2018-05-01

Signal Structure for a Class of Nonlinear Dynamic Systems

Meilan Jin
Brigham Young University

Signal Structure for a Class of Nonlinear Dynamic Systems

Meilan Jin

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Sean Warnick, Chair
Jacob Crandall
David Wingate

Department of Computer Science

Brigham Young University

ABSTRACT

Signal Structure for a Class of Nonlinear Dynamic Systems

Meilan Jin
Department of Computer Science, BYU
Master of Science

Signal structure is a partial structure representation for dynamic systems. It characterizes the causal relationship between manifest variables and is depicted in a weighted graph, where the weights are dynamic operators. Earlier work has defined signal structure for linear time-invariant systems through dynamical structure function. This thesis focuses on the search for the signal structure of nonlinear systems and proves that the signal structure reduces to the linear definition when the systems are linear. Specifically, this work:

1. Defines the complete computational structure for nonlinear systems.
2. Provide a process to find the complete computational structure given a state space model.
3. Define the signal structure for dynamic systems in general.
4. Provide a process to find the signal structure for a class of dynamic systems from their complete computational structure.

Keywords: signal structure, complete computational structure, dynamical structure functions, nonlinear dynamic systems, linear time-invariant systems, partial structure representations, structure representations

ACKNOWLEDGMENTS

I would like to thank my mother and my husband for their continuous support. I would also like to give a special thank you to my adviser, Dr. Sean Warnick, who has given me tremendous help and guidance on this project. Finally, I want to thank my committee and my colleagues in the IDeA Labs for offering feedback and insights about my thesis.

Table of Contents

List of Figures

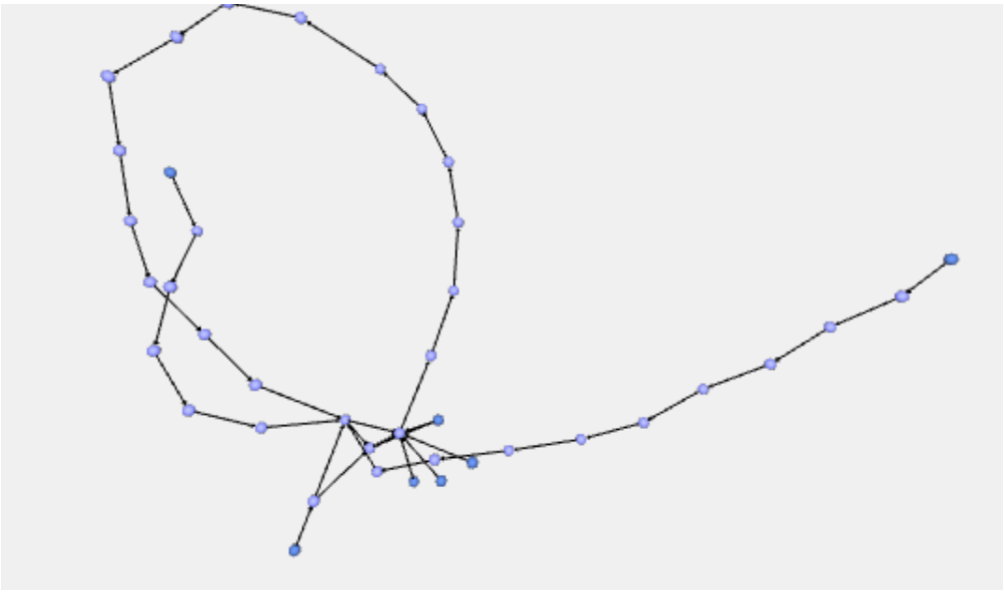
Chapter 1

Introduction

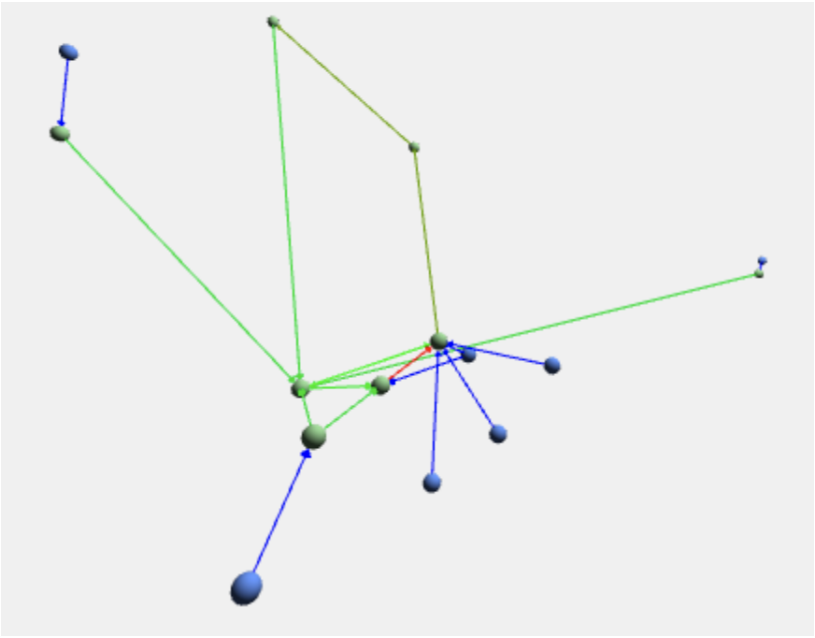
Studies of network structure and behavior play an important role in understanding dynamic systems and vulnerability analysis. Consider a prototype application of the Sevier River system in Utah; it is the primary water supply that irrigates 286,600 acres of farmlands [1]. Sensors are placed at various locations along the river to measure the water flow. The river commissioner controls the gates at reservoirs that adjust the amount of water leading into the downstream river systems according to readings of water flow data he gets every day on his electronic device. One problem for such a cyber-physical system is that this layer of computers/sensors introduces additional vulnerabilities. Cyber attackers could modify the readings of water flow data and cause more water to be let out of the reservoirs, such that not enough water is stored by July for irrigation. The attack could lead to millions of dollars lost. However, this problem can be detected and solved through the analysis on the signal structure of the river system. The signal structure is a novel notion of system structure developed recently to study the causal relationships among a system's manifest variables. We will use the Sevier River example to introduce a system's signal structure.

Water flow out to the branching canals along the main Sevier River and the flow is controlled by the gate at each canal. Each canal is manged by an individual company, which in turn services requests made by farmers needing irrigation water downstream. A sensor is placed at each canal gate to measure the water flow. The River commissioner receives aggregated request through his electronic device and he then schedules releases from dams to meet the demand represented by these requests. With these details, we can first

model the Sevier River system and represent it with a linear state-space model with states $\{x_1,\ x_2,\ \ldots\ ,\ x_n\} \in \mathbb{R}^n$, inputs $\{u_1,\ u_2, \ldots, u_m\} \in \mathbb{R}^m$, and outputs $\{y_1,\ y_2,\ \ldots,\ y_p\} \in \mathbb{R}^p$. The states are all the water flow in cubic feet per seconds(cfs) along the Sevier River. The outputs are the positive water flow in cfs out of the mains river stream to the branching canals. The controlled inputs includes the water flow into the Sevier River from the upstream reservoir chosen by the river commissioner. More details about how the river system is modeled can also be found in [1] and [2]. Then, exposed states are identified. In an attacker's view, the exposed states can be, for example, the measured water flow at each canal gate, which are the outputs in the river's state space model. These measured water flow data are published on the internet for the public. After identifying the exposed variables, an attack surface can be constructed with a signal structure, see in Figure 1.1b. A signal structure is a graph where nodes represent the manifest variables and edges characterize the causal dependency relationships between these nodes. Signal structure considers only the causal dependency relationships between manifest variables. Since attacker's have access only to exposed variables, the signal structure will precisely characterize the attack surface in the attackers perspective and reveal the dynamic relationships among these variables. After we construct the attacker's view of the river through the signal structure, vulnerable links can be identified and analyzed. Then we can design possible attacks and create attack mitigation plans to secure the system. The same technique can be applied to many other cyber-physical systems, including, air traffic controls, power systems, municipal water systems, chemical plants and many more.



(a) The complete computational structure for the Gunnison section of the Sevier River system. The nodes are inputs and states of the river system, and a directed edge represents direct dependency between two nodes. A node that the directed edge points to is said to directly depend on the node a edge starts.



(b) The signal structure for the Gunnison section of the Sevier River system. Blue nodes represents inputs to the system, and green nodes are the system states. Blue links from a input to a state is always secure and cannot be perturbed. The red links represents links that are most vulnerable within this system and can be manipulated by attackers, while green links are secure.

Figure 1.1: The complete computational structure and the attack surface for the Gunnison section of the Sevier River.

The signal structure is a powerful tool for studying the "manifest behavior" of a system. Vulnerability analysis is only one of its many applications. The signal structure has also been widely used in network reconstruction for financial, social, chemical and biological networks. However great the signal structure may be, it is defined only for linear time-invariant (LTI) systems. The purpose of this thesis is to define the signal structure for a class of arbitrary nonlinear system and provide a process to find the signal structure given the state space equations of that system.

1.1 Problem Statement

This thesis will extend the existing definition of signal structure given for linear time-invariant systems to a class of nonlinear systems, and provide a process for finding it given a state space model of the form:

$$\begin{aligned} \dot{x} &= f(x(t), u(t)), \\ y(t) &= \begin{bmatrix} I & 0 \end{bmatrix} x(t), \end{aligned} \tag{1.1}$$

where state $x \in \mathbb{R}^n$, input $u \in \mathbb{R}^m$, and output $y \in \mathbb{R}^p$, with $p < n$. When $t \in \mathbb{R}$, $\dot{x} = \dfrac{dx}{dt}$ for continuous time systems, or $t \in \mathbb{Z}$ and $\dot{x} = x[k+1]$ for discrete time systems. We follow the same notation style throughout this thesis.

Much of this thesis is devoted to the definition of signal structure and the process for constructing a signal structure given a nonlinear state space model in the form of (1.1). The construction of nonlinear signal structure builds on the linear signal structure and another type of structure representation for dynamic systems, the complete computational structure. The linear signal structure is characterized by the dynamic structure function $(Q(s), P(s))$. A detailed derivation and some related property of dynamic structure function (DSF) is included in Chapter 3. The complete computational structure defined in Chapter 4 is a core step for constructing the signal structure. The last and most important part of this thesis is included in Chapter 5, which provides a definition for signal structure in general and a

process to find it for a class of nonlinear systems. The main contributions of this thesis are as follows:

1. A precise definition of the complete computational structure.
2. A process for finding the complete computational structure of an arbitrary nonlinear system.
3. A precise definition of the signal structure.
4. A process for finding the signal structure given the state space model of a class of nonlinear dynamic systems.

Because both the definition and process for finding the signal structure depend on the complete computational structure, it is critical to understand the relationship between the state space model, the complete computational structure, and the signal structure. To illustrate their relationship, we will continue with the Sevier River example. A state space model describes the time evolution of states for a system with specific rules of how these states change. It contains the most comprehensive information about a dynamic system. The system views in Figure 1.1 are built from the state space model of the Gunnison section of the Sevier River system. The complete computational structure is in one-to-one correspondence with the state space model. It is a complete view of the system, which characterizes mainly the causal dependency relationships among all the system inputs, states, and outputs. Figure 1.1a is the complete computational structure for Gunnison. This structure shows the dependency relationships of all the variables in the state space model and is equivalent to having all the states measured/exposed. An outsider, however, had access to only part of the systems information and can measure only a subset of states. Figure 1.1b shows what an outsider's view of the same system looks like - only a partial view of the Gunnison section, a view that is created by selecting the variables that are considered measured/exposed from the complete computational structure, which are then used to find the signal structure. The

signal structure only shows the causal dependency of inputs to manifest states, and between two manifest states.

Chapter 2

Related Works

Studying system structure is important because the structure of a system can directly impact its behavior. Sometimes different structures can yield the exactly same behavior. A considerable amount of literature has been written about the representations of system structures and their properties. [3], [4] and [5] provide a framework for understanding different ways to represent system structures, specifically in linear systems. My work extends the idea of the complete computational structure and signal structure presented in these works to nonlinear systems.

The same system can be represented by different types of structures. However, each structural representation contains different level of information about the system. The complete computational structure discussed in this work contains the most structural information about a system, detailing in the internal computations and shows the dependency relationships of all the variables in a system. On the other hand, black box models, often in the form of transfer functions, describe only the manifest behavior and not the internal computations. The *manifest structure* associated with the black box model is a graph with nodes representing inputs and outputs of the system and edges showing the mapping from inputs to outputs. Black box modeling is most useful in fitting data but contains the least amount of structural information.

The signal structure is a partial structure representation. It shows the *causal* dependencies between manifest variables without affecting other manifest variables. Another most commonly used partial structure representation is the subsystem structure [4]. Interconnect-

ing systems together through variable sharing and forcing these variables to take on the same values creates a new composed system. Each subsystem is represented by its transfer function. The internal structure of each subsystem is unknown. The subsystem structure is a graph that describes the interconnection of subsystems in a composite system. [6] provides a methodology that models interconnected subsystems with tearing, zooming and linking. First, tearing a black box model into several mini black box models, where these mini models can be modeled by first principle physical laws. Then, zooming into each subsystem and model individually. Finally, all subsystems are linked by setting up interconnection constraints.

The subsystem structure is a great tool for analyzing dynamic systems, especially for large complex systems. For many physical systems, the structure can be understood as how components are interconnected together. In this case, subsystem structure is great for representing these engineered systems. However, for fluidic, biological and chemical systems, components are hard to be compartmentalized and described as interconnected components. The signal structure is particularly useful for such systems. The signal structure is also used in the design of an LTI controller over a network with a prespecified topology [7]. As mentioned in the introduction, the signal structure is useful in the vulnerability analysis of cyber-physical systems [8], [9]. More studies about meanings of structure in dynamic systems are published in [10], [11].

This work focuses on the signal structure for nonlinear systems, but a lot of the linear results are very helpful for analysis and verification in the nonlinear cases in the later chapters. The signal structure for a linear invariant system is represented by the system's dynamical structure function [12]. Following this remark ,[13], [14], [15], [16] discuss conditions when dynamical reconstruction is possible and applications of network reconstruction from data.

Besides modeling from first principle and computer simulations, system theorists often draw from the studies of graphical models when studying structures for *complex networks*. In [17], the authors exemplify how graphical models help solve control and identification problems. Neural networks are also used in the identification of nonlinear discrete dynamic

systems in [18]. A survey on major concepts and results in the study of modeling structure and dynamics of a complex network as graphs is included in [19]. These works pose a learning problem to find the underlying model. Different from these works, we are not learning the structure from data. Rather, my work starts with a given state space model, which could be modeled from first principles and computer simulations. Then the state space model is utilized to construct the complete computational structure and the signal structure.

Unlike lots of existing research that assumes a static relationship between the time series data, the signal structure characterizes the dependency relationships of manifest variables with a dynamic operator for both continuous and discrete time. It shows the inter-relatedness of data that are dynamically related. The linear dynamical graph is another representation of stochastic processes that are interconnected through dynamic relations. Inspired by random variables interconnected through static relations and d-separation, [20] carries over the technique to linear dynamical graphs. The concept of linear dynamical graphs are used in multiple studies for reconstructing networks in [21], [22], [23] and [24]. Essentially, the linear dynamic graph and the dynamic structure function are the same except that they take different form of inputs. The linear dynamic graph (LDG) is defined to be a pair of $(H(z), e)$, where $H(z)$ is a $n \times n$ matrix of transfer functions, and e is a vector of n rationally related random processes. The output process for LDG is $x(t) = H(z)x(t) + e$, while DSF is defined to be $Y = QY + PU$, where Q and P are matrices of transfer functions and U are the inputs to the dynamic system.

There are many other representations of networks of stochastic processes. In [25], a directed information graph is proposed to represent networks of stochastic processes. The directed information graph is proven to be equivalent to generative model graphs under mild assumptions and can be used for causal inferences and also be generalized to nonlinear settings. To further improve on identifying the nonlinear causal relationship with the presence of hidden nodes, [26] introduced an information-theoretic criteria in general model of stochastic dynamical systems with no restriction on the mapping function or the underlying

structure. The causal structure graph introduced in [26] is developed based on the functional dependencies defined in [27], where it states that v_j causes v_i if v_i is a function of v_j. In Chapter 4, we also adopt the same concept for the definition of dependency. Identifying dependencies contained in the state space model is a very important step towards constructing the signal structure in this work.

Chapter 3

Background

In this section, we will show the derivation of the dynamic structure function (DSF), which is essential when validating the process for constructing the complete computational structure and the signal structure for nonlinear systems in Chapters 4 and 5. For linear time-invariant (LTI) systems, the signal structure is represented by the dynamical structure function.

In [8], [15], the authors showed the derivation of dynamical structure functions of LTI systems from its state space representation. Given a state space model of an LTI system,

$$\begin{aligned} \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} &= \begin{bmatrix} \bar{A_{11}} & \bar{A_{12}} \\ \bar{A_{21}} & \bar{A_{22}} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} \bar{B}_1 \\ \bar{B}_2 \end{bmatrix} u, \\ y &= \begin{bmatrix} \bar{C}_1 & \bar{C}_2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \end{aligned} \tag{3.1}$$

where $\begin{bmatrix} \bar{C}_1 & \bar{C}_2 \end{bmatrix}$ is full row rank. We can find a transformation T such that the system can be transformed into

$$\begin{aligned} \begin{bmatrix} \dot{y} \\ \dot{x} \end{bmatrix} &= \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u, \\ y &= \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix}, \end{aligned} \tag{3.2}$$

where vector y represents states that are measured, and vector x represents states that are hidden. When we take the Laplace transform of x, y, and u in terms of t, assuming zero initial condition, we get

$$\begin{bmatrix} sY \\ sX \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} Y \\ X \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} U. \tag{3.3}$$

Solving for X in the second equation in of (3.3), we have

$$X = (sI - A_{22})^{-1} A_{21} Y + (sI - A_{22})^{-1} B_2 U, \tag{3.4}$$

where $(sI - A_{22})^{-1}$ is always invertible. Substituting (3.4) into the first equation of (3.3),

$$\begin{aligned} sY &= WY + VU, \\ W &= A_{11} + A_{12}(sI - A_{22})^{-1} A_{21}, \\ V &= A_{12}(sI - A_{22})^{-1} B_2 + B_1. \end{aligned} \tag{3.5}$$

When we let D be an diagonal matrix, where its diagonal entry is the diagonal term of matrix W,

$$(sI - D)Y = (W - D)Y + VU. \tag{3.6}$$

Rewriting this equation and we get,

$$\begin{aligned} Y &= QY + PU, \\ Q &= (sI - D)^{-1}(W - D), \\ P &= (sI - D)^{-1} V. \end{aligned} \tag{3.7}$$

Through the derivation process of DSF, we suppress the hidden variables and arrive with matrix Q, which preserves only the causal dependency among manifest variables and P,

which captures relationship among inputs and outputs. Together, the pair $(Q(s), P(s))$ is called the dynamical structure function.

Definition 1 (Dynamical Structure Function)**.** *Given a state space model of the form* (3.1)*, we define the dynamical structure function of the* (3.1) *to be a* $(Q(s), P(s))$ *pair, where transfer function matrices* $Q(s)$ *and* $P(s)$ *are the internal structure and control structure, respectively, and is given as in* (3.7)*.*

Both Q and P are transfer function matrices. Entry Q_{ij} in Q characterizes how signal (measured states) Y_i is causally dependent on Y_j, where $i \neq j$. Note that Q is zero on the diagonal and either zero or a strictly proper transfer function on the off-diagonal. The fact that Q is hollow reflects that self-loops are eliminated from the dependency relationships. P is a matrix with entries of zeros or strictly proper transfer functions, where each entry describes how inputs are associated with outputs without depending on any additional measured states. The Boolean structure of $(Q(s), P(s))$ characterizes the *signal structure* graph for an LTI system. The entries in Q and P are corresponding weights on each link in the signal structure.

Earlier work has proven that there is only one pair of Q and P associated with a given linear state space model. On the other hand, given Q and P, we can find many state space realizations of the system. Every DFS specifies a unique transfer function, but many DFS architectures can represent the same transfer function. Given P and Q, the transfer function of a system is given by,

$$G = C(sI - A)^{-1}B = (I - Q)^{-1}P.$$

This result is particularly useful in finding a Q and P pair that secures the system from attacks that aim to destabilize it.

The following is an example of deriving DSF from a state space form of an LTI system. Consider a state space model:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 & 1 & 0 \\ 0 & 0 & -3 & 0 & 0 & 1 \\ 0 & 0 & 1 & -4 & 0 & 0 \\ 1 & 0 & 0 & 0 & -5 & 0 \\ 0 & 1 & 0 & 0 & 0 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \tag{3.8}$$

After following the above process, we can compute a $P(s)$ and $Q(s)$ for this dynamical system.

$$Q(s) = \begin{bmatrix} 0 & 0 & \frac{1}{(s+1)(s+4)} \\ \frac{1}{(s+2)(s+5)} & 0 & 0 \\ 0 & \frac{1}{(s+3)(s+6)} & 0 \end{bmatrix} \quad P(s) = \begin{bmatrix} \frac{1}{(s+1)(s+4)} & 0 & 0 \\ 0 & \frac{1}{(s+2)(s+5)} & 0 \\ 0 & 0 & \frac{1}{(s+3)(s+6)} \end{bmatrix} \tag{3.9}$$

Figure 3.1 shows the corresponding unique signal for the above system. The directed graph shows the dependency of states to states and input to states that are captured by the $Q(s)$ and $P(s)$ matrices. The non-zero entries in P and Q represent a link in the graph. The link from y_3 to y_1 is the Q_{13}th entry in the $Q(s)$ matrix.

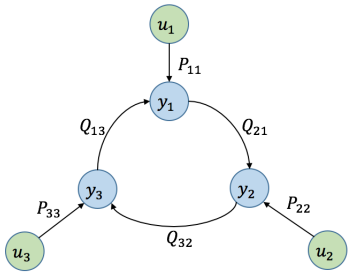
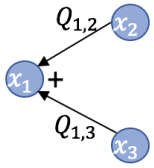
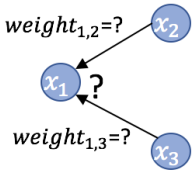


Figure 3.1: Signal structure for system given in (3.8). Green nodes are all the inputs and blue nodes are measured states. Non-zero entries in $P(s)$ indicates dependency of state on inputs without going through other measured states. Non-zero entries in $Q(s)$ indicated dependency of measured states on other measured states without going through other measured states.

In cases where a state y_i depends on multiple other measured states, the node representing y_i in the graphical representation of its signal structure will have multiple links coming in from the nodes it depends on (the parent nodes). For linear cases, this can be easily interpreted as adding all the weights from node it depends on. However, for nonlinear systems, how to interpret multiple dependencies remains a difficult problem. For example, consider $\dot{x_1} = x_2 x_3$. Assume that we know x_1 depends on x_2 and x_3, without a clear derivation of nonlinear signal structure, we cannot interpret how the weights for link (x_2, x_1) and (x_3, x_1) are combined and affects x_1, see in Figure 3.2. The derivation and calculation for weights in nonlinear system in general are non trivial. In this thesis, we will focus on precisely defining the complete computational structure and the signal structure for nonlinear systems and provide a process for finding the graph representations of the complete computational structure and the signal structure. The calculation and interpretation of weights for signal structure will be included in future works.



(a) The signal structure for $\dot{x_1} = x_2 + x_3$. x_1 depends on x_2 and x_3. The weights $Q_{1,2}$ and $Q_{1,3}$ are summed up.



(b) The graphical representation of the signal structure for $\dot{x_1} = x_2 x_3$. The interpretation of how the two weights on link (x_2, x_1) and (x_2, x_1) are combined is unknown.

Figure 3.2: Hyper links in the signal structures.

Chapter 4

Complete Computational Structure

4.1 Definition of the Complete Computational Structure

In order to define the complete computational structure for dynamic systems, we need a description of dependency of a function on its arguments. A notion of dependency defined in [26], [27]:

Definition 2 (Dependency)**.** *A function f(x), which maps an m-dimensional domain to a n dimensional co-domain, is said to depend on its i^{th} variable x_i, if there exist values of the other $m-1$ variables x_j, where $i \neq j$, such that f(x) is not constant over all values of x_i while holding the other variables fixed. If m=1, then f(x) depends on x if it is not constant over all values of x.*

Consider state space systems of the form:

$$\begin{aligned} \dot{x} &= f(x(t), u(t)) \\ y(t) &= g(x(t)), \end{aligned} \tag{4.1}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $y \in \mathbb{R}^p$, with $p < n$ and $g(x(t)) = \begin{bmatrix} I & 0 \end{bmatrix} x(t)$. State x_i, $0 < i \leqslant n$, depends on x_j, $i \neq j$, if x_i is not constant over all values of x_j while holding the other n-1 variables fixed. For example, state x_1 in $\dot{x_1} = x_1(t) + x_2(t)x_3(t) + u_1(t)$, depends on state x_2 when $x_3 \neq 0$. Based on this notion of dependency, the complete computational structure for (3.1) is defined below.

Definition 3 (Complete Computational Structure)**.** *The* complete computational structure *for system 4.1 is a weighted directed graph* $\mathscr{C} = \{V, E\}$*, where* V *is a set of vertices* $V(\mathscr{C})$ *with each vertex representing a variable in system 4.1, and* E *is a set of edges* $E(\mathscr{C})$*. There are m+n+p vertices corresponding to the inputs* u_i*, states* x_j *and outputs* y_k*. A directed edge exists from node i to node j if node j depends on node i, where* $(i, j) \in E$*. The weight on edge* (i, j) *is a dynamic operator that describes how node j depends on i.*

4.2 Process for Finding the Nonlinear Complete Computational Structure

Based on the notion of dependency in Definition 2, and complete computational structure in Definition 3, an edge exists from node j to node i if and only if i depends on j. Here, we use three matrices to represent a system's complete computational structure: a state adjacency matrix $\mathscr{X}$, a input-to-state adjacency matrix $\mathscr{U}$, and a state-to-output adjacency matrix $\mathscr{Y}$. Matrix $\mathscr{X}$ is $n \times n$ in size and describes the dependency relationship of one state to another state. $\mathscr{U}$, and $\mathscr{Y}$ are $n \times m$ and $p \times n$ matrices that represent dependency relationships of a system's inputs to states, and the states to outputs. With each matrix, the ij_{th} entry is 1 if node i depends on node j, otherwise, ij_{th} entry is 0. In the corresponding graph, there is an edge from node j to node i if the ij_{th} entry is 1, and no edge exists from j to i if the ij_{th} entry is 0. For example, if the i_{th} state depends on the j_{th} state, then $\mathscr{X}_{ij} = 1$. Since x_1, x_2, ..., x_p are identically equal to the measured outputs y, we call these first p states "measured states" and the remaining $n - p$ states "hidden states".

Given a state space model of the form (4.1), we can find the complete computational structure of the system by:

1. Instantiating all nodes for each element of X, U, and Y, where X is a set of states, U is a set of inputs, and Y is a set of outputs.
2. Constructing a state adjacency matrix $\mathscr{X}$, where entry $\mathscr{X}_{ij} = 1$ if state i depends on state j and $i \neq j$; otherwise $\mathscr{X}_{ij} = 0$.

3. Constructing the input-to-state adjacency matrix $\mathscr{U}$, where entry $\mathscr{U}_{ij} = 1$ when state i depends on input j; otherwise $\mathscr{U}_{ij} = 0$.

4. Constructing the state-to-output adjacency matrix $\mathscr{Y}$ and $\mathscr{Y} = \begin{bmatrix} I & 0 \end{bmatrix}$. $\mathscr{Y}$ is a p x n matrix indicating the first p states are measured and last $n - p$ states are hidden.

5. Drawing directed edges based on the adjacency matrices constructed from Steps (2), (3), and (4). No edges are allowed from u_i to u_j or y_i to y_j.

4.3 Nonlinear Complete Computational Structure Example

Example 1.

$$\begin{aligned}
\dot{x_1} &= x_1(t)x_2(t) + x_3(t)x_5(t) + u_1(t) \\
\dot{x_2} &= x_3(t) + x_4(t) + x_2(t) + u_2(t)u_1(t) \\
\dot{x_3} &= x_1(t)x_2(t) + x_5(t) + x_3(t) + u_3(t) \\
\dot{x_4} &= x_3(t)x_4(t) + x_1(t) + u_4(t) \\
\dot{x_5} &= x_4(t)x_2(t) + x_3(t) + x_5(t) + u_5(t)
\end{aligned} \tag{4.2}$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \\ y_4(t) \end{bmatrix} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix}$$

Given the state space model in (4.2), first, instantiate all nodes from $X = \{x_1, x_2, x_3, x_4, x_5\}$, $U = \{u_1, u_2, u_3, u_4, u_5\}$, and $Y = \{y_1, y_2, y_3, y_4\}$. Then follow Steps 2, 3, 4 to construct the state, input-to-state and state-to-output matrices: $\mathscr{X}$, $\mathscr{U}$ and $\mathscr{Y}$, see in Equation (4.3). After drawing all the edges following the adjacency matrices, the resulting complete computational structure produced by the above procedure is shown in Figure 4.1. Note that we do not draw

self loops in the complete computational structure and $\mathscr{X}$ is always hollow.

$$\mathscr{X} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}, \mathscr{U} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathscr{Y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.3}$$

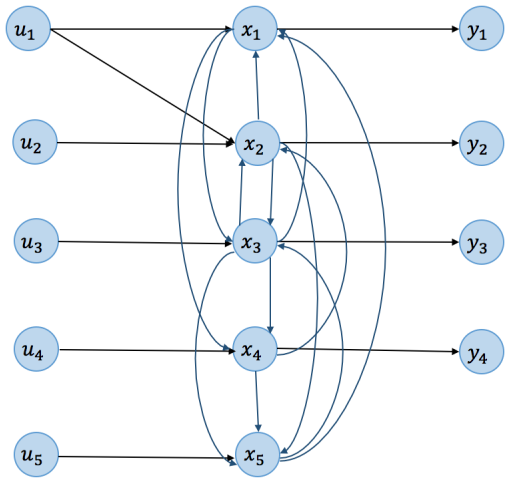


Figure 4.1: Complete computational structure for System (4.2).

4.4 Validation

In this section, we will validate that the process for constructing the complete computation structure in Section 4.2 is correct by applying this process to an LTI system and show that it produces the right complete computational structure.

Consider an LTI system, in the form of:

$$\begin{aligned} \dot{x} &= Ax(t) + Bu(t), \\ y(t) &= Cx(t), \end{aligned} \tag{4.4}$$

where vector $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $y \in \mathbb{R}^p$, with $p \leq n$.

Following the process for constructing the complete computational structure in Section 4.2, we need to first find the state adjacency matrix $\mathscr{X}$, input-to-state adjacency matrix $\mathscr{U}$, and state-to-output adjacency matrix $\mathscr{Y}$. From the definition of dependency, it is easy to see that for an LTI system, the Boolean structure of matrix A in Equation (4.4) contains the dependency relationships among all states. Because we do not draw any self-loops in the complete computational structure, the Boolean structure of $(A - diag(A))$ is the state adjacency matrix $\mathscr{X}$; $diag(A)$ denotes the diagonal of matrix A. Similarly, the Boolean structure of matrix B is the input-to-state adjacency matrix $\mathscr{U}$ and the Boolean structure of matrix C is the state-to-output adjacency matrix. Thus,

$$\mathscr{X} = Bool(A - diag(A)),$$
$$\mathscr{U} = Bool(B),$$
$$\mathscr{Y} = Bool(C).$$

$Bool()$ denotes the Boolean structure of a matrix. Then we can construct the complete computational structure graph from $\mathscr{X}$, $\mathscr{U}$, and $\mathscr{Y}$.

Another way to find the complete computational structure is to look at the signal structure where all states are measured. For LTI systems, we can find the complete computational structure using DSF with all states are measured and prove that the result is the same as the complete computational structure constructed by process 4.2. From the Background section, for an LTI system that:

1. The $(Q(s), P(s))$ pair is the DSF of an LTI system. $Q(s)$ characterizes how measured states depend on other measured states and $P(s)$ represents how measured states depends on the inputs;
2. The Boolean structure of Q and P constructs the signal structure graph and the entries in Q and P are weights for each link;

3. The signal structure describes the causal dependencies of manifest variables.

The complete computational structure of a state space model can be considered the same as the signal structure with no hidden states. Extending DSF, we can derive the complete computational structure for an LTI system.

Theorem 1. *Consider an LTI system in the form of* (4.4)*, its complete computational structure is characterized by* $(\hat{Q}, \hat{P}, \hat{C})$*, where*

$$\hat{Q} = (sI - diag(A))^{-1}(A - diag(A))$$
$$\hat{P} = (sI - diag(A))^{-1}B$$
$$\hat{C} = C.$$

Proof. According to the derivation of dynamical structure function in Chapter 3, $Q = (sI - D)^{-1}(W - D)$, and $W = A_{11} + A_{12}(sI - A_{22})^{-1}A_{21}$. Here, D is the diagonal entries of W. Suppose this LTI system has zero hidden states, then y = x, and $A_{11} = A, A_{12} = A_{22} = A_{21} = 0$. Therefore, $W = A_{11} = A$. Then $\hat{Q} = (sI - diag(A))^{-1}(A - diag(A))$. Similarly, when there are no hidden states, $V = A_{12}(sI - A_{22})^{-1}B_2 + B_1 = B1 = B$. Then $\hat{P} = P = (sI - D)^{-1}V = (sI - diag(A))^{-1}B$. Because $y(t) = \begin{bmatrix} I & 0 \end{bmatrix} x(t)$, it is trivial that $\hat{C} = C$. □

Transfer function matrix $\hat{Q}$ shows how states directly depends on other states and $\hat{P}$ represents how states causally depends on inputs. $\hat{C}$ maps states to outputs. We know the structure of a matrix remains the same when it is multiplied by a diagonal matrix on its left side. Since $(sI - diag(A))^{-1}$ is a diagonal matrix and $(A - diag(A))$ is hollow with zero on its diagonal, then $\hat{Q}$ remains to be hollow. Thus, A-diag(A) has the same Boolean structure as $\hat{Q}$. Similiarly, B has the same Boolean structure as $\hat{P}$. Therefore, $\mathscr{X} = Bool(\hat{Q})$, $\mathscr{U} = Bool(\hat{P})$, and $\mathscr{Y} = \hat{C}$. We have validated that process 4.2 produces the right complete computational structure.

Example 2. *Consider an LTI state space model:*

$$\dot{x} = \begin{bmatrix} -4 & 2 & -7 & 0 & 6 \\ 0 & -9 & 10 & 4 & 0 \\ 3 & 5 & -6 & 0 & 8 \\ 1 & 0 & 4 & -5 & 0 \\ 0 & 5 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{bmatrix} + \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & -8 & 0 \\ 0 & 0 & 0 & 0 & 7 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \\ 4_5(t) \end{bmatrix}$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \end{bmatrix}$$

When there are no hidden states, the computational structure is the same as the signal structure. Using DSF, we find:

$$\hat{Q} = \begin{bmatrix} 0 & \frac{2}{s+4} & \frac{-7}{s+4} & 0 & \frac{6}{s+4} \\ 0 & 0 & \frac{10}{s+9} & \frac{4}{s+9} & 0 \\ \frac{3}{s+6} & \frac{5}{s+6} & 0 & 0 & \frac{8}{s+6} \\ \frac{1}{s+5} & 0 & \frac{4}{s+5} & 0 & 0 \\ 0 & \frac{5}{s+1} & \frac{1}{s+1} & \frac{1}{s+1} & 0 \end{bmatrix}, \hat{P} = \begin{bmatrix} \frac{3}{s+4} & 0 & 0 & 0 & 0 \\ \frac{1}{s+9} & \frac{4}{s+9} & 0 & 0 & 0 \\ 0 & 0 & \frac{5}{s+6} & 0 & 0 \\ 0 & 0 & 0 & \frac{-8}{s+5} & 0 \\ 0 & 0 & 0 & 0 & \frac{7}{s+1} \end{bmatrix}$$

Observe that the Boolean structures of $\hat{P}$ and $\hat{Q}$ are the same as $A - diag(A)$ and $B - diag(B)$:

$$A - diag(A) = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}, B - diag(B) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Chapter 5

Signal Structure

5.1 Definition of the Signal Structure

The definition of complete computational structure allows us to define and find the signal structure of dynamic systems. Before defining signal structure, we need to understand the notion of path. In a directed graph, a path exists from node i to node j if there is a set of distinct and directed edges of the form $\{(i, n_1), (n_1, n_2), ..., (n_k, j)\}$. When there are multiple paths from node i to node j, the netpath is the sum of all paths from node i to node j.

Consider a system of the form 4.1 from Chapter 4 with p measured signals; its complete computational structure is given by $\mathscr{C} = \{V, E\}$. The signal structure can be defined as:

Definition 4 (Signal Structure)**.** *Consider a system of the form (4.1) with complete computational structure $\mathscr{C}$. It's* signal structure *is a weighted directed graph $\mathscr{S} = \{V, E\}$, where V is a set of vertices V($\mathscr{S}$), and E is a set of edges E($\mathscr{S}$). There are m+p nodes corresponding to the m inputs and p outputs. A directed edge exists from node i to node j if the netpath from node i to node j — without going through any measured state x_i — is nonzero. Weights on the edges are dynamic operators describing the causal dependencies of each node on the others.*

The idea of netpath (through the hidden states) is very important when constructing a signal structure for dynamic systems. A directed edge exists from node i to node j in the signal structure if there exists a path from node i to node j without going through any measured states in the complete computational structure. However, sometimes there can be multiple paths from node i to j in the complete computational structure that directly cancel each other, so that the net path from i to j is zero. In this case, there is no edge from i to j

in the signal structure, since j does not depend on i. On the other hand, partial cancellation of paths only changes the weights on the edge from node i to node j without causing the link to disappear.

5.2 Constructing a Signal Structure With No Exact Cancellation: The Graph Theoretic Approach

Assume there is no exact cancellation among paths. By Definition 4, we can find the signal structure of a dynamic system following the procedure below. Note, we limit our problem to only systems in the form of (4.1), where $y(t) = \begin{bmatrix} I & 0 \end{bmatrix} x(t)$.

1. Instantiate all nodes for each element in U and Y. U is a set of all the inputs and Y is a set of all the outputs in a new signal structure graph.
2. Mark all the measured states in its complete computational structure.
3. For each input node u_i in the complete computational structure, if a path exits from u_i to a measured state x_j without going through another measured state, (i can be the same as j), draw an edge from u_i to y_i in the signal structure graph.
4. For each measured state x_i in the complete computational structure, if a path exits from x_i to another measure state x_j without going through another measured state, $i \neq j$, draw an edge from y_i to y_j in the signal structure graph.

Note, when a path from a measured node j to another measured node i goes through one or more hidden nodes in the complete computational structure, we draw a link from node j to i in the signal structure as long as it does not go through any other measured nodes.

Example 3.

$$\dot{x_1} = x_1(t)x_2(t) + x_3(t)x_5(t) + u_1(t)$$

$$\dot{x_2} = x_3(t) + x_4(t) + x_2(t) + u_2(t)u_1(t)$$

$$\dot{x_3} = x_1(t)x_2(t) + x_5(t) + x_3(t) + u_3(t)$$

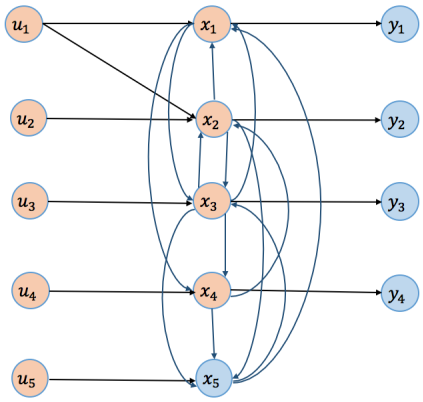
$$\dot{x_4} = x_3(t)x_4(t) + x_1(t) + u_4(t)$$

$$\dot{x_5} = x_4(t)x_2(t) + x_3(t) + x_5(t) + u_5(t)$$

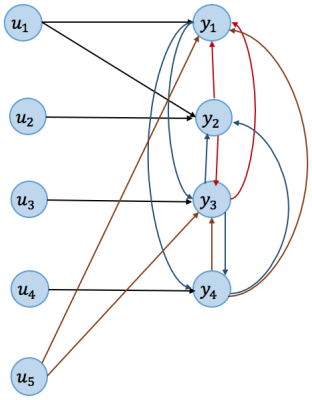
$$\begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \\ y_4(t) \end{bmatrix} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix}$$

Figure 5.1b is the signal structure of Example 3 constructed by following the procedure defined in 5.2. Given the complete computational structure of Example 3, we first identify the measured signals, $U = \{u_1, u_2, u_3, u_4, u_5\}$ and $Y = \{y_1, y_2.y_3, y_4\}$, and instantiate these nodes. Mark all the measured signals in the complete computational structure. All the measured signals for Example 3 have been colored as orange in Figure 5.1a. Check each input node in the complete computational structure according to Step 3. For example, there is a path from u_1 to x_1 and x_2, denoted as $\{(u_1, x_1)\}$ and $\{(u_1, x_2)\}$. As mentioned earlier, there is always a 1:1 relationship between a measured state x_i to its corresponding output y_i, so we draw a direct edge from u_1 to y_1, and u_1 to y_2 in the signal structure. When we check each input node, we find path $\{(u_2, x_2)\}, \{(u_3, x_3)\}, \{(u_4, x_4)\}$. There is also a path from $\{(u_5, x_5), (x_5, x_1)\}, \{(u_5, x_5), (x_5, x_2)\}$, which does not go through another measured state. Draw all the edge from inputs to outputs based on the valid paths we found. Note, a path is only valid when it does not go through any measured states. The last step is to identify the dependencies among the measured states. As illustrated in Step 4, find all valid paths from

each measured state to other measured states in the complete computational structure and then draw the corresponding edge in the signal structure.



(a) The complete computational structure with measured states highlighted as orange.



(b) The signal structure of Example 3. Red links are generated by paths that are from a measured node to another measured node through hidden nodes in the complete computational structure. For example, red link $\{u_5, y_1\}$ is generated by $\{(u_5, x_5), (x_5, x_1)\}$ in its corresponding complete computational structure.

Figure 5.1: The complete computational structure and signal structure for Example 3.

5.3 Validation

In order to validate the graphical approach for finding the signal structure for a dynamic system in process 5.2, we will prove that this process will produced the correct signal structure for LTI systems that have no cancellation of paths. Clearly, the graphical approach in process

5.2 will not detect exact path cancellation. We will discuss cases of cancellation in the next section.

There are two conditions for a link exists from node j to node i in a signal structure when constructing it from its corresponding complete computational structure using process 5.2:

1. Direct link: A link from node j to node i is called a direct link in the signal structure when it is generated by a direct link from a measured node to another measured node in the complete computational structure.
2. Indirect link: A link from node j to node i is called a indirect link in the signal structure when it is generated by a path from a measured node to another measured node through one or more hidden states in the complete computational structure.

Recall in the Background section, the signal structure for an LTI system is represented by $(Q(s), P(s))$:

$$Q = (sI - D)^{-1}(W - D),$$
$$W = A_{11} + A_{12}(sI - A_{22})^{-1}A_{21},$$
$$P = (sI - D)^{-1}V,$$
$$V = A_{12}(sI - A_{22})^{-1}B_2 + B1.$$

The structure of a matrix remains the same when it is multiplied by a diagonal matrix on its left side. Thus, Q has the same structure as W and P contains the same structure as V. Therefore, $Q_{ij} = 0$ if and only if $W_{ij} = 0$. Likewise for P and V. When we calculate the DSF, we first partition state matrix A to A_{11}, A_{12}, A_{21} and A_{22} based on measured and hidden states. A_{11} shows links from measured states to measured states; A_{12} shows links from hidden states to measured states; A_{21} shows links from measured states to hidden states and A_{22} shows links from hidden states to hidden states. From the definition of W, it is clear that all the direct links among measured states in a signal structure are contained in A_{11}, while all the indirect links are generated from $A_{12}(sI - A_{22})^{-1}A_{21}$. Likewise, all the direct links from

an input to an output are contained in B_1 and all indirect links are in $A_{12}(sI - A_{22})^{-1}B_2$. To prove process 5.2 and that the DSF will generate the same signal structure graph, we only need to prove that they will generate the same direct and indirect links of a signal structure.

Lemma 1. *DSF and the graphical approach find the same direct links in a signal structure.*

Proof. A direct link in the complete computational structure is always preserved in the signal structure constructed using the graphical approach in process 5.2. Because the direct links are all from A_{11} in DSF, it is easy to see that the direct links are also all preserved. From Section 4.4, for an LTI system, its state adjacency matrix is $A - diag(A)$. When we partition the state adjacency matrix to $\hat{A_{11}}, \hat{A_{12}}, \hat{A_{21}}$ and $\hat{A_{22}}$ base on measured and hidden states the same way as DSF, then $Bool(A_{11}) = Bool(\hat{A_{11}})$. The same logic applies to matrix B and the input-to-state adjacency matrix. Therefore, DSF and the graphical approach find the same direct links. □

Lemma 2. *DSF and the graphical approach find the same indirect links in a signal structure.*

Proof. Suppose A_{12}, A_{22} and A_{21} are weighted adjacency matrices, then A_{12} contains links from hidden nodes to measured nodes, A_{22} contains links from hidden nodes to other hidden nodes and A_{21} contains links from measured nodes to hidden nodes. The product of $A_{12}(sI - A_{22})^{-1}$ finds all paths from hidden nodes to measured nodes that goes through other hidden nodes. Then multiply this product with A_{21} on its right, we find all paths from measured nodes to hidden nodes and from these hidden nodes to other measured nodes. This final matrix $A_{12}(sI - A_{22})^{-1}A_{21}$ finds all paths from measured states to other measured states through one or more hidden nodes without going through measured nodes. This is the same as the graphical approach where we find all valid paths from measured node j to measured node i without going through any measured nodes. □

Theorem 2. *For an LTI system, $Q = 0$ iff there are no direct and indirect links between any measured states.*

Proof. From DSF, assume $Q = 0$, then all entries of Q are zero. Therefore, $A_{11} = 0$, and $A_{12}(sI - A_{22})^{-1}A_{21} = 0$. From Lemma 1 and Lemma 2, when $A_{11} = 0$, and $A_{12}(sI - A_{22})^{-1}A_{21} = 0$, there are no direct or indirect links between measured nodes. Assume there are no direct and indirect links between any measured states, then $A_{11} = 0$, and $A_{12}(sI - A_{22})^{-1}A_{21} = 0$. Therefore, $Q_{ij} = 0$. □

Theorem 3. *For an LTI system, $P = 0$ iff there are no direct and indirect links from any inputs to any outputs.*

Similar to Theorem 2, we can prove that Theorem 3 is also true. From Theorem 2 and Theorem 3, it is clear that when there is a link in DSF from node j to node i, the same link also exits from node j to node i in the signal structure generated by the graphical approach. There are no links between measured node j to node i in the graphical approach when no link exists in the DSF from node j to node i.

5.4 Case with Cancellations

Subsection 5.2 defines the process for constructing the signal structure works under the assumption that no links will be canceled from node i to node j when the netpath between them are zero. In reality, exact cancellation almost never happens.

5.4.1 Cancellation in Linear Time Invariant System

Below is an example of a cancellation of path in the signal structure of a linear time invariant system. Given a LTI system 5.1:

$$\begin{bmatrix} \dot{x}_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -3 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 \\ 1 & 0 & -4 & 1 \\ 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{5.1}$$

Its complete computational structure can be represented by first constructing the adjacency matrices: $\mathscr{X}$, $\mathscr{U}$, and $\mathscr{Y}$. The state adjacency matrix $\mathscr{X}$ is the Boolean structure of matrix A in LTI system 5.1. The input-to-state and the state-to-ouput adjacency matrix is the Boolean structure of B and C. Figure 5.2 shows the complete computational structure for 5.1.

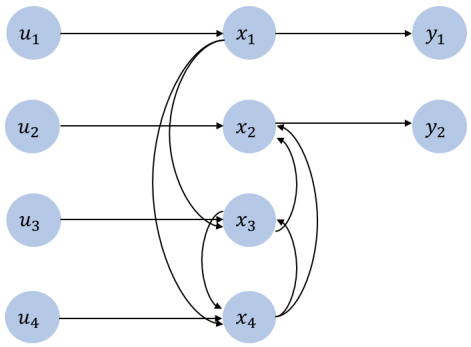


Figure 5.2: Complete computational structure for System (5.1).

Under the assumption of no cancellation of paths in the system's signal structure and following the procedure in Section 5.2, we can construct the signal structure for 5.1. First,

instantiate input nodes u_1, u_2, u_3, u_4, and measured output nodes y_1, y_2. Then, check each input in the complete computational structure. The resulting graph is shown in Figure 5.3. The signal structure shows that y_2 depends on y_1.

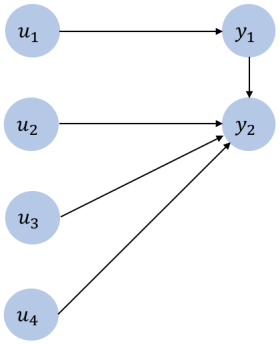


Figure 5.3: The signal structure for System (5.1) constructed by the process in Section 5.2.

Although the actual signal structure generated by DSF for this LTI system shows no edge exists between y_1 to y_2. We can quickly validate this by calculating $W = A_{11} + A_{12}(sI - A_{22})^{-1}A_{21}$.

$$A_{12}(sI - A_{22})^{-1}A_{21} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

In this case, $W = A_{11}$. By subtracting the diagonal entries of W, we get $Q = 0$. There is no direct dependency among any of the system's measured states. Such cancellation is also possible in P.

5.4.2 Cancellation in Nonlinear Systems

The following is a very simple example to illustrate that cancellations can also happen in nonlinear system. Given a nonlinear system:

$$\begin{aligned}
\dot{x_1} &= -0.5x_1 \\
\dot{x_2} &= x_3 + x_4 \\
\dot{x_3} &= \sqrt{x_1^2 + 5} \\
\dot{x_4} &= -\sqrt{x_1^2 + 5} \\
y &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}
\end{aligned} \tag{5.2}$$

The complete computational structure constructed by following the process in Section 4.2 is shown in Figure 5.4.

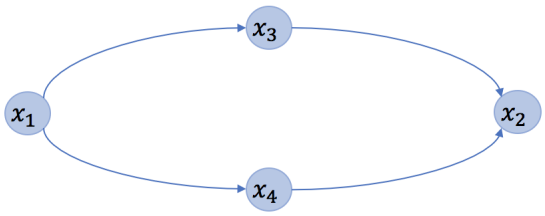


Figure 5.4: The complete computational structure for Equation (5.2).

We do not yet have the technique for computing the weights for complete computational structure of nonlinear systems, but for this example, it is clear that there is an exact cancellation in the signal structure when x_1 and x_2 are measured signals. The non-linearity is contained in only x_3 and x_4, and the relationship between x_3 and x_4 to x_2 remains to be linear. From Equation (5.2), there are two paths from x_1 and x_2 that does not go through

another measured variable: $\{x_1, x_3, x_2\}$ and $\{x_1, x_4, x_2\}$, which cancel each other. The actual signal structure for (5.2) has no links from x_1 to x_2.

5.4.3 Process with Cancellation

So far, we have defined only the process for finding the signal structure with no exact cancellation. To find the signal structure with exact cancellations like Section 5.4.1 and 5.4.2, we can find another system that well approximates the original system but has no exact path cancellation.

To illustrate the idea, consider the signal structure for the LTI system in (5.1). There are supposed to be four paths from x_1 to x_2: $\{x_1, x_3, x_2\}$, $\{x_1, x_4, x_2\}$, $\{x_1, x_3, x_4, x_2\}$, and $\{x_1, x_4, x_3, x_2\}$, but because their weights exactly cancel each other, the resulting signal structure actually has no links from x_1 to x_2. By adding a different perturbation that is small enough on each link from in Figure 5.5, we can construct a new system that well approximates the original system and these four paths no longer exactly cancel each other. Then we could apply process 5.2 to the class of systems that have exact path cancellations.

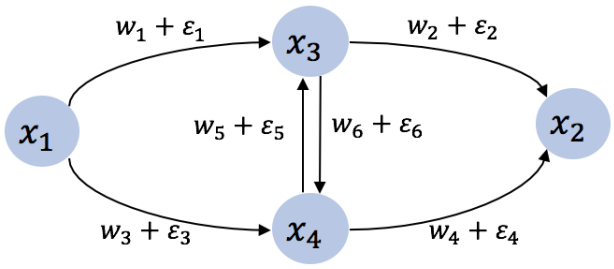


Figure 5.5: Perturbation on the complete computational structure. The four paths from x_1 to x_2 exactly cancel each other out: $\{x_1, x_3, x_2\}$, $\{x_1, x_4, x_2\}$, $\{x_1, x_3, x_4, x_2\}$, and $\{x_1, x_4, x_3, x_2\}$

Given a stable LTI state space model $\dot{x} = Ax$, we can always find a stable perturbation Δ, such that $\dot{x} = Ax + \Delta x$ stays stable with no exact cancellations and the solutions of the original system and the perturbed system stay close.

Consider a stable LTI state space model below:

$$\dot{x} = Ax. \tag{5.3}$$

We will find a stable perturbation Δ such that the perturbed system $\dot{\hat{x}} = A\hat{x} + \Delta\hat{x}$ meets the following three conditions:

1. The perturbed system $\dot{\hat{x}} = A\hat{x} + \Delta\hat{x}$ is stable.
2. The error between the solutions of the original system and the perturb system is small.
3. The perturbed system has no exact cancellations.

In order to find such Δ, we first pick an arbitrary stable Δ, where $Bool(A) = Bool(\Delta)$ and each entries of Δ is different. This guarantees that there is no exact cancellation in the perturbed system.

Then we will find a perturbation $\bar{\Delta}$ with the smallest Frobenius norm that destabilizes system $\dot{\hat{x}} = A\hat{x} + \bar{\Delta}\hat{x}$ and results in $A + \bar{\Delta}$ having one or a pair of eigenvalues at the imaginary axis. Pick a small ϵ, where $0 < \epsilon < 1$, such that $\|\epsilon\Delta\|_F < \|\bar{\Delta}\|_F$. This way, we are guaranteed to have all eigenvalue of $A + \epsilon\Delta$ on the left half plane and the perturbed system $\dot{\hat{x}} = A\hat{x} + \epsilon\Delta\hat{x}$ is stable.

The perturbed system then becomes $\dot{\hat{x}} = A\hat{x} + \epsilon\Delta\hat{x}$. This perturbed system will have no exact cancellation and is stable. Now, we need to verify that the error between this perturbed system and the original LTI system is small.

The solution to an unforced LTI system $\dot{x} = Ax$ is $x(t) = e^{At}x(0)$. And the solution of the perturbed system $\dot{\hat{x}} = A\hat{x} + \Delta x$ is $\hat{x}(t) = e^{(A+\Delta)t}x(0)$.

Let e be the error between x and $\hat{x}$, and assume $x(0) = \hat{x}(0)$,

$$e(t) = \hat{x}(t) - x(t) = e^{(A+\Delta)t}\hat{x}(0) - e^{At}x(0) = e^{\Delta t}x(0). \tag{5.4}$$

Because Δ is very small and stable, e(t) stays small and $e \to 0$ as $t \to 0$. Therefore, $\dot{\hat{x}} = A\hat{x} + \Delta\hat{x}$ meets the three conditions. Now we can find the signal structure for $\dot{x} = Ax$ using the process defined in Section 5.2 on the perturbed system. We will leave the reader to verify that the same process can be used to find the perturbation for LTI state space model with inputs.

Example 4. *System* (5.1) *is a stable LTI system with exact cancellation of paths in the signal structure. We can find a perturbed system that well approximates* (5.1) *and has no exact cancellation of paths following the above methodology.*

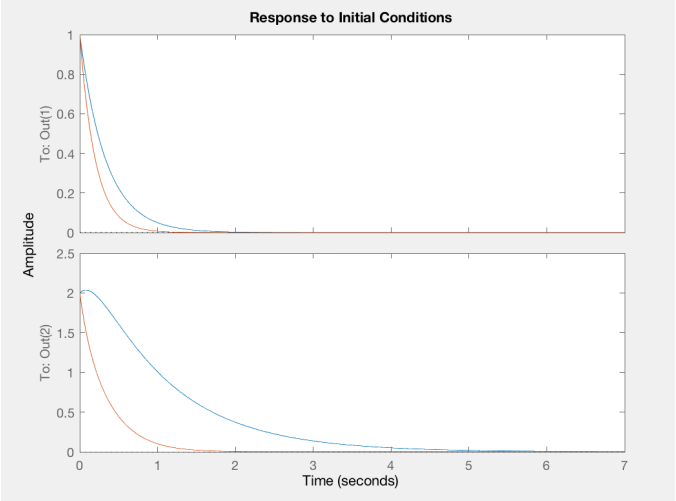
First, an arbitrary Δ *is chosen, with* $Bool(\Delta) = Bool(A)$ *and each entry of* Δ *is different:*

$$\Delta = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -2.2 & -1.3 & -1.4 \\ -1.5 & 0 & -2.6 & 1.1 \\ -1.8 & 0 & -1.9 & -2.1 \end{bmatrix}. \tag{5.5}$$

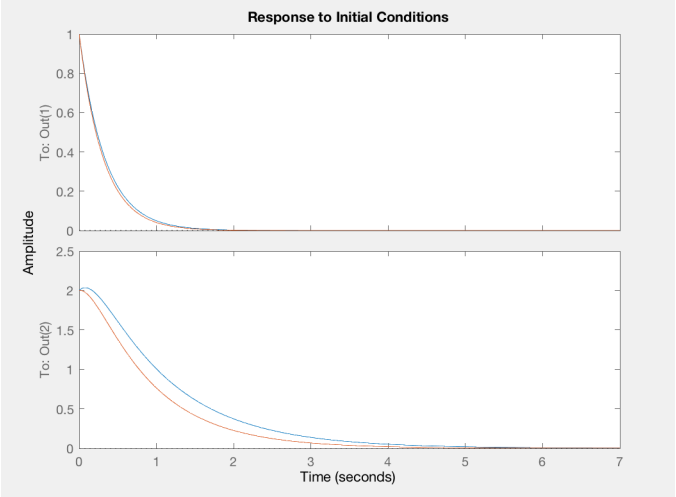
Then we can pick a small ϵ*. Any* $\epsilon\Delta$ *with Frobenius norm smaller than the Frobenius norm of perturbation* $\bar{\Delta}$*, which has the smallest Frobenius norm that results in* $A + \bar{\Delta}$ *having just one or a pair of eigenvalues at the imaginary axis, is a good candidate of perturbation that guarantees the stability of the new perturbed system. Here we simulated three values of* ϵ*: 1, 0.1, 0.01. All values of* $\epsilon\Delta$ *are verified to make the perturbed system* $\dot{\hat{x}} = (A + \epsilon\Delta)\hat{x}$ *stable. The smaller the* ϵ *is, the closer the solutions of the original and the perturbed system stay. Figure 5.6 shows the responses of the original and the perturbed system.*

The last step is to verify that the new perturbed system has no cancellation of paths in the signal structure. Computing the DSF for $\epsilon = 0.1$*:*

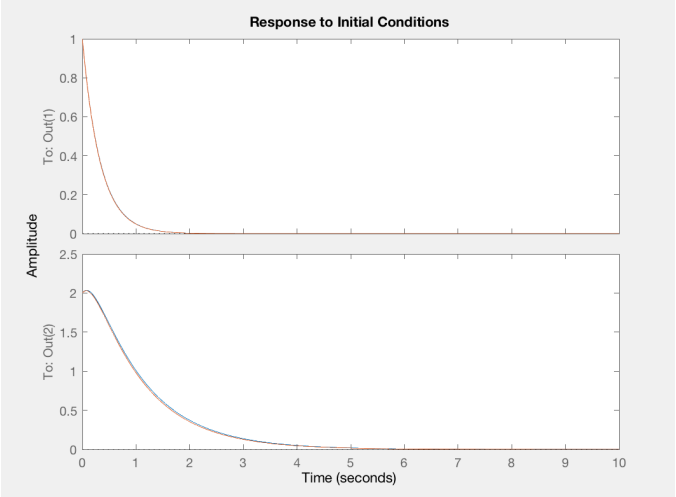
$$\hat{Q} = \begin{bmatrix} 0 & 0 \\ * & 0 \end{bmatrix}.$$



(a) $\epsilon = 1$



(b) $\epsilon = 0.1$



(c) $\epsilon = 0.01$

Figure 5.6: Simulation comparing the original and perturbed system. All three simulation has an initial condition of $x(0) = [1; 2; 3; 0]$.

*Note, * in $\hat{Q}$ represents a transfer function. The DSF for the original system is* $Q = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. *Comparing* $\hat{Q}_{21}$ *with* Q_{21}*, the path from* y_1 *to* y_2 *has been cancelled out for the original system, while with the perturbed system, there is no exact cancellation from* y_1 *to* y_2.

The perturbation $\epsilon\Delta$ *satisfies all the three conditions. Once we find a desired perturbed system with no exact cancellation, we can apply Process 5.2 on the perturbed system to construct the signal structure for the original system.*

Similar to LTI systems, we can also perturb a stable nonlinear system with exact cancellations, such that the perturbed system is stable with no exact cancellations and well approximates the original system. However, nonlinear systems need to be stable and "additively separable" in order to use this perturbation method. Given a nonlinear function $\dot{x} = f(x_1, x_2, x_3, \ldots, x_n)$, f is only additively separable if $f(x_1, x_2, x_3, \ldots, x_n) = f_1(x_1) + f_2(x_2) + f_3(x_3) + \cdots + f_n(x_n)$. When a nonlinear system is additively separable, we can design a linear perturbation Δx that satisfies the three conditions mention earlier.

Suppose $\dot{x} = f(x_1, x_2, \ldots, x_n)$ and it could be separate to $f_i(x_1, x_2, \ldots, x_n) = f_{i1}(x_1) + f_{i2}(x_2) + \cdots + f_{in}(x_n)$. Then we can find perturbation Δx:

$$f(x_1, x_2, \ldots, x_n) + \Delta x = \begin{bmatrix} f_{11}(x_1) + \Delta_{11}x_1 + f_{12}(x_2) + \Delta_{12}x_2 + \cdots + f_{1n}(x_n) + \Delta_{1n}x_n \\ f_{21}(x_1) + \Delta_{21}x_1 + f_{22}(x_2) + \Delta_{22}x_2 + \cdots + f_{2n}(x_n) + \Delta_{2n}x_n \\ \vdots \\ f_{n1}(x_1) + \Delta_{n1}x_1 + f_{n2}(x_2) + \Delta_{n2}x_2 + \cdots + f_{nn}(x_n) + \Delta_{nn}x_n \end{bmatrix}.$$

We outline the idea of proof as follow:

Consider the nominal nonlinear system:

$$\dot{x} = \begin{bmatrix} f_{11}(x_1) + f_{12}(x_2) + \cdots + f_{1n}(x_n) \\ f_{21}(x_1) + f_{22}(x_2) + \cdots + f_{2n}(x_n) \\ \vdots \\ f_{n1}(x_1) + f_{n2}(x_2) + \cdots + f_{nn}(x_n) \end{bmatrix} + g(u), \tag{5.6}$$

and u is vector valued.

1. Picking an arbitrary Δ with unique entries and let $Bool(\Delta) = Bool(\mathscr{X})$. Recall from Chapter 4.1, $\mathscr{X}$ is the state adjacency matrix of the nonlinear system's complete computational structure.
2. Picking a small ϵ, and let the perturbation be $\epsilon\Delta$.
3. Linearizing the perturbed system around the trajectory of $(u(t), x(t))$, which are solutions to the original nonlinear system in (5.6). The result will be a linear time varying(LTV) system: $\dot{x} = A(t)x(t) + B(t)u(t)$.
4. Bounding the resulting LTV system with an LTI system, such that $\|x_{ltv}(t)\| < \|x_{lti}(t)\|$, where $x_{ltv}(t)$ and $x_{lti}(t)$ are solutions to the LTV and LTI system.
5. Verifying that Δ will not destabilize the LTI system that bounds the LTV system.

If the Δ we found does not destabilize the LTI system, and the LTI system bounds the LTV system, then this Δ will not destabilize the LTV system. Base on the Lyapnov's indirect method, if the LTV system is stable, then the perturbed nonlinear system is also stable.

Since we have validated the graphical approach will produced the correct signal structure for nonlinear systems without exact path cancellation and we know that we could perturb a nonlinear system that is stable and separable, we conclude that the graphical approach is the process for a class of nonlinear systems we proposed in this thesis.

Chapter 6

Future Work

There are several directions for future work. First, the process in Chapter 5 works on a certain class of nonlinear dynamic systems. The nonlinear state space model has to be stable and separable. The work can extend the process to work on other classes of nonlinear systems. Another clear extension for this thesis is to expand the form of nonlinear system we use. We limited the output equation of the state space model in the form of $y = \begin{bmatrix} I & 0 \end{bmatrix} x$. This conveniently saves us any effort of transforming the system to partition the measured and hidden states. Future work can include systems with different forms of output equations, and discuss how to transform these systems to partition the measured and hidden states.

Secondly, the current process only constructs the signal structure without weights on any of the edges. Computing weights on the edges opens up some question. Computing the weights for links in the complete computational structure is not so trivial. Even for a simple system like $\dot{x}_1 = x_2 x_3$, putting a weight on edge $\{x_2, x_1\}$ and $\{x_3, x_1\}$ requires some thought. Another problem is how weights are aggregated when multiple paths in the complete computational structure form one link between two manifest states in the signal structure. For LTI system, the weight for a single link in the signal structure is the netpath, which is the sum of all the paths. But for nonlinear systems, the relationship of how multiple paths forms a single link is unclear.

Once we could find a way to compute the weight with a proper interpretation of how the weights work on the dependent nodes, we could apply this theoretical result to many applications. A current ongoing project created a software application that utilized the results

on the signal structures of LTI systems and helps mitigate any vulnerabilities in distributed systems. This software can be extended to solve same problems but for nonlinear distributed systems.

In summary, this work leveraged the linear DSF and some of its properties to create a signal structure for a certain class of nonlinear systems. Specifically, we defined both complete computational structure and signal structure. Then by finding the complete computational structure of a nonlinear system with the corresponding adjacency matrices $\mathscr{X}, \mathscr{U}$ and $\mathscr{Y}$, we were able to use some linear properties to construct the signal structure. My definition and the corresponding processes for finding the complete computational structure and signal structure is validated by proving the process produces the correct signal structure in linear cases and thus is a proper generalization of the existing linear theory.

References

[1] D. Grimsman, V. Chetty, N. Woodbury, E. Vaziripour, S. Roy, D. Zappala, and S. Warnick. A case study of a systematic attack design method for critical infrastructure cyber-physical systems. In *American Control Conference*, pages 296–301, Boston, USA, 2016.

[2] M. Maxwell and S. Warnick. Modeling and identification of the Sevier River system. In *American Control Conference*, pages 5342–5347, Minneapolis, USA, 2006.

[3] V. Chetty. *Theory and applications of network structure of complex dynamical systems*. Ph.D. Dissertation, Brigham Young University, Provo, UT, 2017.

[4] V. Chetty and S. Warnick. Network semantics of dynamical systems. In *Conference on Decision and Control*, pages 1557–1562, Osaka, Japan, 2015.

[5] E. Yeung, J. Gonalves, H. Sandberg, and S. Warnick. Representing structure in linear interconnected dynamical systems. In *Conference on Desicion and Control*, pages 6010–6015, Atlanta, USA, 2010.

[6] J. C. Willems. The behavioral approach to open and interconnected systems. *Control Systems Magazine*, 27(6):46–99, 2007.

[7] S. Sabau, C. Oara, S. Warnick, and A. Jadbabaie. Structured coprime factorizations description of linear and timeinvariant networks. In *Conference on Decision and Control*, pages 2720–2725, Florence, Italy, 2013.

[8] A. Rai, D. Ward, S. Roy, and S. Warnick. Vulnerable links and secure architectures in the stabilization of networks of controlled dynamical systems. In *American Control Conference*, pages 1248–1253, Montral, Canada, 2012.

[9] V. Chetty, N. Woodbury, E. Vaziripour, and S. Warnick. Vulnerability analysis for distributed and coordinated destabilization attacks. In *Conference on Decision and Control*, pages 511–516, Los Angeles, USA, 2014.

[10] A. Vosughi, C. Johnson, S. Roy, S. Warnick, and M. Xue. Local control and estimation performance in dynamical networks: structural and graph-theoretic results. In *Conference on Decision and Control*, pages 4032–4037, Melbourne, Australia, 2017.

[11] E. Young, H. Sandberg, J. Goncalves, and S. Warnick. Mathematic relationships between representations of structure in linear interconnected dynamical systems. In *American Control Conference*, pages 4348–4353, San Francisco, USA, 2011.

[12] J. Goncalves, R. Howes, and S. Warnick. Dynamical structure functions for the reverse engineering of LTI networks. In *Conference on Decision and Control*, pages 1516–1522, New Orleans, USA, 2007.

[13] V. Chetty and S. Warnick. Necessary and sufficient conditions for identifiability of interconnected subsystems. In *Conference on Decision and Control*, pages 5790–5795, Melbourne, Australia, 2017.

[14] J. Adebayo, T. Southwick, V. Chetty, E. Yeung, Y. Yuan, J. Gonalves, J. Grose, J. Prince, G. B. Stan, and S. Warnick. Dynamical structure function identifiability conditions enabling signal structure reconstruction. In *Conference on Decision and Control*, pages 4635–4641, Maui, USA, 2012.

[15] Y. Yuan, G. B. Stan, S. Warnick, and J. Gonalves. Minimal dynamical structure realisations with applications to network reconstruction from data. In *Conference on Decision and Control*, pages 4808–4813, Shanghai, China, 2009.

[16] R. Howes, L. Eccleston, J. Gonalves, G. Stan, and S. Warnick. Dynamical structure analysis of sparsity and minimality heuristics for reconstruction of biochemical networks. In *Conference on Decision and Control*, pages 173–178, Cancun, Mexico, 2008.

[17] D. Materassi and M. V. Salapaka. Methods for network identification robust with respect to uncertainties in a topology. In *American Control Conference*, pages 4680–4680, Boston, USA, 2016.

[18] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. In *IEEE Transactions on Neural Networks*, volume 1, pages 4–27, 1990.

[19] Y. Moreno M. Chavez D.-U. Hwang S. Boccaletti, V. Latora. Complex networks: Structure and dynamics. In *NPG Neurologie - Psychiatrie - Griatrie*, volume 424, pages 175–308, 2006.

[20] D. Materassi and M. V. Salapaka. Notions of separation in graphs of dynamical systems. In *Proceedings of the International Federation of Automatic Control Conference*, 2014.

[21] D. Materassi and M. V. Salapaka. On the problem of reconstructing an unknown topology via locality properties of the Wiener filter. In *IEEE Trans.Autom. Control*, volume 57, pages 1765–1777, 2012.

[22] D. Materassi. Reconstruction of topologies for acyclic networks of dynamical systems. In *American Control Conference*, pages 37–41, San Francisco, USA, 2011.

[23] D. Materassi and M. V. Salapaka. Network reconstruction of dynamical polytrees with unobserved nodes. In *Conference on Decision and Control*, pages 4629–4634, Maui, USA, 2012.

[24] P. Geiger, K. Zhang, M. Gong, D. Janzing, and B. Scholkopf. Causal inference by identification of vector autoregressive processes with hidden components. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1917–1925, Lille, France, 2015.

[25] C. Quinn, N. Kiyavash, and T. Coleman. Equivalence between minimal generative model graphs and directed information graphs. In *IEEE International Symposium on Information Theory Proceedings*, pages 293–297, 2011.

[26] S. Salehkaleybar, J. Etesami, and N. Kiyavash. Identifying nonlinear 1-step causal influences in presence of latent variables. In *IEEE International Symposium on Information Theory Proceedings*, pages 1341–1345, 2017.

[27] J. Etesami and N. Kiyavash. Measuring causal relationships in dynamical systems through recovery of functional dependencies. In *IEEE transactions on neural networks*, volume 1, pages 4–27, 1990.

[28] D. Siljac. *Large scale dynamic systems: stability and structure*. New York: North-Holland, 1978.

[29] F. Harary. *Graph theory*. Massachusetts: Addison-Wesley Pub. Co., 1969.

[30] E.Estrada. *Introduction to complex networks: structure and dynamics*, pages 93–131. Springer International Publishing, Cham, 2015.

[31] N. Wouw, E. Lefeber, and I. Arteaga. *Nonlinear systems: techniques for dynamical analysis and control*. Springer International Publishing Switzerland, 2017.

[32] F. Verhulst. *Nonlinear differential equations and dynamical systems.* Springer-Verlag Berlin Heidelberg, 1990.

[33] J.M. Souriau. *Structure of dynamic systems: a symplectic view of physics.* Birkhäuser Boston, 1997.