficient way to analyze the dynamics of linear real-world complex networks

Limitations of the dataset

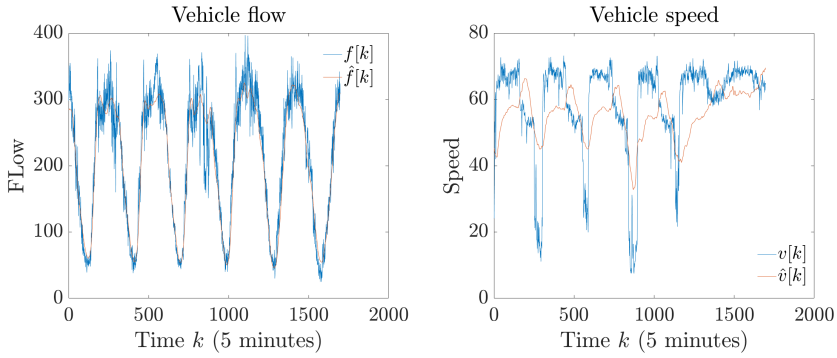


Figure 4.36: External output 2

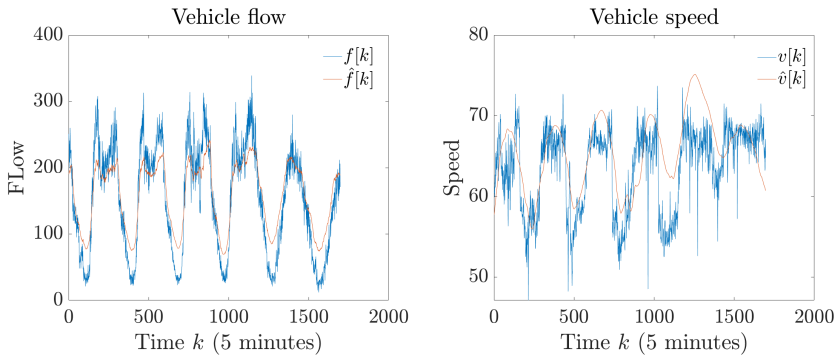


Figure 4.37: External output 3

Model	$u[k]$ → DLSS → $y[k]$ <small>External input</small> <small>External output</small>	$u[k]$ → Combined DLSS → $y[k]$ <small>External input</small> <small>External output</small>
MSE	438.8	64.21

Table 4.1: MSE Comparison

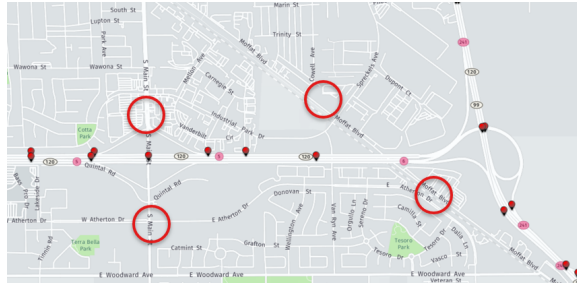


Figure 4.38: Limitations of the dataset

In this part we present limitations of the road data set in our model, which cause the estimated error in the combined systems. When we construct the data set for the interconnected systems, there are many small road crossings (red circles showed in Figure 4.38), which have no sensors to indicate vehicle data but there must be a small number of vehicles passing. Though the dataset contains these small limitations, the results of applying our linear combined theory perform a reasonable estimation accuracy, which further demonstrates the implementation ability of our identification method.

A future study for road network

A region of California Road Network can be divided into a set of interconnected dynamical units. For example, the area in Figure 4.39 is divided into four interconnected systems.

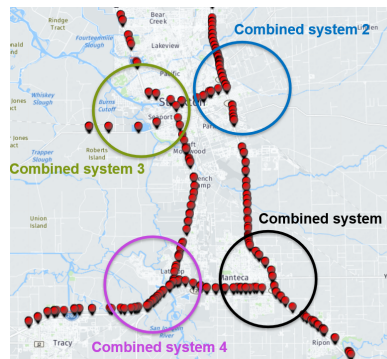


Figure 4.39: Area with multiple combined systems

Since we have successfully constructed a single DLSS model to analyze the dynamics of each networked systems, we easily analyse the dynamics of a bigger networked road system illustrated in Figure 4.39.

5

CONCLUSIONS AND FUTURE RESEARCH

This chapter is the last chapter summarizing our findings and proposing some suggestions for future research.

5.1. CONCLUSIONS

In this thesis, we apply a new networked system identification approach to real-world networks. Unlike common approach, we split the real-world network into small units and identify the dynamics of them first, then relate their real input/output vectors to identify the dynamics of the entire network. In chapter 4, we apply this new approach to two real-world networks, water networks and road networks.

For water networks, we model each pipe in the water network as a small dynamical unit, then we identify the dynamics of the individual units independently through the DLSS model. Though the estimation accuracy of this small unit is high, the pipe flow is only estimated as a linear combination of the flow in adjacent pipes from examining the the values of the $p_i \times m_i$ feed-forward matrix D_i . EPANET at each time step k provides the estimated steady state as the current state of the water network. Consequently, transient processes are not present. Therefore, the identification approach is not able to capture internal dynamics of individual systems.

For road networks, we first analyze the dynamics of two dynamical units, a single road segment and a junction. The dynamics of these two small units of the California road network are illustrated in reasonable error. Next we create a networked road system which consists of these two small units. Depending on how this dynamical units are connected, we distinguish internal links, external input links and external output links to present a diagram of this networked road system. Next we relate their real input and output vectors according to the structure of the networked road system and provide a single DLSS model to identify the dynamics of the entire networked system. In addition, we found that applying this new approach to road networks greatly increases

the estimation accuracy compared with the black-box identification approach.

Overall, in this thesis, we successfully apply this new identification method to real road networks in California and prove that it greatly increase the estimation accuracy, which not only validates the implementability of this method, but more importantly, provides a more efficient way to analyze the dynamics of linear real-world complex networks.

5.2. FUTURE RESEARCH

In this section, we propose some directions for future research.

1. Find a suitable data set of real water networks in order to see the results of applying this new identification approach.
2. In road networks, the estimation accuracy of speed is not perfect when focusing on the dynamics of the combined system. Explore the causes and solve them. Perhaps include feedback vectors in various parts of the combined system.
3. We only illustrate the estimation accuracy of our identification approach, so explore whether our approach can predict the dynamics is also a future study.
4. This identification approach works better only when the dynamics of the network is close to linear, so how to deal with the dynamic analysis of nonlinear complex networks is also a future research direction.

BIBLIOGRAPHY

- [1] M. E. J. Newman, "The structure and function of complex networks," SIAM Review, vol. 45, no. 2, pp. 167–256, 2003.
- [2] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks: Structure and dynamics," Physics Reports, 2006.
- [3] P. v. Mieghem, Graph Spectra for Complex Networks. Cambridge University Press, 2010.
- [4] S. H. Strogatz, "From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators," Physica D: Nonlinear Phenomena, vol. 143, no. 1, pp. 1–20, 2000.
- [5] N. Masuda, M. Porter, and R. Lambiotte, "Random walks and diffusion on networks," Physics Reports, vol. 716, 12 2016.
- [6] B. Prasse and P. V. Mieghem, "The viral state dynamics of the discrete-time NIMFA epidemic model," IEEE Transactions on Network Science and Engineering, vol. 7, no. 3, pp. 1667–1674, 2020.
- [7] M. van den Berg, A. Hegyi, B. De Schutter, and J. Hellendoorn, "A macroscopic traffic flow model for integrated control of freeway and urban traffic networks," in 42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475), vol. 3, 2003, pp. 2774–2779 Vol.3.
- [8] A. Barrat, M. Barthélemy, and A. Vespignani, Dynamical Processes on Complex Networks. Cambridge University Press, 2008.
- [9] B. Barzel, Y.-Y. Liu, and A.-L. Barabasi, "Constructing minimal models for complex system dynamics," Nature communications, vol. 6, p. 7186, 05 2015.
- [10] L. Xiang, F. Chen, W. Ren, and G. Chen, "Advances in network controllability," IEEE Circuits and Systems Magazine, vol. 19, no. 2, pp. 8–32, 2019.
- [11] Y.-Y. Liu and A.-L. Barabási, "Control principles of complex systems," Rev. Mod. Phys., vol. 88, p. 035006, Sep 2016.
- [12] J. Schoukens and L. Ljung, "Nonlinear system identification: a user-oriented road map," IEEE Control Systems, vol. 39, no. 6, pp. 28–99, Dec. 2019.
- [13] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," Proceedings of the IEEE, vol. 95, no. 1, pp. 215–233, 2007.

- [14] L. Xiang, P. Wang, F. Chen, and G. Chen, "Controllability of directed networked MIMO systems with heterogeneous dynamics," IEEE Transactions on Control of Network Systems, vol. 7, no. 2, pp. 807–817, jun 2020.
- [15] I. Jokic and P. Van Mieghem, "Linear processes on complex networks," Journal of Complex Networks, vol. 8, no. 4, pp. 1–41, 2020.
- [16] Z. Yang, R. Guo-sheng, W. Lin, and S. Wen-jun, "Predictive algorithm for duffing demodulation system by Kalman gain," in 2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS), 2014, pp. 1–4.
- [17] T. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," Automatica, vol. 47, pp. 39–49, 01 2011.
- [18] G. Ascheid, "Estimation and detection theory, lecture notes." ICE, RWTH Aachen University (2013).
- [19] E. Lehmann and G. Casella, Theory of Point Estimation, ser. Springer Texts in Statistics. Springer New York, 2006.
- [20] B. De Moor, J. Vandewalle, M. Moonen, L. Vandenberghe, and P. Van Mieghem, "A geometrical strategy for the identification of state space models of linear multivariable systems with singular value decomposition," IFAC Proceedings Volumes, vol. 21, no. 9, pp. 493–497, 1988, 8th IFAC/IFOORS Symposium on Identification and System Parameter Estimation 1988, Beijing, PRC, 27-31 August.
- [21] G. Golub and C. Van Loan, Matrix Computations, ser. The North Oxford Academic paperback. North Oxford Academic, 1983.
- [22] M. Verhaegen and V. Verdult, Filtering and System Identification: A Least Squares Approach. Cambridge University Press, 2007.
- [23] J. Delgado-Aguíñaga, F. Becerra-López, L. Torres, G. Besançon, V. Puig, and M. R. J. Magaña, "Dynamic model for a water distribution network: application to leak diagnosis and quality monitoring," IFAC-PapersOnLine, vol. 53, no. 2, pp. 16 679–16 684, 2020, 21st IFAC World Congress.
- [24] A. Downs, "Traffic: Why it's getting worse, what government can do," Mar 2022. [Online]. Available: <https://www.brookings.edu/research/traffic-why-its-getting-worse-what-government-can-do/>
- [25] —, "Traffic: Why it's getting worse, what government can do," Mar 2022. [Online]. Available: <https://www.brookings.edu/research/traffic-why-its-getting-worse-what-government-can-do/>
- [26] "The dutch roads are getting busier, traffic congestion rises 17%," Jan 2020. [Online]. Available: <https://www.dutchnews.nl/news/2019/12/the-dutch-roads-are-getting-busier-traffic-congestion-rises-17/>

A

APPENDIX A

```
1 function [A,B,C,D,n_estim,x_0] = C20220308_SubspIdent_v1(input,output,max_order,i)
2
3 clc
4 %% ~~~~~STEP 1~~~~~
5 %Input sequence
6 u = input;
7 %Output sequence
8 y = output;
9
10 %% ~~~~~STEP 2~~~~~
11 % Check and compare dimensions of the measurements
12
13 % m - number of input signals
14 % l - number of output signals
15 [ m, nu ] = size(u);
16 [ l, ny ] = size(y);
17
18 % First check: if the length of measurements are the same
19 if nu ~= ny
20     error('Number of samples of input and output variable differs!!!')
21 end
22
23 %% ~~~~~STEP 3~~~~~
24 % Determine two coefficients that define dimensions of block Hankel
25
26 j = nu - i + 1;
27
28 %% ~~~~~STEPS 4 & 5~~~~~
29 % Create the Hankel matrices from input/output data
30
31 U_H = blockHankel(u, i, j);
32
```

A

```

33 Y_H = blockHankel(y, i, j);
34
35 %% ~~~~~STEP 6~~~~~
36 % Perform the SVD over matrices [Yh*Uhoc*Uhoc']
37
38 [~,S,V] = svd( U_H );
39 r = sum(sum( S>(max(size(S)) * eps(norm(S)))));
40 disp(r)
41 U_Hoc = V( :, r+1:j);
42
43
44 [P,S,~] = svd( Y_H*(U_Hoc*U_Hoc') );
45
46 n_estim = min([sum(sum( S>(max(size(S)) * eps(norm(S) )))), i, max_order ] );
47 disp(sum(sum( S>(max(size(S)) * eps(norm(S) )))))
48
49
50 %% ~~~~~STEP 7~~~~~
51 % Estimate matrices A and C
52 P1 = P( :, 1:n_estim );
53 P2 = P(:, n_estim+1:end);
54 C = P1( 1:l, : );
55
56 U1 = P1( 1:(i-1)*l, : );
57 U2 = P1( l+1:i*l, : );
58
59 A = U1\U2;
60
61
62 %% ~~~~~STEP 8~~~~~
63 % Estimate B and D matrices
64 K = P2'*Y_H*pinv(U_H);
65
66 [D B]=compute_D_B(K,P2',i,j,l,m,n_estim,A,C);
67
68 %% ~~~~~STEP 9~~~~~
69 clc
70 disp('Order of system:')
71 disp(n_estim)
72 disp('Matrix A:')
73 disp(A)
74
75 disp('Matrix B:')
76 disp(B)
77
78 disp('Matrix C:')
79 disp(C)
80
81 disp('Matrix D:')

```

```

82 disp(D)
83
84 figure
85 subplot(2,1,1)
86 plot(real(eig(A)),imag(eig(A)),'*')
87 subplot(2,1,2)
88 plot(abs(eig(A)))
89
90 Y_est = dlsim(A,B,C,D,u);
91 figure
92 plot(y)
93 hold on
94 plot(Y_est)
95 disp('Press Enter to continue')
96 pause()
97 clc
98 close all
99
100 end
101
102 %% Used functions
103 function H = blockHankel(y,i,j)
104
105 % Make a (block)-row vector out of y
106 [l,nd] = size(y);
107 if nd < l
108     y = y';
109     [l,nd] = size(y);
110 end
111
112 % Check dimensions
113 if i < 0
114     error('blkHank: i should be positive');
115 end
116 if j < 0
117     error('blkHank: j should be positive');
118 end
119 if j > nd-i+1
120     error('blkHank: j too big');
121 end
122
123 H=zeros(l*i,j);
124 for k=1:i
125     H((k-1)*l+1:k*l,:)=y(:,k:k+j-1);
126 end
127 end
128
129 function [D, B] = compute_D_B(K,P2t,i,j,l,m,n,A,C)
130 % First check the dimensions

```

A

```

131 if (size(K)~= [l*i-n m*i])
132     error('Dimensions of K are not correct');
133 end
134
135 if (size(P2t)~= [l*i-n l*i])
136     error('Dimensions of P2t are not correct');
137 end
138
139 % Building matrix Km
140 Km=zeros(i*(l*i-n),m);
141 % Km=zeros(i*(l*i-n),m*i);
142 for p=1:i
143     Km((l*i-n)*(p-1)+1:(l*i-n)*p,:)=K(:,m*(p-1)+1:m*p);
144 end
145 % Now matrix Pm
146 Pm=zeros(i*(l*i-n),l*i);
147 Pm1=zeros(i*(l*i-n),l);
148 for p=1:i
149     Pm1((l*i-n)*(p-1)+1:(l*i-n)*p,:)=P2t(:,l*(p-1)+1:l*p);
150 end
151
152 for q=1:i
153     Pm(:,l*(q-1)+1:l*q)=[Pm1((l*i-n)*(q-1)+1:end,:);zeros((q-1)*(l*i-n),l)];
154 end
155 % Building matrix Gm
156 G=[];
157 for p=1:i-1
158     G=[G;C*A^(p-1)];
159 end
160 Gm = [[eye(l) zeros(l,n)];[zeros(l*(i-1),l) G]];
161
162 %Preparing the matrix relation
163 Rl=Km;
164 Rr=Pm*Gm;
165 Un=Rr\Rl;
166 D=Un(1:l,:);
167 B=Un(l+1:end,:);
168 end

```

B

APPENDIX B

Notation	Explanation
G	Graph
N	Number of nodes in graph G
L	Number of links in graph G
W	Adjacency matrix of graph G
A_i	State matrix of a DLSS model of node/system i
B_i	Input matrix of a DLSS model of node/system i
C_i	Output matrix of a DLSS model of node/system i
D_i	Feedforward matrix of a DLSS model of node/system i
A_d	Diagonal block matrix composed of A_i matrices,
B_d	Diagonal block matrix composed of B_i matrices,
C_d	Diagonal block matrix composed of C_i matrices,
D_d	Diagonal block matrix composed of D_i matrices,
A_e	State matrix of a DLSS model of the network
B_e	Input matrix of a DLSS model of the network
C_e	Output matrix of a DLSS model of the network
D_e	Feedforward matrix of a DLSS model of the network

Table B.1: Notations for the graph G and DLSS models

Notation	Explanation
G_e	Extended graph
e_e	Set of N_e nodes of extended graph G_e
L_e	Set of L_e links of extended graph G_e
N_e	Number of nodes in extended graph G_e
L_e	Number of links in extended graph G_e
W_e	Adjacency matrix of extended graph G_e
Λ	Incidence matrix of extended graph G_e
Γ	Transposed incidence matrix Λ with all negative entries set to 0
Γ_w	Internal sub-matrix of Γ
Γ_ϕ	Input sub-matrix of Γ
Γ_ψ	Output sub-matrix of Γ
Γ_z	External sub-matrix of Γ
Φ	Matrix that defines the input links existence
Ψ	Matrix that defines the output links existence
Z	Matrix that defines the external links existence
F	Extension of the matrix Γ for higher-dimensional vectors in G_e
F_w	Internal topology matrix, defined upon Γ_w
F_ϕ	Input topology matrix, defined upon Γ_ϕ
F_ψ	Output topology matrix, defined upon Γ_ψ
F_z	External topology matrix, defined upon Γ_z

Table B.2: Notations for the extended graph G_e

Notation	Explanation
$f[k]$	flow of a pipe
$p[k]$	pressure of a junction
$d[k]$	water demand
$h[k]$	hydraulic head
$f[k]$	traffic flow [Veh/5min]
$v[k]$	traffic speed[mph]

Table B.3: Notations for the water network and road network

C

APPENDIX C

```
1  clc
2  clear
3  close all
4
5  %% Create a networked system
6  % We use below a toy example, of the DLSS systems, forming a path network
7  G{2,1}.N = 2;
8  G{2,1}.r = 1;
9  G{2,1}.q = 1;
10 G{2,1}.n = [3;3];
11 G{2,1}.m = [2;2];
12 G{2,1}.p = [2;2];
13 G{2,1}.mu = 2;
14 G{2,1}.rho = 2;
15 G{2,1}.W = [0 1; 0 0];
16 G{2,1}.Phi = [1 0];
17 G{2,1}.Ksi = [0; 1];
18 G{2,1}.Z = zeros(G{2,1}.r,G{2,1}.q);
19
20
21 for counter = 1 : G{2,1}.N
22     Model = drss(G{2,1}.n(counter),G{2,1}.p(counter),G{2,1}.m(counter));
23     G{1,counter}.A = Model.A;
24     G{1,counter}.B = Model.B;
25     G{1,counter}.C = Model.C;
26     G{1,counter}.D = Model.D;
27 end
28
29 % Compute the DLSS of the entire network
30 [A_e,B_e,C_e,D_e] = compute_scaled_DLSS(G)
31
32
```

```

33 %% Used functions
34 function [A_e,B_e,C_e,D_e] = compute_scaled_DLSS(G)
35 [G_w,G_phi,G_ksi,G_z] = compute_G_matrices(G);
36 [F_w,F_phi,F_ksi,F_z] = compute_F_matrices(G,G_w,G_phi,G_ksi,G_z);
37 [A_d,B_d,C_d,D_d] = compose_diagonal_ABCD(G);
38 [A_e,B_e,C_e,D_e] = compute_DLSS(A_d,B_d,C_d,D_d,F_w,F_phi,F_ksi,F_z);
39 end
40
41 function [G_w,G_phi,G_ksi,G_z] = compute_G_matrices(G)
42 N = G{2,1}.N;
43 r = G{2,1}.r;
44 q = G{2,1}.q;
45
46 W = G{2,1}.W;
47 Phi = G{2,1}.Phi;
48 Ksi = G{2,1}.Ksi;
49 Z = G{2,1}.Z;
50
51 W_e = [zeros(r) Phi Z;...
52        zeros(N,r) W Ksi;...
53        zeros(q,r + N + q)];
54
55 % Define number of internal, input, output and external links
56 L_w = nnz(W);
57 L_phi = nnz(Phi);
58 L_ksi = nnz(Ksi);
59 L_z = nnz(Z);
60
61 % Construct modified incidence matrix L
62 L = zeros(r + N + q, L_w + L_phi + L_ksi + L_z);
63 ind_count = 1;
64 for counter_f_1 = 1 : r + N + q
65     for counter_f_2 = 1 : r + N + q
66         if(W_e(counter_f_2,counter_f_1))
67             L(counter_f_1,ind_count) = -1;
68             L(counter_f_2,ind_count) = 1;
69             ind_count = ind_count + 1;
70         end
71     end
72 end
73 Gamma = 0.5*(L' + abs(L'));
74 % Define Gamma matrices
75 G_w = Gamma(1 : L_w + L_phi, r + 1 : r + N);
76 G_phi = Gamma(1 : L_w + L_phi, 1 : r);
77 G_ksi = Gamma(L_w + L_phi + 1 : end, r + 1 : r + N);
78 G_z = Gamma(L_w + L_phi + 1 : end, 1 : r);
79 end
80
81 function [F_w,F_phi,F_ksi,F_z] = compute_F_matrices(G,G_w,G_phi,G_ksi,G_z)

```



```

82 % Store the parameters
83 n = G{2,1}.n;
84 p = G{2,1}.p;
85 m = G{2,1}.m;
86 mu = G{2,1}.mu;
87 rho = G{2,1}.rho;
88 N = length(n);
89 r = length(mu);
90 % q = length(rho);
91
92 L_w = nnz(G{2,1}.W);
93 L_phi = nnz(G{2,1}.Phi);
94 L_ksi = nnz(G{2,1}.Ksi);
95 L_z = nnz(G{2,1}.Z);
96
97 S_temp = [G_phi G_w; G_z G_ksi]*[mu zeros(r,1);zeros(N,1) p];
98
99 s_w = S_temp(1:L_w+L_phi,2);
100 s_phi = S_temp(1:L_w+L_phi,1);
101 s_ksi = S_temp(L_w+L_phi+1:L_w+L_phi+L_ksi+L_z,2);
102 s_z = S_temp(L_w+L_phi+1:L_w+L_phi+L_ksi+L_z,1);
103
104 s_w_phi = s_w + s_phi;
105 s_ksi_z = s_ksi + s_z;
106
107 % Initialise the matrices
108 F_w = zeros(sum(m) , sum(p)); % F_w
109 F_phi = zeros(sum(m) , sum(mu)); % F_phi
110 F_ksi = zeros(sum(rho) , sum(p)); % F_ksi
111 F_z = zeros(sum(rho) , sum(mu)); % F_z
112
113 % Construct the matrices F_w, F_phi, F_ksi and F_z, as in relation (4.14)
114 for counter_f_1 = 1 : L_w + L_phi
115     for counter_f_2 = 1 : N
116         if(G_w(counter_f_1,counter_f_2) == 1)
117             F_w(sum(s_w_phi(1 : counter_f_1 - 1)) + ...
118                 1: sum(s_w_phi(1 : counter_f_1)) , sum(p(1 : counter_f_2 - 1)))...
119                 + 1: sum(p(1 : counter_f_2))) = eye(p(counter_f_2));
120         end
121     end
122     for counter_f_3 = 1 : r
123         if(G_phi(counter_f_1,counter_f_3) == 1)
124             F_phi(sum(s_w_phi(1 : counter_f_1 - 1)) + ...
125                 1: sum(s_w_phi(1 : counter_f_1)) , sum(mu(1 : counter_f_3 - 1)))...
126                 + 1: sum(mu(1 : counter_f_3))) = eye(mu(counter_f_3));
127         end
128     end
129 end
130

```

```

131 for counter_f_1 = 1 : L_ksi + L_z
132     for counter_f_2 = 1 : N
133         if(G_ksi(counter_f_1,counter_f_2) == 1)
134             F_ksi(sum(s_ksi_z(1 : counter_f_1 - 1)) + ...
135                 1: sum(s_ksi_z(1 : counter_f_1)), sum(p(1 : counter_f_2 - 1))...
136                 + 1: sum(p(1 : counter_f_2))) = eye(p(counter_f_2));
137         end
138     end
139     for counter_f_3 = 1 : r
140         if(G_z(counter_f_1,counter_f_3) == 1)
141             F_z(sum(s_ksi_z(1 : counter_f_1 - 1)) + ...
142                 1: sum(s_ksi_z(1 : counter_f_1)), sum(mu(1 : counter_f_3 - 1))...
143                 + 1: sum(mu(1 : counter_f_3))) = eye(mu(counter_f_3));
144         end
145     end
146 end
147
148 end
149
150 function [A_d,B_d,C_d,D_d] = compose_diagonal_ABCD(G)
151 n = G{2,1}.n;
152 m = G{2,1}.m;
153 p = G{2,1}.p;
154
155 % Initialise block diagonal matrices A_d, B_d, C_d and D_d
156 A_d = zeros(sum(n));
157 B_d = zeros(sum(n),sum(m));
158 C_d = zeros(sum(p),sum(n));
159 D_d = zeros(sum(p),sum(m));
160
161 % Apply relation (4.10)
162 for counter_1 = 1 : length(n)
163     A_d((sum(n(1 : counter_1 - 1)) + 1) : sum(n(1:counter_1)),...
164         (sum(n(1 : counter_1 - 1)) + 1) : sum(n(1:counter_1))) = G{1,counter_1}.A;
165     B_d((sum(n(1 : counter_1 - 1)) + 1) : sum(n(1:counter_1))...
166         ,sum(m(1 : counter_1 - 1)) + 1 : sum(m(1:counter_1))) = G{1,counter_1}.B;
167     C_d((sum(p(1 : counter_1 - 1)) + 1) : sum(p(1:counter_1)),...
168         (sum(n(1 : counter_1 - 1)) + 1) : sum(n(1:counter_1))) = G{1,counter_1}.C;
169     D_d(sum(p(1 : counter_1 - 1)) + 1 : sum(p(1:counter_1)),...
170         (sum(m(1 : counter_1 - 1)) + 1) : sum(m(1:counter_1))) = G{1,counter_1}.D;
171 end
172 end
173
174 function [A_e,B_e,C_e,D_e] = compute_DLSS(A_d,B_d,C_d,D_d,F_w,F_phi,F_ksi,F_z)
175 p = size(D_d,1);
176 % Compute the DLSS matrices of the entire network, as in relation (5.1)
177 Inv_M = (eye(p) - D_d*F_w)^-1;
178 A_e = B_d*F_w* Inv_M*C_d + A_d;
179 B_e = B_d*F_w* Inv_M*D_d*F_phi + B_d*F_phi;

```

```
180 C_e = F_ksi*Inv_M*C_d;  
181 D_e = F_ksi*Inv_M*D_d*F_phi + F_z;  
182 end
```