Title: An Information Theoretic Framework and Self-organizing Agent-based Sensor Network Architecture for Power Plant Condition Monitoring

FINAL REPORT

Reporting Period: November 1, 2011-October 31, 2016

Principal Author: Kenneth A. Loparo

Date: January 30, 2017

DOE Award Number: DE-FC2611-FE0007270

Case Western Reserve University (CWRU)
10900 Euclid Ave
Cleveland, OH 44106-7015

Abstract

A central goal of the work was to enable both the extraction of all relevant information from sensor data, and the application of information gained from appropriate processing and fusion at the system level to operational control and decision-making at various levels of the control hierarchy through: 1. Exploiting the deep connection between information theory and the thermodynamic formalism, 2. Deployment using distributed intelligent agents with testing and validation in a hardware-in-the loop simulation environment. Enterprise architectures are the organizing logic for key business processes and IT infrastructure and, while the generality of current definitions provides sufficient flexibility, the current architecture frameworks do not inherently provide the appropriate structure. Of particular concern is that existing architecture frameworks often do not make a distinction between ``data" and ``information." This work defines an enterprise architecture for health and condition monitoring of power plant equipment and further provides the appropriate foundation for addressing shortcomings in current architecture definition frameworks through the discovery of the information connectivity between the elements of a power generation plant. That is, to identify the correlative structure between available observations streams using informational measures. The principle focus here is on the implementation and testing of an emergent, agent-based, algorithm based on the foraging behavior of ants for eliciting this structure and on measures for characterizing differences between communication topologies. The elicitation algorithms are applied to data streams produced by a detailed numerical simulation of Alstom's 1000 MW ultra-super-critical boiler and steam plant. The elicitation algorithm and topology characterization can be based on different informational metrics for detecting connectivity, e.g. mutual information and linear correlation.

Table of Contents

1.0 Executive Summary of Accomplishments

A central goal of the work was to enable both the extraction of all relevant information from sensor data, and the application of information gained from appropriate processing and fusion at the system level to operational control and decision-making at various levels of the control hierarchy through:

1. Exploitation of the deep connection between information theory and the thermodynamic formalism,
2. Deployment using distributed intelligent agents with testing and validation in a hardware-in-the loop simulation environment.

Enterprise architectures are the organizing logic for key business processes and IT infrastructure and, while the generality of current definitions provides sufficient flexibility, the current architecture frameworks do not inherently provide the appropriate structure. Of particular concern is that existing architecture frameworks often do not make a distinction between ``data'' and ``information.'' This work defines an enterprise architecture for health and condition monitoring of power plant equipment and further provides the appropriate foundation for addressing shortcomings in current architecture definition frameworks through the discovery of the information connectivity between the elements of a power generation plant. That is, to identify the correlative structure between available observations streams using informational measures. The principal focus here is on the implementation and testing of an emergent, agent-based, algorithm based on the foraging behavior of ants for eliciting this structure and on measures for characterizing differences between communication topologies. The elicitation algorithms are applied to data streams produced by a detailed numerical simulation of Alstom's 1000 MW ultra-super-critical boiler and steam plant. The elicitation algorithm and topology characterization can be based on different informational metrics for detecting connectivity, e.g. mutual information and linear correlation.

Significant Results

This project proposes an algorithm from the foraging behavior of ants to discover the information connectivity between the elements of a system. We applied our algorithm to the elements of a power generation plant, have used graph similarity measures and two change point detection techniques, spectral graph distance and graph diameter distance, for detecting the changes in the system structure based on the discovered topologies from our algorithm. Graph diameter distance was one approach for detecting the changes in the system, and we also examine another graph similarity measure and use it for detecting the current operating condition and changes in that operating condition.

Challenges/Opportunities: Requirements for improved operating efficiency, increased reliability, and enhanced system performance can be achieved through the use advanced sensing, communication, and control technologies integrated into ``smart'' systems with increased functionality and, usually, more complexity. Large-scale power generation systems are ``cyber-physical'' systems where communication and control technologies are embedded within the physical subsystems of the plant to improve their performance, monitor their condition, and coordinate their behaviors. While the increased capabilities provide greater flexibility and utility, the associated engineering problems are not amenable to neat solutions devised within the confines a single discipline.

In many applications (e.g. operational monitoring), instrumentation can be applied in a case-by-case manner

to obtain useful observations, however, this point-wise approach rapidly loses utility in more demanding applications such as device, subsystem and system health and condition monitoring. During normal operation, the complexity of assimilating observations falls within intuitive limits of human operators. However, when operating outside of normal bounds or when contemplating more sophisticated approaches to operational functions, such as replacing schedule-based maintenance with condition-based maintenance, point-wise approaches are inadequate.

The following observations are essential to our development: 1) assumptions that are valid for individual elements of a system when considered in isolation are not necessarily valid when these elements are considered in the context of the interconnected system, and 2) a system can exhibit dynamic behaviors that are not present in any component considered in isolation but that emerge from the interactions between its separate components. Thus decompositions that permit component-wise approaches are not generally applicable in ``complex'' systems. Real-world power generation systems are: 1) nonlinear and the range of possible behaviors destroys intuition and precludes simple frameworks for composing system models, 2) subject to uncertain (random) events, inputs, and disturbances and thus are stochastic systems, and 3) large-scale as they are composed of a large number of disparate elements where potential interactions scale geometrically. This requires that the information contained in the data from disparate instrumentation systems are considered within a larger (systems) context as a network of sensors.

Table 1.1: Project Objectives and Planned Outcomes

TABLE of PROJECT OBJECTIVES	
OBJECTIVE 1	Develop an intelligent **agent-based information-theoretic architecture** for advanced power plant applications
PRODUCT	Architecture Description Document that describes the architectural components, their interactions, and organizing principles.
OBJECTIVE 2	Develop **computational algorithms** to be employed by intelligent agents to maximize the collection, transmission, aggregation, and conversion of data into actionable information for monitoring, diagnosis, prognosis and control of the power plant
PRODUCT	Algorithm Description Document that describes computational algorithms available to intelligent agents.
PRODUCT	Algorithm & model library - Library of algorithms implemented in Matlab - Library of power plant process and equipment models
OBJECTIVE 3	**Evaluate** the effectiveness of these **computational algorithms** in organizing agents for maximizing information content from power plant data through an **integrated hardware-in-the-loop simulation test bed**
PRODUCT	Analytic tools for measuring effectiveness of computational algorithms
PRODUCT	Algorithm Evaluation Report documenting the measured effectiveness of each computational algorithm.
PRODUCT	Simulation test bed with a capability to do hardware-in-the-loop testing
PRODUCT	Demonstration of a simulated power generation unit that illustrates application of the proposed framework to a power plant. Demonstration will

	include typical faults on sensors, actuators, and process system elements.

Table 1.2: Milestone Log

ID	WBS Element	Description	Planned Complete Date	Actual Complete Date	Comment
1	1	Kick-off Meeting	28-Nov. 2011		
2	1	Close-out Meeting	31-Oct. 2014		
		Information Architecture Milestones			
3	2	Network Performance Objectives Specified	26-Oct. 2012		
		Virtual Sensor/Sensor Processing Milestones			
4	3	Virtual Sensor Methods Identified	26-Oct. 2012		
		System Integration Milestones			
5	5	Preliminary Design Review (PDR)	26-Jul. 2013		
6		Critical Design Review (CDR)	31-Oct. 2013		
7		Demonstration System Completed	26-Sept. 2016		
8		System Demonstration Results Completed	31-Oct. 2016		

Application/Approach: The information-theoretic sensor network architecture for health and condition monitoring of power plant equipment addressed here has several crucial features: 1) individual sensors and instrumentation packages are considered as members of a sensor network comprised of heterogeneous sensor elements, and 2) the sensor network has a hierarchical structure wherein information obtained from sensors or sensor fusion processes at lower levels of the hierarchy is combined to provide aggregate views of subsystems or ``summary'' information necessary for information assimilation at higher levels of the hierarchy. The hierarchical structure is adaptive and has the ability to change in order to meet the requirements of different operating conditions (i.e. load, active equipment), equipment condition (e.g. component faults), or operational needs (e.g. detection versus diagnosis/prognosis verses control).

The sensor network needs to be ``self-organizing'' so that it can be reconfigured based on relationships across sets of components, subsystems, and the system as a whole, to provide the functionality and robustness required in an operational setting. The ability of the network to automatically determine alternative information paths to detect sensor faults and reconstitute lost sensing capabilities from remaining

sensor processes, is an essential feature. Figure 1 provides an overview of the information-theoretic architecture for future power plants.

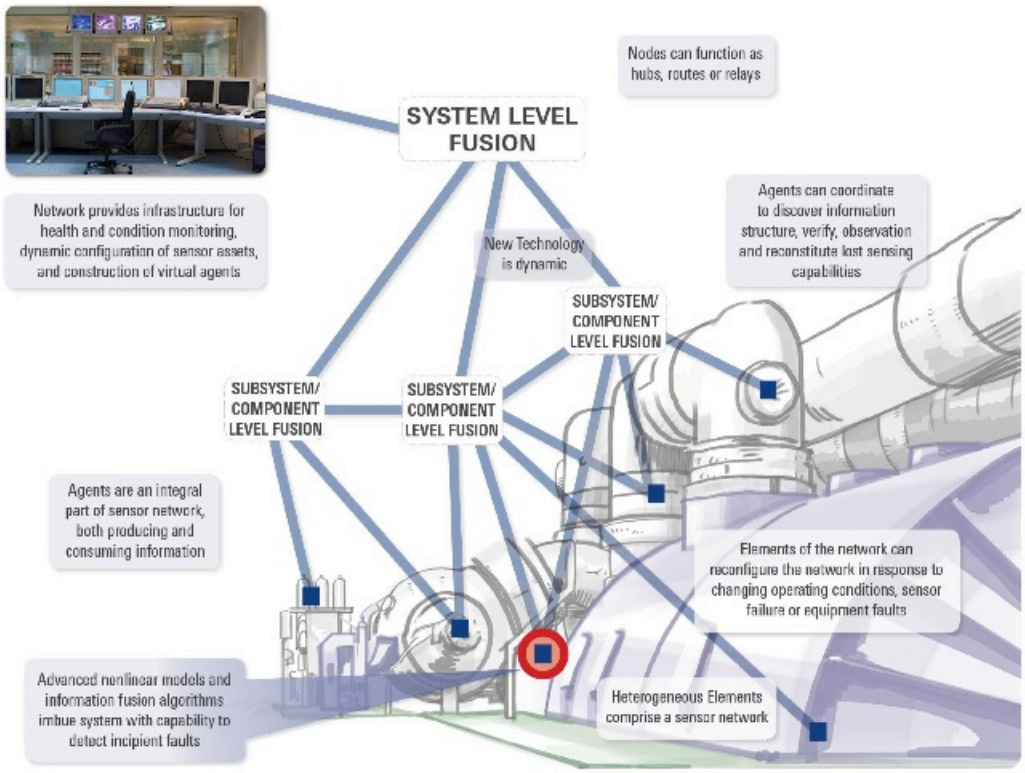


Figure 1.1: Information-theoretic Architecture for Future Power Plants

2.0 Information Theoretic Framework

The crucial notion in our design paradigm is that **information** is not the same as **data**, in particular information is the amount of ``surprise'' contained in a new observation or measurement and, as such, has the potential to tell you something new about the current situation. This concept, through the realization of innovation representations of stochastic signals, is the basis for most of modern estimation theory. The basic idea in our framework is to transmit information not data, and to only transmit data that is informative in the context of a given problem.

Enterprise information systems are **communication systems** (Figure 2.2), with the purpose of delivering a message W that contains actionable information. Of additional importance in our framework is that the physical substrate for the instrumentation is also a communication system (this is the essence of the cyber-physical system paradigm) where the messages are now the states of constituent elements and are transmitted via physical phenomena (e.g. vibration, heat, chemical concentrations) to other elements in the system. The component and system dynamics pose constraints and limitations on the observations and exogenous (or endogenous) noise sources, such that the observation or message W is encoded into a signal X that is transmitted over a communication channel. In the case of sensor data, the communication channel commonly packetizes X, incurs delays and drop outs, that alter the original signal so that what is delivered is the signal Y which must then be unpacked and decoded to provide an estimate of the original message W. Thus any observation must be considered in the context of the system/environment from which it is drawn and information on the states of system elements is contained in the states of other

elements. An information-centric view of systems is critical for accommodating these considerations and for exploiting the synergies that results from modeling sensor and cyber-physical systems as communication processes. In doing so, we are also able to effectively bring to bear the accomplishments of over 50 years of research and development in information theory.

Figure 2.2: Communication System

Information Theory: Information theory provides the basic notions needed to quantify and characterize the flow of information, as opposed to data. Given that information is the amount of ``surprise'' contained in data, in 1948 Claude Shannon proposed a fundamental measure of information (**Shannon Entropy**) for a discrete random variable X taking values in X, known as the **alphabet,** with probability mass function, $p(x) = \Pr(X = x)$, $x \in \mathrm{X}$, given by the following formula:

$$H(X) = \sum_{x \in \mathrm{X}} p(x) \log_d p(x)$$

Shannon entropy can be interpreted as the average word size needed to specify the events and the logarithmic base (*d*) most commonly takes the value *d=2* so that it has units of bits. Shannon entropy extends to the multivariate case, for example, to the joint and conditional entropy of a pair of discrete RVs *(X,Y)*, with joint and conditional distributions given by $p(x,y)$ and $p(x \mid y)$, respectively.

An important quantity related to Shannon entropy is **mutual information**. Given two discrete RVs *(X,Y)* with joint distribution $p(x,y)$ and marginal distributions $p_X(x)$ and $p_Y(y)$ the mutual information *I(X;Y)* is given by:

$$I(X,Y) = \sum_{X \in \mathrm{X}, Y \in \mathrm{Y}} p(x,y) \log_2 \frac{p(x,y)}{p_X(x) p_Y(y)}$$

One interpretation of mutual information is as a measure of information loss that is inherent if two RVs are independent. Another interpretation is the reduction in the uncertainty of a RV *X* due to knowledge of another RV *Y* as given by:

$$I(X,Y) = H(X) - H(X \mid Y)$$

Information Channels: **Information** or **communication channels** are the fundamental mechanisms through which information is communicated between disparate elements of a given system. Let X and Y be the input and output alphabets and \ respectively, and let Σ be the set of channel states. A **discrete channel** is a system of probability mass functions:

$$p_n(\beta_1,...,\beta_n \mid \alpha_1,...,\alpha_n;\sigma)$$

where $\alpha_1,...,\alpha_n \in \mathrm{X}$, $\beta_1,...,\beta_n \in \mathrm{Y}$, and $\sigma \in \Sigma$ for $n = 1,2,...$ The probability mass functions can be interpreted as the probability that the sequence $\beta_1,...,\beta_n$ will be produced if the input sequence is $\alpha_1,...,\alpha_n$ and the initial state of the channel is σ. The information processed by the channel can be characterized in terms of the mutual information between the input and output sequences, and the channel transmittance *T* is thus given by:

$$T(\mathrm{X},\mathrm{Y}) = H(\mathrm{X}) - H(\mathrm{X}\,|\,\mathrm{Y})$$

Note that the information processing is a function of the probability mass function on X, and the probability mass function can be varied until the channel transmittance is at its maximum, known as the **channel capacity**.

Systems and Information: The development of our information-theoretic architecture exploits the properties of information to deduce a useful structure for information flow and management. The properties of information provide a fundamental basis for the decomposition of systems and hence a structure for the transmission and combination of observations at desired levels of resolution (e.g. component, subsystem, system). The basic idea is that the generalization of information theory to N-dimensions can be viewed as a statistical analysis tool for understanding systems in terms of the information geometry of its variables.

Major advantages of the information-theoretic approach over other statistical analysis techniques are: 1) it permits the measurement and analysis of rates of constraints (i.e., conditioned on history), and 2) a robust means for system decomposition follows from the decomposition of constraints provided by the axiomatic properties of information (i.e. additivity and branching).

Information Rates: Information measures capture the variability of a system and relationships between its disparate variables. These measure are robust to nonlinearity, but must be modified for application to dynamical systems to account for constraints on present values based upon past history of the system. These constraints can be described by the **entropy rate**, defined as the entropy of *X* conditioned on all of its prior values. An alternative formulation is available by recognizing that the total uncertainty of a long sequence $< X_1,...,X_n >$ is (approximately) equal to the entropy rate times the length of the sequence. The entropy rate is then defined as:

$$\overline{H}(X) = \lim_{n\to\infty} \frac{1}{n} H(X_1,...,X_n)$$

Information Structure: The properties of information and the basic decomposition principles they engender will be used to identify a optimal communications architecture for a heterogeneous sensor network in terms of the information properties of the associated observation processes and the required system output and performance requirements. This necessarily entails the aggregation of information from parallel observation processes to construct estimates at each level of the network (sensor, subsystem, system) as well fusion of disparate information sources to construct estimates between various levels of the sensor network such as: 1) aggregating information from multiple sensors to reconstitute lost sensing capability; 2) estimating the condition of a particular component via the fusion of sensor information from multiple phenomena pertaining to lower level dynamics.

The properties of information measures (and their rates) provide a robust basis for quantifying the correlative structure of systems that enable system decomposition (i.e. identification of independent or at least weakly interacting subsystems), information fusion (i.e. determining which subsystems interact and how they are related), and mesoscopic modeling (i.e. characterizing system behavior at a useful level of resolution in terms of appropriately chosen **summary variables**). To this end, a system shown schematically in Figure 2.3 is defined as a set of ordered variables with inputs from the environment and directly observable (output) variables.

As shown in Figure 2.4, the system can be partitioned into N disjoint subsystems each comprising a set of internal variables. The subsystem can have (local) inputs from the system environment, and has a set of output and internal variables. Each subsystem can also be partitioned similarly and thus examination of a

system in terms of the ``communication topology'' provides a hierarchical system decomposition.

The systems are characterized by their associated variables and, hence, the communication topology associated with a system can be described in terms of information-theoretic measures. The definitions of information measures can be extended to quantify information properties associated with systems, their constituent elements (subsystems), or communication links between constituent elements. For example, the Shannon Entropy associated with a system is:

$$H(S) = \sum_{s \in S} p(s) \log_2 p(s)$$

where *S* is the set of possible values of $X_1,...,X_n$ and *p* is the joint probability mass function. Other measures on the system can likewise be defined.

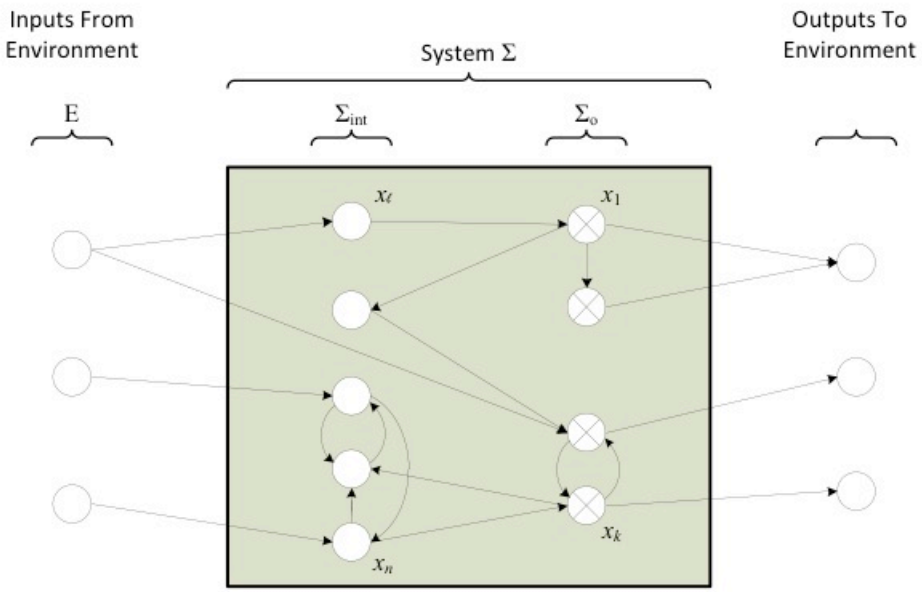


Figure 2.3: Overall System Structure

The definition of information measures on systems provides a simple calculus for partitioning the system, for example through the mutual information rate of a system defined by:

$$\bar{I}(\Sigma_i,\Sigma_j) = \overline{H}(\Sigma_i) + \overline{H}(\Sigma_j) - \overline{H}(\Sigma_i,\Sigma_j)$$

where Σ_i and Σ_j denote subsytems and the last term on the right hand side is the joint entropy rate of the subsystems, which captures the relatedness or constraint between the two systems and is an upper bound on the information transmission between the disjoint sets of variables defining Σ_i and Σ_j. If this term vanishes, the two systems are independent and thus the mutual information rate of a system provides a basis for determining a information-theoretic partition such that its subsystems are independent (or approximately independent).

In addition to these properties of information measures (and their rates), several additional ``Laws of Information'' exist that govern the interactions between the constituents of a system and the external environment. These laws clarify the fundamental tension between throughput, blockage, and processing rates and thus provide the theoretical basis for the design of fusion processes and hence the machinery for hierarchical decomposition in a general setting. Imposing a hierarchical decomposition implies the construction of system descriptions at different levels of resolution and scope. As the decomposition proposed here is determined by the statistical properties of the sensor data streams rather than by physical organization of the equipment, the development of summary variables that are not necessarily tied to the condition of specific subsystems or sensors but rather to a more general notion of overall system condition is implied. The deep relationship between information theory and thermodynamics provides the fundamental mechanism for extracting these mesoscopic system descriptions and provides a valuable framework for diagnosis and prognosis at the system level.

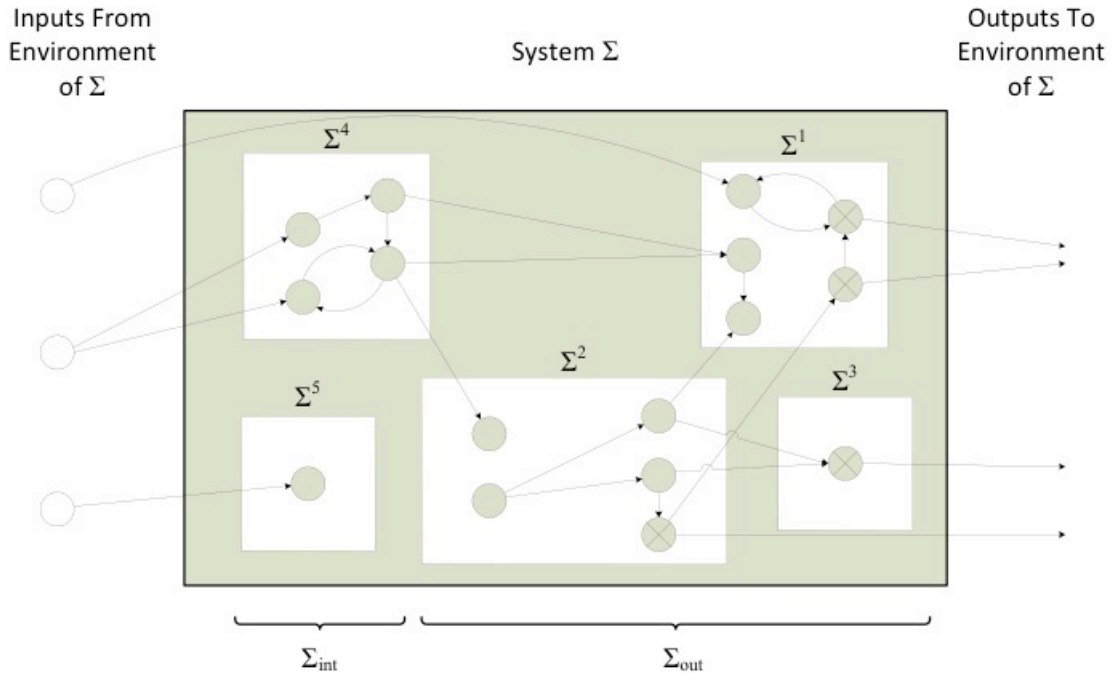


Figure 2.4: System Decomposition

Illustrative Application: This section presents an exemplary application of the information-theoretic framework that illustrates its support for the detection and diagnosis of faults in rotating equipment and in the embedded sensing systems, and, at least partially, the reconstruction of lost sensing capabilities resulting from sensor or communication failures.

Consider a pair of rotating machines (e.g. turbine-generator pairs) modeled as modified Jeffcott rotor systems with damped flexible bearing pedestals. The Jeffcott rotors consist of large unbalanced disks mounted at the midpoint of massless flexible shafts supported at their endpoints by hydrodynamic bearings. Assume that the pedestal stiffness and damping are low in order to minimize vibration transmission and, further, that the rotor systems are instrumented such that the vibration (e.g. acceleration) levels can be measured at each bearing location.

The rotor systems communicate with one another through both electrical connection (i.e. both generators are connected to the same voltage bus) and through mechanically transmitted vibration. The pedestals act as vibration isolation systems and thusduring normal operation the information transmission ($\overline{T}$) between rotors is minimal and the mutual information between bearing measurements on the same rotor is high. Denoting the vibration at each bearing by x_{ij} and the corresponding observation available by y_{ij} where the subscripts index the rotor, and the bearing, respectively, we obtain:

$$\overline{T}(y_{ik}, y_{jl}) \approx 0 \ \forall i,j,k,l \in \{1,2\} \text{ with } i \neq j$$
$$\overline{T}(y_{ik}, y_{il}) \neq 0 \ \forall i,k,l \in \{1,2\}$$

Suppose the pedestal associated with x_{11} becomes damaged or begins to suffer the effects of wear, and its effectiveness as an isolator is diminished and transmitted vibration increases. This increase in transmitted vibration will result in an increase in information transmission between y_{11} and all other observations. Thus,

$$\overline{T}(y_{11}, y_{2j}) \neq 0 \ \forall j \in \{1,2\}$$

and a simple thresholding condition can be used to detect and localize the fault. Moreover, as a small change in either damping or stiffness will induce a significant change in transmissibility in this case, the change in communication topology (i.e. x_{11} is now communicating with bearings y_{21} and y_{22}) is a leading indicator of degrading health. A more sophisticated analysis, perhaps cued by detection of the change in communication topology, can be used to improve confidence in the detection, and to provide more detailed diagnosis and prognosis on the basis of signatures and additional information.

Another failure of interest is the failure of sensing elements. By examining the transmission rates, the loss of communication between y_{11} and y_{12}, i.e.

$$\overline{T}(y_{11}, y_{12}) = \overline{T}(y_{12}, y_{11}) \neq 0$$

is likely to indicate a sensor, rather than process, fault. Thus examining the communication topology can also be used to detect faults in the instrumentation in addition to faults in the equipment. In this case, however, the loss of information precludes the identification of the precise sensor that has failed, and more sophisticated analyses is required to identify which sensor has failed. Once diagnosed as a sensor fault, the additional communication channels can be used to reconstitute lost sensing capability. For example, assuming normal operation in which bearings on one rotor are isolated from those on the other, one may assume that y_{12} is dependent only upon x_{11} and x_{12} and is conditionally independent of x_{21} and x_{22}. In this case, y_{21} or y_{22}, y_{11}, and y_{12} form a Markov chain. The transmission between y_{12} and y_{21} provide lower bounds for the information transmission between y_{11} and y_{21} or y_{22} e.g.: $\overline{T}(y_{21}, y1) \geq \overline{T}(y_{21}, y_{12})$ and the entropy rates $\overline{H}(Y_{21})$ and $\overline{H}(Y_{22})$ provides the corresponding upper bounds. The key idea here is that, through examination of the information transmission, it is not only possible to detect sensor failures but to reconstitute, at least partially, the lost capability in sensing without using a priori knowledge of the system dynamics. With this additional knowledge, more sophisticated analyses are possible, thus providing an avenue for significantly tightening these bounds and improving overall system performance.

Connection between Control Theory and Information theory: Past research has investigated the connection between control theory and information theory [1-8]. This work primarily focused on the observation/estimation error filtering problem and/or feedback control, and information theory was used to develop a deeper understanding of the system from a control engineering perspective. It is possible to investigate how control theory enhances understanding of information theory, and we follow Shannon's

original work where input data (sender) is processed through an I/O system (communication channel) to generate output data (receiver).

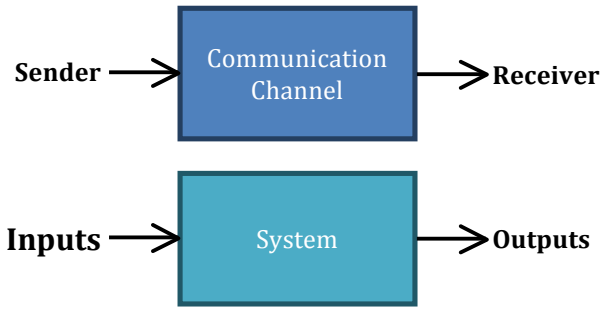


Figure 2.5: Communication channel as an I/O system

A first step is to investigate the effect the relationship between controllability and observability of a realization and the information measures applied to inputs, states and outputs. We begin with the study of linear and bilinear systems.

Consider the discrete system

$$\begin{bmatrix} x_k^{(1)} \\ x_k^{(2)} \end{bmatrix} = \begin{bmatrix} f_1(x_{k-1}) + g_1\left(u_{k-1}^{(1)}, u_{k-1}^{(2)}\right) + v_{k-1}^{(1)} \\ f_2(x_{k-1}) + g_1\left(u_{k-1}^{(1)}, u_{k-1}^{(2)}\right) + v_{k-1}^{(2)} \end{bmatrix}$$

$$\begin{bmatrix} y_k^{(1)} \\ y_k^{(2)} \end{bmatrix} = \begin{bmatrix} q_1\left(x_k^{(1)}\right) + w_k^{(1)} \\ q_2\left(x_k^{(2)}\right) + w_k^{(2)} \end{bmatrix}$$

Where

$u_k^{(1)}, u_k^{(2)}$ are input data at time k
$x_k^{(1)}, x_k^{(2)}$ are internal states of the system at time k

$y_k^{(1)}, y_k^{(2)}$ are output data at time k
$v_k^{(1)}, v_k^{(2)}$ are noises in the system
$w_k^{(1)}, w_k^{(2)}$ are noises in the observations

For simplicity v_k and w_k are white noise sequences. In addition, we also represent u_k as a "white noise" process in a different context than v_k and w_k. For v_k and w_k, we usually have no information about the disturbances, so white noise presumably represents the worst-case scenario. In contrast, a "white noise" model for u_k means that we assume the input data contains maximum information.

The information measure we focus on is the mutual information between two data streams in the system. For this purpose, we define the additional notation: $z_{1:k} = \{z_1, z_2, \dots, z_k\}$

The mutual information between a group of observations and a group of controls can be express by:

$$I\left(y_{1:k}^{(i)}; u_{1:k}^{(j)}\right) = I\left(x_{1:k}; u_{1:k}^{(j)}\right) - I\left(x_{1:k}^{(j)}; u_{1:k}^{(j)} | x_{1:k}^{(i)}\right) - I\left(x_{1:k}^{(i)}; u_{1:k}^{(j)} | y_{1:k}^{(i)}\right)$$

The term on the left side of the equation is the mutual information between observations, $y_{1:k}^{(i)}$ and

controls, $u_{1:k}^{(j)}$; the first tem of the right side of the equation is the mutual information between the internal states, $x_{1:k}$, and the control inputs, $u_{1:k}^{(j)}$. The next term is the measure of how much information about the internal state, $x_{1:k}^{(j)}$ which is not directly observed in $y_{1:k}^{(i)}$, or the control inputs, $u_{1:k}^{(j)}$, is contained in direct observation of the state, $x_{1:k}^{(i)}$. The information measure $I\left(x_{1:k}^{(j)}; u_{1:k}^{(j)} | x_{1:k}^{(i)}\right)$ decreases when $x_{1:k}^{(i)}$ contains more information about $x_{1:k}^{(j)}$ or $u_{1:k}^{(j)}$ causing an increasing in $I\left(y_{1:k}^{(i)}; u_{1:k}^{(j)}\right)$. The interpretation of the last term is similar to the previous term except that it represents how much $y_{1:k}^{(i)}$ tells us about $x_{1:k}^{(i)}$ or $u_{1:k}^{(j)}$. We can see that with a specified system structure there is a chain of information processing that can give us insight into how mutual information between set of inputs and outputs are related through internal states.

Controllability and observability, respectively, relate to how inputs influence states and how outputs influence states in a realization of a given input/output system. Specifically, a controllable region in the state space has the property that there exists control inputs that can achieve point-to-point steering in that region. Analogously, an observable region in the state space has the property that from observations of states in this region it is possible to determine the initial starting point (initial condition) of the state trajectory. of internal states such that if the system start from anywhere inside this region, We believe that controllability and observability have information-theoeretic analogs, and next we demonstrate the relationship between controllability/observability and mutual information in a linear system.

Linear systems: Consider the linear discrete-time system

$$x_k = Ax_{k-1} + B\begin{bmatrix} u_{k-1}^{(1)} \\ u_{k-1}^{(2)} \end{bmatrix} + v_{k-1}$$

$$\begin{bmatrix} y_k^{(1)} \\ y_k^{(2)} \end{bmatrix} = Cx_k + w_k$$

This system can be transformed into so-called controllable and observable forms by a linear change of coordinates in the state space. For example, let $i = 1$ and $j = 1$, we get

$$\begin{bmatrix} x_k^{\prime(c)} \\ x_k^{\prime(uc)} \end{bmatrix} = \begin{bmatrix} A_{11}' & A_{12}' \\ 0 & A_{22}' \end{bmatrix}\begin{bmatrix} x_{k-1}^{\prime(c)} \\ x_{k-1}^{\prime(uc)} \end{bmatrix} + \begin{bmatrix} B_{11}' & B_{12}' \\ 0 & B_{22}' \end{bmatrix}\begin{bmatrix} u_{k-1}^{(1)} \\ u_{k-1}^{(2)} \end{bmatrix} + T^{(c)}v_{k-1}$$

$$\begin{bmatrix} y_k^{(1)} \\ y_k^{(2)} \end{bmatrix} = C'x_k' + w_k$$

$$\begin{bmatrix} x_k^{\prime\prime(o)} \\ x_k^{\prime\prime(uo)} \end{bmatrix} = \begin{bmatrix} A_{11}'' & 0 \\ A_{21}'' & A_{22}'' \end{bmatrix}\begin{bmatrix} x_{k-1}^{\prime\prime(o)} \\ x_{k-1}^{\prime\prime(uo)} \end{bmatrix} + B''\begin{bmatrix} u_{k-1}^{(1)} \\ u_{k-1}^{(2)} \end{bmatrix} + T^{(o)}v_{k-1}$$

$$\begin{bmatrix} y_k^{(1)} \\ y_k^{(2)} \end{bmatrix} = \begin{bmatrix} C_{11}'' & 0 \\ C_{21}'' & C_{22}'' \end{bmatrix}x_k'' + w_k$$

Where

$x_k' = T^{(c)}x_k$ and $x_k'' = T^{(o)}x_k$

$T^{(c)}$ and $T^{(o)}$ are linear transformations that transform the original system into controllable and observable forms respectively. Now, we can write the mutual information for any inputs and outputs as

$$I\left(y_{1:k}^{(i)}; u_{1:k}^{(j)}\right) = I\left(x_{1:k}^{\prime(c)}; u_{1:k}^{(j)} | x_{1:k}^{\prime(uc)}\right) - I\left(x_{1:k}^{\prime\prime(uo)}; u_{1:k}^{(j)} | x_{1:k}^{\prime\prime(o)}\right) - I\left(x_{1:k}^{\prime\prime(o)}; u_{1:k}^{(j)} | y_{1:k}^{(i)}\right)$$

The right hand side in equation above shows how controllability and observability affect the information measure of interest. Controllability defines an upper bound of $I\left(y_{1:k}^{(i)};u_{1:k}^{(j)}\right)$ because we know that $u_{1:k}^{(j)}$ and $x_{1:k}^{\prime(uc)}$ are not related and $x_{1:k}^{\prime(c)}$ is driven by both $x_{1:k}^{\prime(uc)}$ and $u_{1:k}^{(j)}$ so $I\left(y_{1:k}^{(i)};u_{1:k}^{(j)}\right)$ will be less than the maximum possible mutual information, $I\left(x_{1:k}^{\prime(c)};u_{1:k}^{(j)}|x_{1:k}^{\prime(uc)}\right)$. Similarly, for the unobservable states in the realization, the second term is related to the amount of information about the control inputs, $u_{1:k}^{(j)}$, contained in an observable states, $x_{1:k}^{\prime\prime(o)}$. Finally, the third term is inversely proportional to the information in the observations, $y_{1:k}^{(i)}$, for the observable states, $x_{1:k}^{\prime\prime(o)}$. In conclusion, the mutual information between the input and output data streams is governed by the controllability and observability in the linear system.

Computation of Information measures in linear systems: Consider the general linear system

$$x_k = Ax_{k-1} + Bu_{k-1} + v_k$$
$$y_k = Cx_k + Du_k + w_k$$

With an assumption that v_k and w_k are independent and white noise sequences.
We can compute the upper and lower bounds of mutual information between all inputs and all outputs of the system by

$$I(y_{1:k};u_{1:k}) \le I(x_{1:k};u_{1:k})$$
$$I(y_{1:k};u_{1:k}) \ge h(u_{1:k}) - \sum_{l=1}^{k} h(u_l|u_{l-1}) - \sum_{l=1}^{k-1} h(y_{l+1},y_l|u_l) + \sum_{l=1}^{k-1} h(y_{l+1},y_l|u_{l-1})$$

Although, the computation of $I(y_{1:k};u_{1:k})$ is possible, it is tedious. Thus, we omit it here. For this linear system with the additional assumption that u_k is a white noise, we can also compute the mutual information between inputs and internal states by

$$I(x_{1:k};u_{1:k}) = \sum_{l=1}^{k}\big(h(Bu_l + v_l) - h(v_l)\big)$$

Computation of Information Measure in Time-series data: Many researchers have investigated the computation of entropy and mutual information of time-series data. For example, the computations of entropy for discrete-valued random processes are given in [15-19], and the estimation of entropy and/or mutual information or continuous-valued random signals is given in [20,21]. However, all of these computations require very strong assumptions, e.g., ergodicity and/or strict stationarity of the random signals, and the meaning of these computations when these assumptions are not satisfied needs to be further investigated. In real application, we know that verifying the strict stationarity and/or ergodicity of measured signals is very difficult. In contrast, is we know the dynamical system that is generating the time series data, the computation of entropy measures of the time series data is possible, but usually tedious. In this section, we propose an approach that could be used to compute mutual information for any time-series data.

It is well known that, for any column random vector, X,

$$h(X) \le h\big(N(m,\Sigma_X)\big)$$

Where

$m = E\{X\}$ and $\Sigma_X = E\{(X-m)(X-m)^T\}$

It follows that

$$\frac{1}{k}h(x_{1:k}) \leq h\big(N(m, \bar{\Sigma}_X)\big)$$

Where

$$\bar{\Sigma}_X = \frac{1}{k}\sum_{l=1}^{k}\Sigma_{X_l}$$

and from this expression we can find a lower bound of any time-series data by using the sample covariance. For any time-series data $x_{1:T}$ with sample covariance, Σ, we have

$$\frac{1}{k}h(x_{1:k}) \leq h\big(N(m, \Sigma)\big)$$

Although, this computation is simple, it is a very loose upper bound and we also need to estimate a lower bound of entropy to be able to obtain bounds on mutual information. To be precise, $I(X;Y) = h(X) - h(X|Y)$ and the subtraction of two upper bounds is not meaningful.

Motivation: To deal with this computation, we use the concept of space partitioning and symbolic dynamics. For example, the time-series data and partition in Figure 2.6 gives the symbol sequence "ccddddcbaaaabcddddcc".

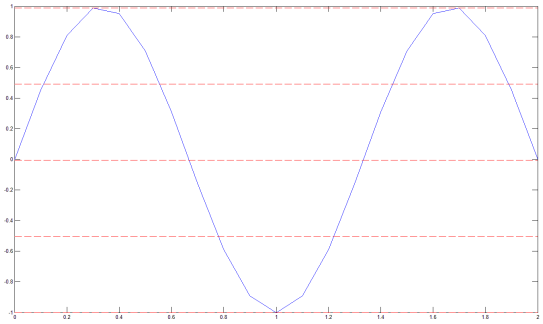


Figure 2.6: Space partitioning of a sinusoidal waveform

A node and edge representation as in Figure 2.7 can be used to represent this symbolic system as a hidden Markov model (HMM).

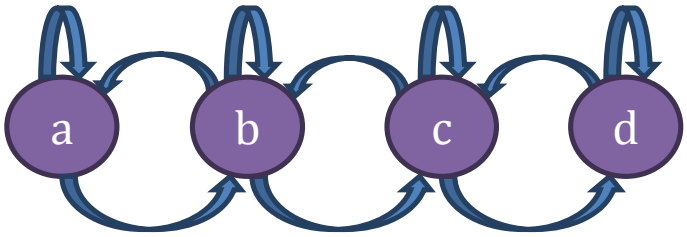


Figure 2.7: Nodes and edges representation of time-series data and partition in Figure 2.6

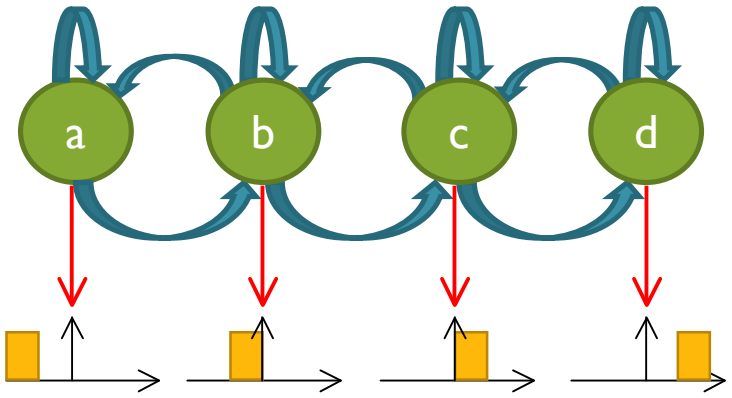


Figure 2.8: HMM model for of time-series data and partition in Figure 2.6

We use the HMM instead of simply partitioning the data due to the fact that signal can be corrupted by "noise" as in Figure 2.9. The same signal corrupted by noise gives the symbol sequence "bcddddcbaaaaacdddccb" using the same partition as in Figure 2.6.

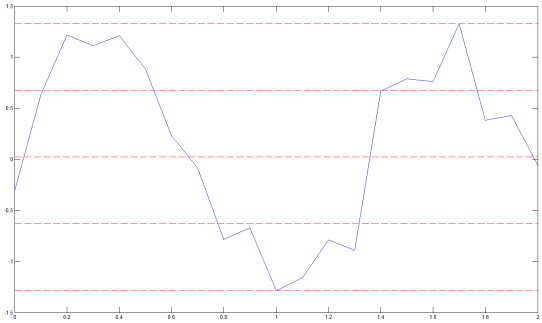


Figure 2.9: Space partitioning of a noisy sinusoidal waveform

The idea is to use the HMM modeling framework as shown in Figure 2.10. By allowing the emission of the hidden states to be random instead of the sharp boundaries defined in the state space partitioning, the model is more robust. However, the complexity of model construction is much higher than state space partitioning.

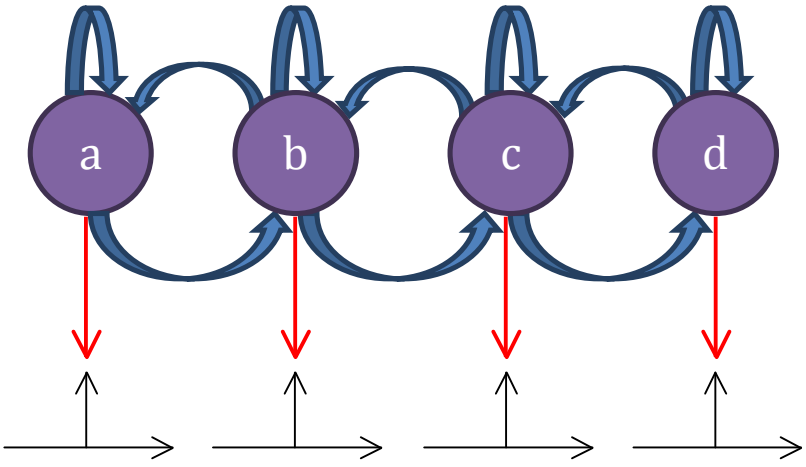


Figure 2.10: HMM for modeling time-series data and partitioning

Hidden Markov Model (HMM) Development: Let

$q_{1:T}$ be a trajectory of the hidden state
$O_{1:T}$ be an observation process

Where
$q_l \in \{1, \dots, M\}\ \forall i = 1,2, \dots, T$
$O_l \in \mathbb{R}$

The observation process $O_{1:T}$ is generated by the trajectory of hidden states, $q_{1:T}$ that is, for each t $(O_t|q_t = l)$ is a random variable with probability distribution f_l. The hidden state is modeled as a Markov chain, so we require the initial probability distribution and state transition matrix as additional model parameters. Thus, model parameters of the HMM include:

M: number of nodes
$\{f_1, f_2, \dots, f_M\}$: emissions (probability distributions) for all nodes
p_1: initial probability for hidden state
A: state transition matrix for the Markov chain

Three basic problems of HMM modeling are

- What is a probability (or Likelihood) of observation sequences given model parameters, $P(O_{1:T}|Model)$
  - This problem is solved by the Forward-backward procedure (Baum)
- What is the most probable trajectory of the hidden state given the observations and model parameters, $P(q_{1:T}|O_{1:T}, Model)$
  - This problem is solved by the Veterbi algorithm
- Determining the model parameters
  - This problem is solved by Baum-Welch method.

For determining the model parameters given the observation sequences, because we are dealing with continuous-valued signals, we choose the HMM with Gaussian emission, e.g., $\{f_1, f_2, \dots, f_M\}$ are M different normal distributions. Next, we briefly mention about model parameter estimation, the general concepts and full detail of the construction of HMM can be found in [9]. For specific information for HMM model parameter estimation when the HMM has Gaussian or Gaussian mixture emission refer to [10-12].

Let
$b_j(O_t) = f_l(O_t)$ denote the value of the distribution function f_l at the point O_t.

We can compute the forward and backward variables for the HMM by
$\alpha_t(j) = \sum_{i=1}^{M} \alpha_{t-1}(i) b_j(O_t) a_{ij}$
$\beta_t(j) = \sum_{i=1}^{M} \beta_{t+1}(i) b_i(O_{t+1}) a_{ji}$

Where
$\alpha_1(j) = b_j(O_1) p_1(j) \quad \forall j = 1, \dots, M$
$\beta_T(j) = 1 \quad \forall j = 1, \dots, M$

Estimation for the q^{th} iteration for the model parameters is given by

$$a_{ij}^{(q)} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij}^{(q-1)} b_j(O_{t+1}) \square_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)}$$
$$\mu_i^{(q)} = \frac{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i) O_t}{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)}$$
$$C_i^{(q)} = \frac{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)\left(O_t - \mu_t^{(q-1)}\right)\left(O_t - \mu_t^{(q-1)}\right)^T}{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)}$$

Where

$$f_i = N(\mu_i, C_i) \quad \forall i = 1, \dots, M$$

We observe that

$$h(x_{1:T}|Model) = \sum_{l=1}^{T} h(x_l|x_{1:l-1}, Model)$$

Due to the fact that our model is HMM has Gaussian emission, $h(x_l|x_{1:l-1}, Model)$ is the entropy of the Gaussian mixture. Therefore, we need to compute the entropy (or bounds) for the Gaussian mixture.

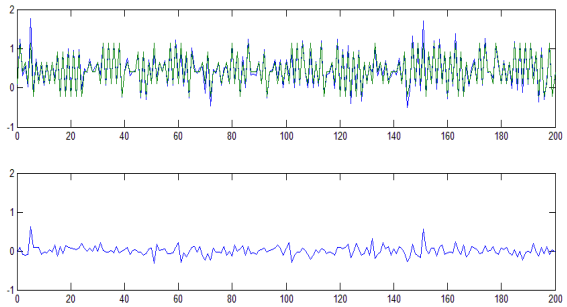


Figure 2.11: Top figure: HMM approximation (blue) and original time series data (green) using the Viterbi algorithm with 6 hidden states. Bottom figure: Approximation error

Entropy Estimation of Gaussian Mixture I: Huber and his colleagues addressed this problem in [13], which provides three computations for estimating the entropy of the Gaussian mixture, including lower and upper bounds. These results are summarized next.

For the Gaussian mixture $p_X = \sum_{i=1}^{M} \alpha_i N(m_i, C_i)$, the differential entropy can be computed by

$$\begin{aligned} h(X) &= \int \left(\sum_{i=1}^{M} \alpha_i N(m_i, C_i)\right) \log\left(\sum_{i=1}^{M} \alpha_i N(m_i, C_i)\right) dx \\ &= \sum_{i=1}^{M} \alpha_i \int N(m_i, C_i) \log\left(\sum_{i=1}^{M} \alpha_i N(m_i, C_i)\right) dx \end{aligned}$$

A Taylor series expansion of $\log\left(\sum_{i=1}^{M} \alpha_i N(m_i, C_i)\right)$ around the mean m_i can be used to estimate the entropy. There are two issues: (1) The number of terms in the Taylor series expansion affects the accuracy of the estimate of $h(X)$, and (2) It is difficult to determine the radius of convergence of the Taylor series. The authors approximate a Gaussian distribution by a sum of Gaussians with smaller covariance. As given by

$$h(X) \approx \sum_{i=1}^{M} \alpha_i \sum_{l=1}^{N_i} \beta_{i,l} \int N\left(m_{i,l}, C_{i,l}\right) \log\left(\sum_{i=1}^{M} \alpha_i N(m_i, C_i)\right) dx$$

For each i and l, they use a Taylor series expansion of $\log\left(\sum_{i=1}^{M} \alpha_i N(m_i, C_i)\right)$ around $m_{i,l}$. Unfortunately, there is no guarantee on convergence or accuracy of this estimation because we have no idea how small the covariance of the Gaussians need to be to accurately estimate $\log\left(\sum_{i=1}^{M} \alpha_i N(m_i, C_i)\right)$ by a Taylor series using only a few terms. In addition, approximating a Gaussian with a Gaussian mixture can be computationally expensive for higher dimension data. With these concerns, we place our focus on estimating upper and lower bounds of the entropy for a Gaussian mixture.

.
The lower bound of the entropy of a Gaussian mixture can be computed by [13]

$$h(X) \geq -\sum \alpha_i \log \sum \alpha_i z_{ij}$$

Where

$$z_{ij} = N\left(m_j, C_i + C_j\right)|_{m_i}$$

Although this lower bound can be easily computed, it will never converge to $h(X)$ and there can be large errors as will be shown later in next section.

In the same paper, they propose a loose upper bound given by

$$h(X) \leq \sum \alpha_i h_{N(m_i, C_i)} - \sum \alpha_i \log \alpha_i$$

which is also easily computed. The drawback of this computation is that the upper bound is inaccurate when individual terms in the mixtures are clustered near to each other. They additionally refine this upper bound by progressively merging two Gaussians in the mixture together and show that for a Gaussian mixture $X \sim \sum_l \alpha_l N(m_l, C_l), \quad h(X) \leq h\left(\tilde{X}\right)$
where

$\tilde{X} \sim \sum_{l \neq i,j} \alpha_l N(m_l, C_l) + \tilde{\alpha} N\left(\tilde{m}, \tilde{C}\right)$
$\tilde{\alpha} = \left(\alpha_i + \alpha_j\right)$
$\tilde{\alpha} N\left(\tilde{m}, \tilde{C}\right)$ has the same mean and covariance as $\alpha_i N(m_i, C_i) + \alpha_j N\left(m_j, C_j\right)$

The upper bound estimation is obtained using the following steps:

Step 1: For $X^{(i)} \sim \sum_l \alpha_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right)$, compute $h_{upper}\left(X^{(i)}\right)$

Step 2: Pick the pair r and s where $r \neq s$ such that

a. $\tilde{\alpha}^{(i)} N\left(\tilde{m}^{(i)}, \tilde{C}^{(i)}\right)$ has the same mean and covariance as $\alpha_r^{(i)} N\left(m_r^{(i)}, C_r^{(i)}\right) + \alpha_s^{(i)} N\left(m_s^{(i)}, C_s^{(i)}\right)$
b. It minimizes distance between $\tilde{\alpha}^{(i)} N\left(\tilde{m}^{(i)}, \tilde{C}^{(i)}\right)$ and $\alpha_r^{(i)} N\left(m_r^{(i)}, C_r^{(i)}\right) + \alpha_s^{(i)} N\left(m_s^{(i)}, C_s^{(i)}\right)$. This criterion comes from Runnalls's work that is based on upper bound of Kullback-Leibler distance [14].
c. $X^{(i+1)} = \sum_{\substack{l \\ l \neq r,s}} \alpha_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right) + \tilde{\alpha}^{(i)} N\left(\tilde{m}^{(i)}, \tilde{C}^{(i)}\right)$

Step 3: Repeat Step 1 and Step 2 until all of the mixtures are merged into a single Gaussian

Step 4: $h_{upper}(X) = \min_i h_{upper}\left(X^{(i)}\right)$

Entropy Estimation of Gaussian Mixture II: As mentioned in the previous section, our main focus is on the calculation of upper and lower bounds. Although the procedure for refining the upper bound given in [13] can be used to significantly improve the upper bound estimation, we found that, by changing the criteria in step 2 we can further improve the upper bound. In the 2nd step of the upper bound refinement, the K-L divergence between the sum of Gaussian and the merged Gaussian can be used as a merging criterion. The best we can do now is to find an upper bound of the K-L divergence using [14], and this turns out to be a better merging criterion. We modify the 2nd step in upper bound refinement procedure given in the previous section by

Step 2 (modified): Pick the pair r and s where $r \neq s$ such that

1. $\tilde{\alpha}^{(i)} N\left(\tilde{m}^{(i)}, \tilde{C}^{(i)}\right)$ has the same mean and covariance as $\alpha_r^{(i)} N\left(m_r^{(i)}, C_r^{(i)}\right) + \alpha_s^{(i)} N\left(m_s^{(i)}, C_s^{(i)}\right)$
2. $\tilde{X}^{(i)}(r,s) = \sum_{\substack{l \\ l \neq r,s}} \alpha_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right) + \tilde{\alpha}^{(i)} N\left(\tilde{m}^{(i)}, \tilde{C}^{(i)}\right)$
3. $X^{(i+1)} = \min_{\tilde{X}^{(i)}(r,s)} h_{upper}\left(\tilde{X}^{(i)}(r,s)\right)$

In order words, we select Gaussian pair that has the lowest upper bound after merging. This criterion further tightens the upper bound from Huber's original work. [13].

Now, we place our focus on the lower bound that can also be very loose in some instances. Before we proceed, note that, for any random variable X such that $X \sim \alpha f(X) + (1-\alpha) g(X) = p(X)$ where both $f(X)$ and $g(X)$ are probability distributions, we obtain

$$h(X) = \alpha\left(h_f + D(f||p)\right) + (1-\alpha)\left(h_g + D(g||p)\right)$$

where

$D(f||g)$ is the K-L divergence between f and g

Because the K-L divergence is always non-negative, we obtain

$$h(X) \geq \alpha h_f + (1-\alpha) h_g$$

where h_f is the entropy of $X' \sim f(X')$ and h_g is the entropy of $X'' \sim g(X'')$.

It is clear that for $X \sim \sum_{i=1}^{M} \alpha_i N(m_i, C_i) = \sum_{i=1}^{M} \alpha_{\square} N_i$, there exist a collection $\{g_i\}$ such that minimizes $\sum \beta_i \varphi(g_i)$ where

$X \sim \sum \beta_i g_i = \sum_{i=1}^{M} \alpha_i N_i$
Each g_i is a Gaussian mixture
$\varphi(\cdot)$ is a lower bound estimate

Finding an optimal collection can be extremely tedious and impractical, so we use the upper bound refinement from [13] as a guide to construct a new lower bound as:

Step 1: For $X^{(i)} \sim \sum_{l \in group1} \alpha_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right) + \sum_{l \in group2} \alpha_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right)$, compute $h_{lower}\left(X^{(i)}\right)$ as follows:

Compute a lower bound for $\sum_l \alpha_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right)$ according to [13]

Compute a lower bound for $\sum_l \beta_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right)$ using the *K-L* divergence

Note: We need to modify the weights of both $\sum_l \alpha_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right)$ *and* $\sum_l \beta_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right)$ *so that they are valid probability distributions*

Step 2: Pick r such that

1. $\tilde{X}^{(i)}(r) = \sum_{\substack{l \in group1 \\ l \neq r}} \alpha_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right) + \sum_{l \in group2 \cup \{r\}} \alpha_l^{(i)} N\left(m_l^{(i)}, C_l^{(i)}\right)$
2. $X^{(i+1)} = \max_{\tilde{X}^{(i)}(r)} h_{lower}\left(\tilde{X}^{(i)}(r)\right)$

Step 3: Repeat Step 1 and Step 2 until all of the mixtures are in $group2$ are converted into a single Gaussian

Step 4: $h_{upper}(X) = \max_i h_{lower}\left(X^{(i)}\right)$

This procedure is very similar to the upper bound refinement given in [13] except the we separate the Gaussian mixture into two groups. For the first group, we use the refinement method from [13] to compute the lower bound. For second group, we use the weighted sum of entropies for each individual Gaussians as given in our modified procedure. Then we use a weighted sum these two terms to compute the lower bound. Now, we select a member from the 1st group such that when we move it to the 2nd group it gives us the highest lower bound. We continue this procedure until all of mixtures belong to the 2nd group and use the highest lower bound we computed as our lower bound estimate

.

Simulation results for upper and lower bounds of Gaussian mixture: For simplicity, we use the mixture of 1 dimensional Gaussians for our testing. We can show that our method further reduces the gap between the upper and lower bound as shown in Figures 2.12 to Figure 2.17.

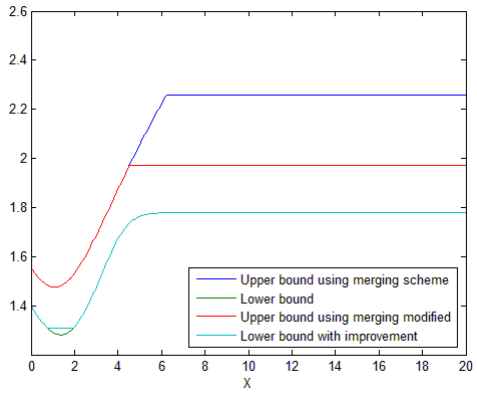


Figure 2.12: Upper and lower bound using the result [13] and our method for Gaussian mixture: $0.2N(0,1) + 0.2N(X, 0.8) + 0.2N(0.75,0.6) + 0.2N(1.5,0.4) + 0.1N(2,0.2) + 0.1N(2.2,0.4)$

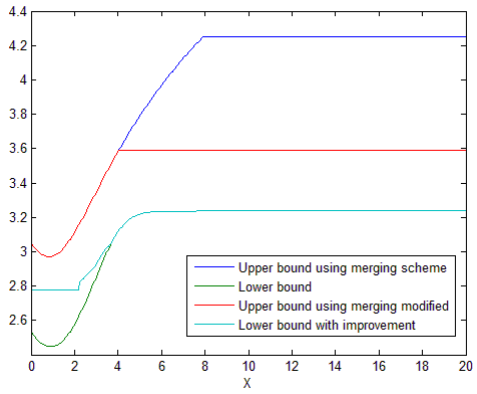


Figure 2.13: Upper and lower bound using the result [13] and our method for Gaussian mixture: $0.2N(0,0.8) + 0.2N(0.5,0.8) + 0.2N(1,0.8) + 0.2N(1.5,0.8) + 0.2N(X, 0.8)$

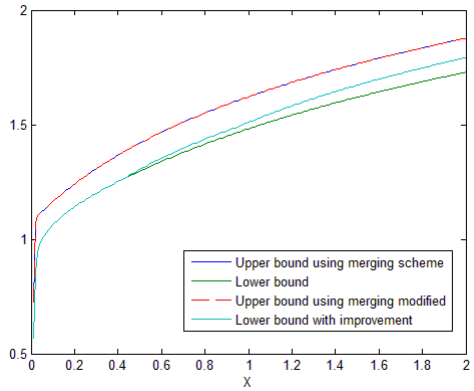


Figure 2.14: Upper and lower bound using the result [13] and our method for Gaussian mixture: $0.2N(0,X)+0.2N(0.5,X)+0.2N(1,X)+0.2N(1.5,X)+0.2N(2,X)$

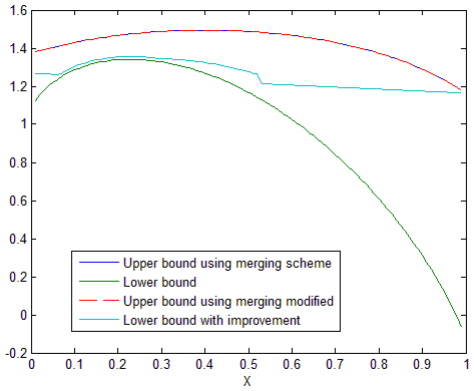


Figure 2.15: Upper and lower bound using the result [13] and our method for Gaussian mixture: $YN(0,0.6)+YN(0.5,0.6)+YN(1,0.6)+YN(1.5,0.6)+XN(2,0.6)$ where $Y=\frac{(1-X)}{4}$

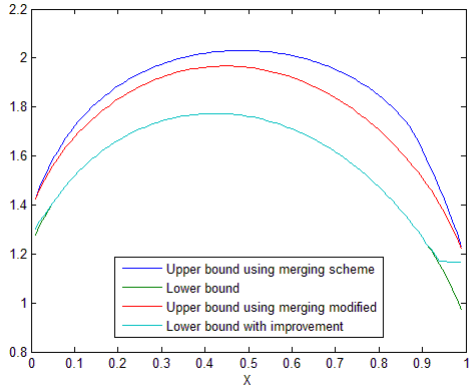


Figure 2.16: Upper and lower bound using the result [13] and our method for Gaussian mixture: $YN(0,0.6)+YN(0.5,0.6)+YN(1,0.6)+YN(1.5,0.6)+XN(4,0.6)$ where $Y=\frac{(1-X)}{4}$

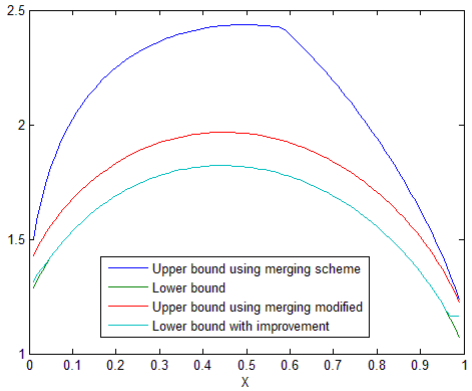


Figure 2.17: Upper and lower bound using the result [13] and our method for Gaussian mixture: $YN(0,0.6) + YN(0.5,0.6) + YN(1,0.6) + YN(1.5,0.6) + XN(6,0.6)$ where $Y = \frac{(1-X)}{4}$

As we can see from these simulation results, in some cases there is not much difference between our lower bound estimate and the method given in [13], e.g., Figure 2.12, Figure 2.16 and Figure 2.17. In contrast, there are cases where there are significant improvements, e.g., Figure 2.15. The same is true for the upper bound in Figure 2.14 and Figure 2.16 that shows no improvement while there are notable improvements in the rest of the results. As a conclusion, we can improve the lower bound and upper bound for calculating the entropy of Gaussian mixtures and make the bounds fairly tight in all of the testing scenarios.

Estimation of Mutual Information Measures: Many, if not most, of the phenomena of interest in nature are continuous. The appropriate information measure for a continuous random variable X is *differential entropy*:

$$h(\mathrm{X}) = -\int_S f(x)\log f(x)dx$$

where $f(x)$ is the density of X. Definitions for joint and conditional entropies, KL divergence, and mutual information can be similarly constructed in a straightforward manner.

Computation of these differential quantities requires the estimation of the associated continuous densities and no generally applicable estimation technique is known. In many cases, the density may not exist or the integral may not exist. Similarly, numerical estimation or quadrature may be intractable or too expensive to provide any utility.

An alternative approach is to discretize the continuous variable and compute the entropy measures in terms of Shannon entropy:

$$H(X) = -\sum_{x\in\mathrm{X}} p(x)\log p(x)$$

where $p(x)$ is the associated probability mass function and X is the alphabet from which the random variable X takes its values. The probability mass function $p(x)$ for a discrete random variable can be obtained empirically (the so-called empirical distribution) via the frequency estimator (i.e. normalized histogram) for instance and thus the computation of the entropy measures can be accomplished in a straightforward manner. Furthermore, these computations are typically more robust than estimates of the analogous differential entropy measures. However, the discretization of a continuous random variable is nontrivial. Furthermore, the alphabet X must be a finite set and thus the random variable must not only be discretized but must also be quantized, an action known to destroy information content (Conant, Laws of Information which Govern Systems, 1975; Feng & Loparo, Active Probing for Information in Control Systems

with Quantized State Measurements: A Minimum Entropy Approach, 1997; Feng, Loparo, & Fang, Optimal state estimation for stochastic systems: an information theoretic approach, 1997). The discretization/quantization of the random variable must therefore be handled with care in the general case. As is the case with direct estimation of differential entropy measures for continuous variables, no generally applicable discretization/quantization technique is known.

Both differential entropy estimation and discretization/quantization of continuous random variables are active areas of research and both approaches to obtaining information measures have pros and cons and ultimately the appropriate approach is application dependent. To that end, we investigate both direct estimation of differential entropy measures and discretization/quantization to identify the appropriate approach or approaches for implementation within out information-theoretic architecture.

Dynamical Systems for mutual information estimation: To investigate the efficacy of mutual information estimation algorithms, several coupled nonlinear dynamical systems (Janjarasjitta & Loparo, 2008) were implemented in simulation. These systems have rich dynamics and the connection strength between the two data stream generated in these models can be adjusted easily. Furthermore, due to the chaotic nature of the systems, different initial conditions give significantly different trajectories. Thus, we can create different scenarios by using random initial conditions with the same interconnection structure between the constituent dynamic systems. This expands the range of test cases that can be examined while still restricting our attention to a small number of systems. In addition, the attention to deterministic, albeit chaotic, systems provides a basis for examining the behavior of probabilistic measures within the context of deterministic systems, which remain an important class of models for many real-world systems and reflects the application of probability theory to noiseless communication channels.

The three systems examined are as follows:

1. Coupled Hénon Maps

$$\begin{cases} x_{n+1}^{(1)} = 1.4 - {x_n^{(1)}}^2 + 0.3x_n^{(2)} \\ x_{n+1}^{(2)} = x_n^{(1)} \end{cases}$$

$$\begin{cases} x_{n+1}^{(3)} = 1.4 - \left(\boldsymbol{C}x_n^{(1)}x_n^{(3)} + (1-\boldsymbol{C}){x_n^{(3)}}^2\right) \\ \qquad\qquad +0.3x_n^{(4)} \\ x_{n+1}^{(4)} = x_n^{(3)} \end{cases}$$

where

- $C = 0,0.04,0.08,\ldots,0.8$
- Observations: $x_{n+1}^{(2)}$ and $x_{n+1}^{(4)}$
- Number of time steps: 10,000

2. Coupled Lorenz Systems

$$\begin{cases} \dot{x}_t^{(1)} = 10\left(x_t^{(2)} - x_t^{(1)}\right) \\ \dot{x}_t^{(2)} = x_t^{(1)}\left(28 - x_t^{(3)}\right) - x_t^{(2)} \\ \dot{x}_t^{(3)} = x_t^{(1)} x_t^{(2)} - \frac{8}{3} x_t^{(3)} \end{cases}$$
$$\begin{cases} \dot{x}_t^{(4)} = 10\left(x_t^{(5)} - x_t^{(4)}\right) \\ \dot{x}_t^{(5)} = x_t^{(1)}\left(28.001 - x_t^{(6)}\right) - x_t^{(5)} \\ \dot{x}_t^{(6)} = x_t^{(4)} x_t^{(5)} - \frac{8}{3} x_t^{(6)} + \boldsymbol{C}\left(x_t^{(3)} - x_t^{(6)}\right) \end{cases}$$

where

- $C = 0,0.1,0.2,\ldots,2$
- Observations: $x_{n+1}^{(1)}$ and $x_{n+1}^{(4)}$
- Number of data points: 10,000
- Numerical Integration: Runge-Kutta with step size 0.01 seconds

3. Coupled Rössler Systems

$$\begin{cases} \dot{x}_t^{(1)} = -0.95x_t^{(2)} - x_t^{(3)} \\ \dot{x}_t^{(2)} = 0.95x_t^{(1)} + 0.15x_t^{(2)} \\ \dot{x}_t^{(3)} = 0.2 + x_t^{(3)}\left(x_t^{(1)} - 10\right) \end{cases}$$
$$\begin{cases} \dot{x}_t^{(4)} = -0.95x_t^{(5)} - x_t^{(6)} + \boldsymbol{C}\left(x_t^{(1)} - x_t^{(4)}\right) \\ \dot{x}_t^{(5)} = 0.95x_t^{(4)} + 0.15x_t^{(5)} \\ \dot{x}_t^{(6)} = 0.2 + x_t^{(6)}\left(x_t^{(4)} - 10\right) \end{cases}$$

where

- $C = 0,0.1,0.2,\ldots,2$
- Observations: $x_{n+1}^{(1)}$ and $x_{n+1}^{(4)}$
- Number of data points: 10,000
- Numerical Integration: Runge-Kutta with step size 0.1 seconds

We remark that the parameters for the test data will be as stated above unless explicitly stated otherwise.

Computation of differential entropy using Hidden Markov Models: Hidden Markov Models (HMMs) provide the basis for one estimation approach as described in our previous report. Several issues have been discovered with this approach that must be addressed. (1) The underflow of forward and backward variables can be remedied by using a log, rather than linear, scale in the internal computations (Mann, 2006). The examination presented here is limited to application of this approach to 1,000 data points. We note that, although studies of this remedy with larger data sets (i.e. more than 5,000 points) are not reported in the literature, preliminary results suggest that internal variables will be corrupted by cumulative error. (2) Another known issue is that the algorithm is sensitive to pathologies in available data and prone to producing degenerate solutions during the construction process. Thus, some of the observed results (i.e. on the Hénon system) are not valid. A related issue pertains to the sensitivity of this approach to initial conditions. That is, different initial conditions often converge to different results. To investigate this issue, sets of 10 different initial conditions were examined for each model order. The model order is determined iteratively by incrementing the model order until the HMM construction process produces invalid parameters due to sensitivity effects. Regardless of initial conditions, solutions converged on low order models, e.g., 1, 2 and 3. As a result, the dynamics of the system were not adequately captured. However, this problem may be numerical rather than algorithmic and further investigation is needed. (3) Another limitation is that this approach relies on the computation of the lower and upper bounds of the entropy of Gaussian mixtures in order to estimate the bounds on mutual information. This is a significant issue as the estimated bounds on mutual information can be nearly 3 times larger than the true entropy bounds. This is not entirely surprising as $\boldsymbol{I(X;Y) = H(X) + H(Y) - H(X,Y)}$. From

Table 2.3, it can be seen that the average estimated bounds of mutual information increases with coupling strength. However, the gap between maximum and minimum upper/lower bound is very large. Since a single realization of the data will be all that is typically available in applications, the uncertainty associated with these computations greatly reduces the significance of the computational results.

Table 2.3: Hénon system data where coupling strength consists of 100 realizations with 1,000 data points

Coupling Strength	Average		Max		Min	
	Upper Bound	Lower Bound	Upper Bound	Lower Bound	Upper Bound	Lower Bound
0	0.388000	0.000000	0.654000	0.000000	0.244000	0.000000
0.04	0.406000	0.000000	0.610000	0.000000	0.252000	0.000000
0.08	0.412000	0.000000	0.691000	0.000000	0.250000	0.000000
0.12	0.375000	0.000000	0.629000	0.000000	0.255000	0.000000
0.16	0.407000	0.000000	0.670000	0.000000	0.222000	0.000000
0.2	0.421000	0.000000	0.686000	0.000000	0.225000	0.000000
0.24	0.386000	0.000000	0.651000	0.000000	0.203000	0.000000
0.28	0.408000	0.000313	0.757000	0.030700	0.191000	0.000000
0.32	0.422000	0.000069	0.744000	0.006910	0.208000	0.000000
0.36	0.447000	0.000000	0.747000	0.000000	0.223000	0.000000
0.4	0.503000	0.028400	0.825000	0.468000	0.273000	0.000000
0.44	0.572000	0.039100	0.962000	0.520000	0.276000	0.000000
0.48	0.628000	0.060900	0.996000	0.550000	0.045600	0.000000
0.52	0.708000	0.135000	1.360000	0.989000	0.078800	0.000000
0.56	0.730000	0.252000	1.950000	1.510000	0.042900	0.000000
0.6	0.675000	0.301000	2.110000	1.840000	0.038000	0.000000
0.64	0.822000	0.598000	2.480000	2.290000	0.090100	0.000000
0.68	1.100000	1.040000	3.270000	3.270000	0.148000	0.076700

0.72	1.460000	1.450000	3.470000	3.470000	0.424000	0.424000
0.76	2.080000	2.080000	3.990000	3.990000	0.570000	0.570000
0.8	2.280000	2.280000	4.030000	4.030000	0.687000	0.687000

Finally, this approach is computationally expensive, particularly when considered in relation to the size of the input data set (i.e. 1,000 points), which is likely to be insufficient for constructing continuous-valued HMMs. Although this problem can be alleviated with more processing capability, increasing the training data will give rise to the cumulative numerical error issue discussed above.

Although, it is believed that this method remains worth investigating in general, the technical challenges associated with the issues delineated above are substantial and an alternative approach to estimating mutual information will be investigated.

Shannon Mutual Information in Time-Series Data: As noted above, the stationarity assumption cannot be justified in the general case. The computation of Shannon entropy and mutual information requires even stronger assumptions, namely that the random variables be independent and identically distributed (i.i.d.) (Cover & Thomas, 2012). Despite the stringency of these requirements with regard to real-world data, these statistics still possess substantial value. That is, if the i.i.d. assumption is violated, the statistic embodied by Shannon entropy may not, strictly speaking, be a measure of information as some axiomatic properties of information may not be preserved. In most cases, however, information-theoretic approaches will still capture relationships between disparate data streams and are more robust to noise and nonlinearity than other available approaches such as those afforded by cross-correlation analyses (Conant, Laws of Information which Govern Systems, 1975). For instance, It can be shown, that if $\boldsymbol{X_{1:T}}$ and $\boldsymbol{Y_{1:T}}$ are independent, then $\boldsymbol{I_{Shannon}(Q_1(X_{1:T}); Q_2(Y_{1:T})) = 0}$. It is also conjectured that, in general, a high level of mutual information implies that two data streams are tightly connected although a low level of mutual information can only be interpreted as a failure to detect inter-connection between two data sets.

Mutual Information and Quantization: Let $\boldsymbol{X} \in \{\boldsymbol{1}, \dots, \boldsymbol{N}\}$ and $\boldsymbol{Y} \in \{\boldsymbol{1}, \dots, \boldsymbol{M}\}$ be two random variables with well-defined distributions and, also, let $\boldsymbol{Z}$ be a quantized version of $\boldsymbol{Y}$, e.g., $\boldsymbol{Z = Q(Y)}$. Clearly, $\boldsymbol{X \to Y \to Z}$. Thus, by the data processing inequality (Cover & Thomas, 2012),

$$I(X;Y) \geq I(X;Z)$$

In other words,

$$I(X;Y) \geq I(Q_1(X); Q_2(Y))$$

where $Q_1(\cdot)$ and $Q_2(\cdot)$ are quantizers.

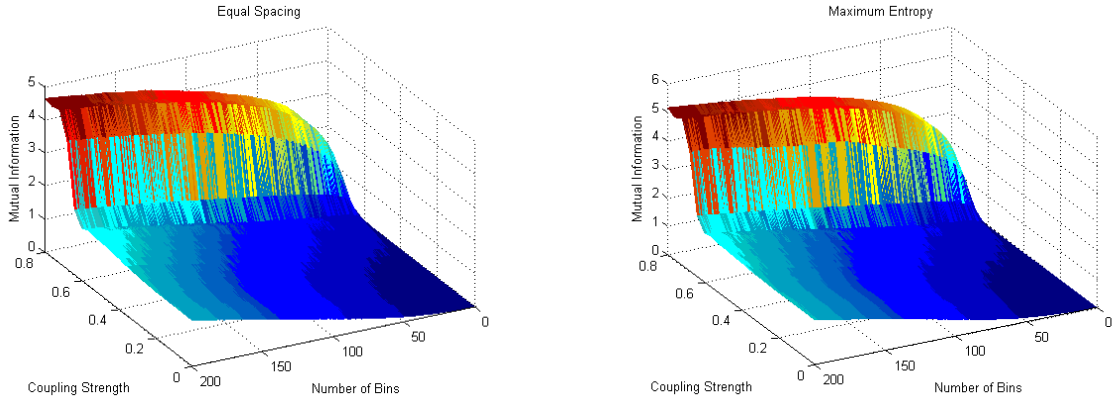
Mutual Information in Shannon sense and independent random processes: Assuming $\boldsymbol{X_k}$ and $\boldsymbol{Y_k}$ are two independent discrete-valued random processes for $\boldsymbol{k = 1, \dots T}$, define $\boldsymbol{\tilde{X}_k}$ and $\boldsymbol{\tilde{Y}_k}$ as two i.i.d. white noise processes as follows:

$$Prob(\tilde{X}_k = i) = \frac{1}{T}\sum_{t=1}^{T} \delta(X_t = i)$$
$$Prob(\tilde{Y}_k = i) = \frac{1}{T}\sum_{t=1}^{T} \delta(Y_t = i)$$

By construction,

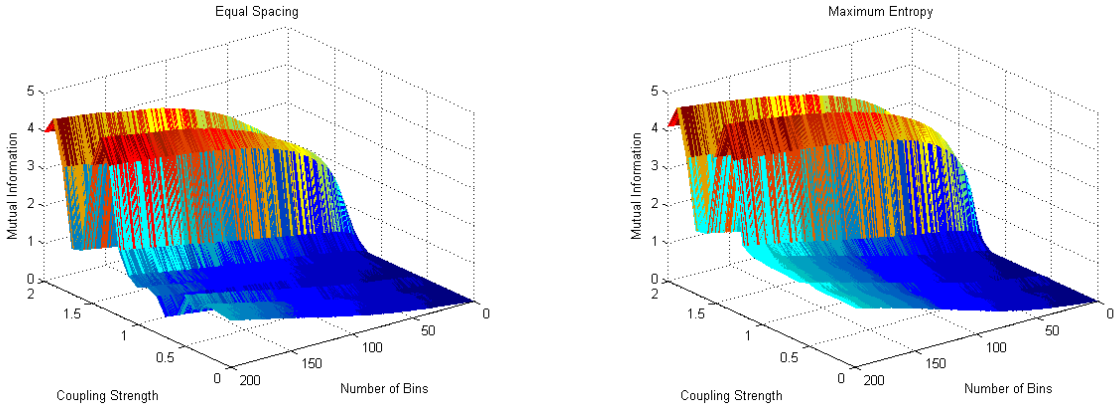
$$I(\tilde{X}_{1:T}; \tilde{Y}_{1:T}) = 0$$

Quantization and the computation of mutual information: Quantization methods have been extensively researched in the field of machine learning in the form of discretization. Note that the idea on using the HMM is also based on quantization, albeit in a different context. The discretization techniques can be broadly categorized into unsupervised and supervised methods (Dougherty, Kohavi, & Sahami, 1995; Kotsianis & Kanellopoulos, 2006). The focus herein is on unsupervised methods due to the nature of our problems. The four unsupervised methods considered here are; equal width partitioning, equal frequency partitioning, unsupervised Monothetic Contrast Criterions (MCC), and k-means clustering. Either the number of bins or the sizes of bins are required for these four approaches. Note that maximum entropy can be used as a modification for equal frequency partitioning to prevent the case that data with the same values are put into different partitions. Equal width partitioning and equal frequency partitioning are used as benchmark methods for evaluating the sensitivity of the estimation and the binning due to the simplicity of their implementations. These four quantization approaches are each applied to the three different systems with the partitions for the associated sequence pairs determined independently.



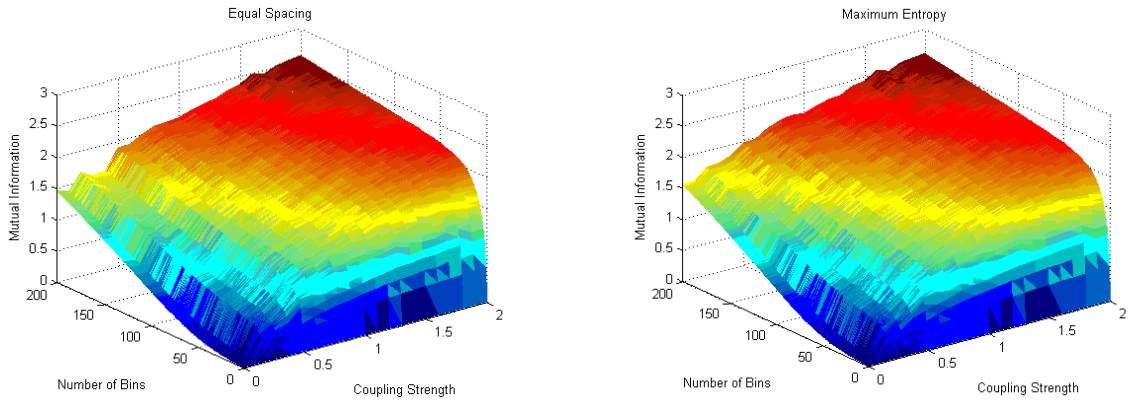
(a) (b)

Figure 2.18: Estimation of mutual information in Shannon sense for the Hénon Map with different coupling strengths and number of bins. The left hand figure uses equal space partitioning and the right hand figure uses maximum marginal entropy partitioning



(a) (b)

Figure 2.19: Estimation of mutual information in Shannon sense for the Lorenz System with different coupling strengths and number of bins. The left hand figure uses equal space partitioning and the right hand figure uses maximum marginal entropy partitioning



(a) (b)

Figure 2.20: Estimation of mutual information in Shannon sense for the Rössler System with different coupling strengths and number of bins. The left hand figure uses equal space partitioning and the right hand figure uses maximum marginal entropy partitioning

From Figures 2.18, 2.19 and 2.20, we can see that maximum marginal entropy partitioning yields slightly higher estimation results than equal space partitioning. The two different quantization methods examined here yield nearly identical curves that may imply that the estimation is relatively insensitive to the partitioning (assuming "reasonable" partitions). With coupling strength fixed, the estimated mutual entropy displays very similar characteristics for all bin resolutions examined, i.e. the corresponding slices of the curve retain a very similar form over all bin resolutions. Further, the estimated mutual information increases nearly monotonically with increasing number of bins.

The estimation of mutual information between coupled Hénon maps and Rössler systems displays a relatively simple structure. As noted above, the basic shape of slices corresponding to fixed coupling strength values are very similar to one another and, similarly, the shapes of slices corresponding to fixed bin counts are very similar over the range of coupling strengths examined, increasing smoothly and monotonically with increasing coupling strength. The investigation of mutual information estimation for coupled Lorenz systems, however, displays far richer behavior than either coupled Hénon maps or Rössler systems. More specifically, the effect of increased coupling strength on the estimated mutual information displays a far more complex structure. The estimation of mutual information for the Hénon map and Rössler system is addressed separately from that for the Lorenz system and is limited to consideration of the two simplest binning methods (i.e. equal width and equal frequency partitioning) since the estimation does not exhibit significant sensitivity to the particular method for partitioning.

Quantization and determination of number of bins: As can be observed from the results presented in the previous section, the basic relationship between mutual information and coupling strength appears to be relatively insensitive to the number of bins. However, the number of bins used in the estimation process does have an effect on the magnitudes of the estimated mutual information, i.e., increasing the number of bins increases the magnitude of estimated mutual information. One way to mitigate this sensitivity to the number of bins is to consider normalized mutual information. The normalization method presented in (Scott, 1979) is applied here to determine the "best" number of bins to be used during the estimation process. The method of Scott (Scott, 1979) assumes a Gaussian i.i.d. random sequence, and the results presented in the previous section strongly suggest that consistent estimation results will be achieved. This expectation will be examined in more detail in the sequel where the estimation performance is evaluated on data from the coupled systems given previously.

Estimating Mutual Information in Shannon sense for time-series data: One attribute of Shannon entropy that is commonly noted is that its value is independent of the order in which the data is considered. That is, the information content (for both self-information and mutual information) is invariant under reordering of the data. While this is often a useful property, it has significant implications regarding the capture of "historical" information, i.e. information pertaining to the evolutionary structure of the data. The ability to capture the evolutionary dynamics of a system is often crucial to properly characterizing dynamical systems (that, by definition, evolve). Several approaches such as examining information rates have been suggested for addressing this deficiency. The approach taken here is analogous to that applied in the computation of auto- and cross-correlation functions, namely considering the value of the information measures as a function of time lag. Given time series data $\boldsymbol{X_{1:T}}$ and $\boldsymbol{Y_{1:T}}$ with the maximum delay, $\boldsymbol{D}$, we compute

$$I^d_{Shannon}\,(X_{1:T}; Y_{1:T}) = I_{Shannon}(X_{1:T-D}; Y_{1+d:T+d-D}).$$

Now, mutual information is computed via

$$\hat{I}_{Shannon}(X_{1:T}; Y_{1:T}) = \max_{d=0,1,\dots,D} I^d_{Shannon}\,(X_{1:T}; Y_{1:T}).$$

The use of a normalized version of mutual information is proposed, where the normalized (lagged) mutual information is estimated via

$$I^d_{nShannon}(X_{1:T}; Y_{1:T}) = \frac{I_{Shannon}(X_{1:T-D}; Y_{1+d:T+d-D})}{H_{Shannon}(X_{1:T-D}; Y_{1+d:T+d-D})},$$

and the mutual information is computed as

$$\hat{I}_{nShannon}(X_{1:T}; Y_{1:T}) = \max_{d=0,1,\dots,D} I^d_{nShannon}(X_{1:T}; Y_{1:T}).$$

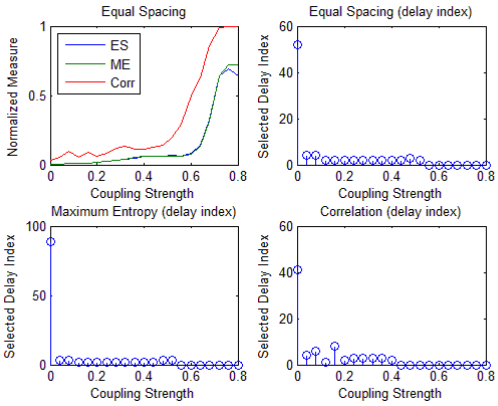
Note that $\hat{I}_{nShannon}(X_{1:T}; Y_{1:T}) \in [0,1]$.

The mutual information estimation results are compared with the linear correlation coefficient, subject to a minor modification:

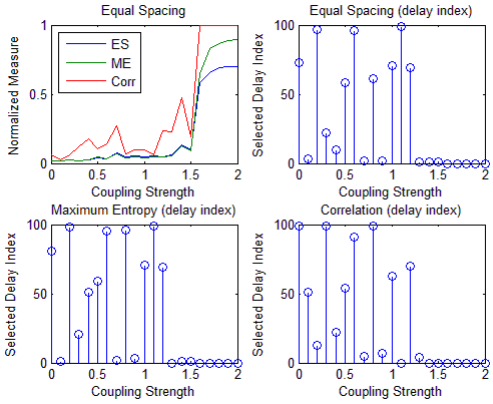
$$\rho^d(X_{1:T}; Y_{1:T}) = \rho(X_{1:T-D}; Y_{1+d:T+d-D}),$$

$$\hat{\rho} = \max_{d=0,1,\dots,D} \rho^d(X_{1:T}; Y_{1:T}).$$

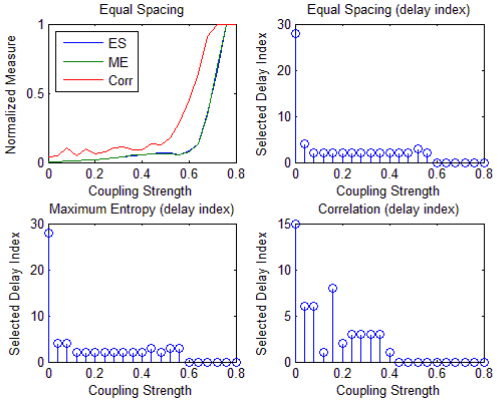
The three systems are used to test mutual information and correlation estimation with a time window, $T = 11{,}000$ steps and a maximum delay, $D = 100$ steps. Additionally, two binning methods, equal space and maximum entropy binning are used to estimate mutual information. The results are shown below. For each case considered, the results are presented in groups of four (4) graphs. The graph on the top-left is the estimation result where the red curve shows the value of the associated correlation coefficient, the blue curve provides the estimated mutual information using equal space partitioning, and the green curve is the estimated mutual information using maximum entropy binning. The remaining three graphs present indicator functions for the delay that yields the maximum value for the mutual information estimation for equal spacing and maximum entropy binning (top right and bottom left, respectively) and for the correlation coefficient (bottom right). The data was collected using the same initial condition in all cases with different coupling strength for each subgroup of figures. The initial conditions were selected at random for a total of twenty different initial conditions. Those shown here have been selected on the basis of possessing exemplary features.



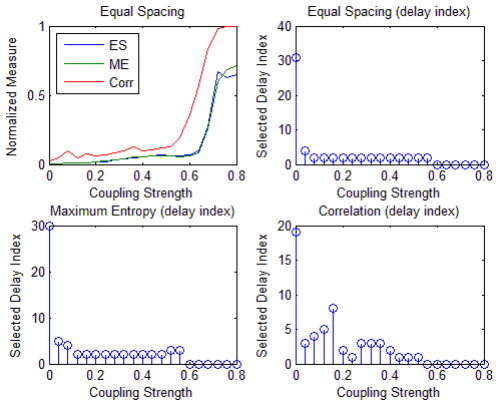
(a)



(b)

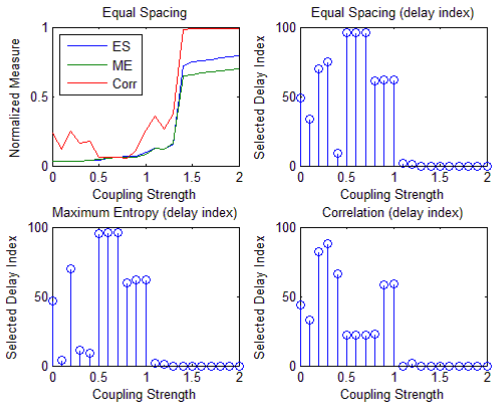


(c)

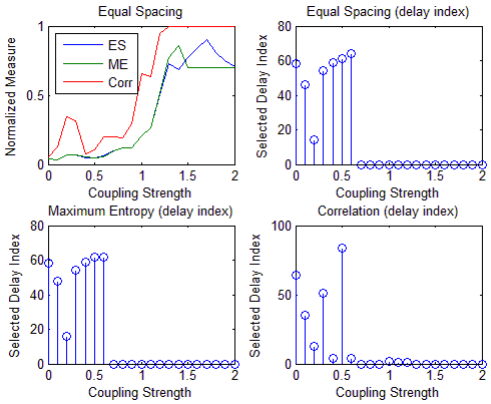


(d)

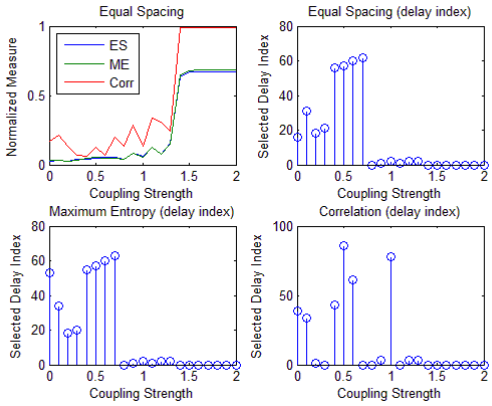
Figure 2.21: Estimation of mutual information in Shannon sense for Hénon Maps with different coupling strengths using the maximum value based on different time-delays



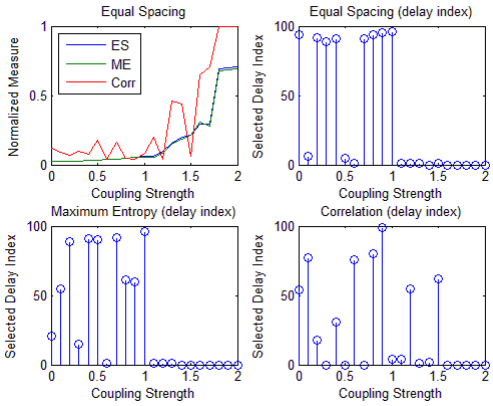
(a)



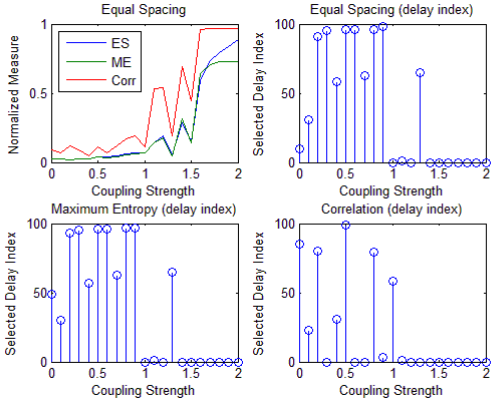
(b)


Equal Spacing
Equal Spacing (delay index)
Maximum Entropy (delay index)
Correlation (delay index)
ES
ME
Corr
Normalized Measure
Selected Delay Index
Coupling Strength

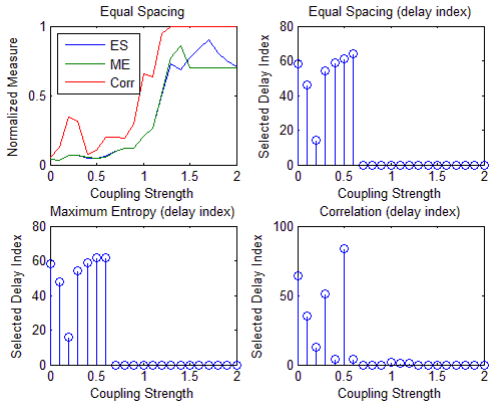
(c)


Equal Spacing
Equal Spacing (delay index)
Maximum Entropy (delay index)
Correlation (delay index)
ES
ME
Corr
Normalized Measure
Selected Delay Index
Coupling Strength

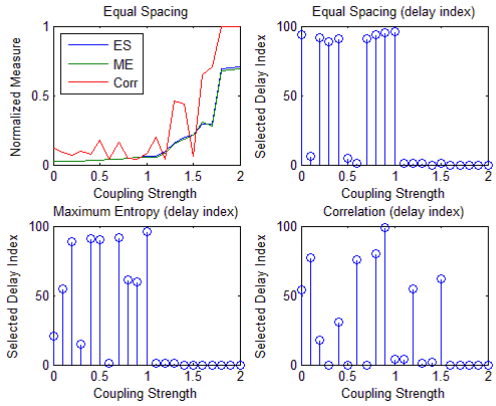
(d)


Equal Spacing
Equal Spacing (delay index)
Maximum Entropy (delay index)
Correlation (delay index)
ES
ME
Corr
Normalized Measure
Selected Delay Index
Coupling Strength

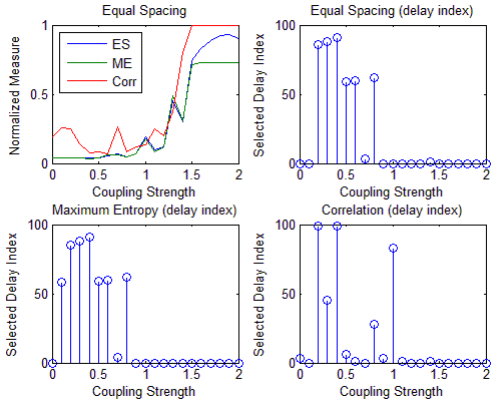
(e)


Equal Spacing
Equal Spacing (delay index)
Maximum Entropy (delay index)
Correlation (delay index)
ES
ME
Corr
Normalized Measure
Selected Delay Index
Coupling Strength

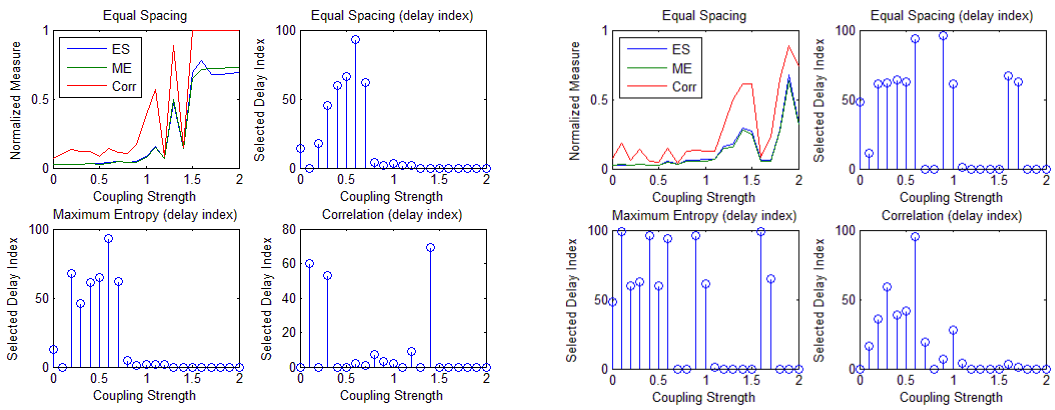
(f)


Equal Spacing
Equal Spacing (delay index)
Maximum Entropy (delay index)
Correlation (delay index)
ES
ME
Corr
Normalized Measure
Selected Delay Index
Coupling Strength

(g)


Equal Spacing
Equal Spacing (delay index)
Maximum Entropy (delay index)
Correlation (delay index)
ES
ME
Corr
Normalized Measure
Selected Delay Index
Coupling Strength

(h)



(i) (j)

Figure 2.22: Estimation of mutual information in Shannon sense for Lorenz Systems with different coupling strengths using the maximum value based on different time-delay

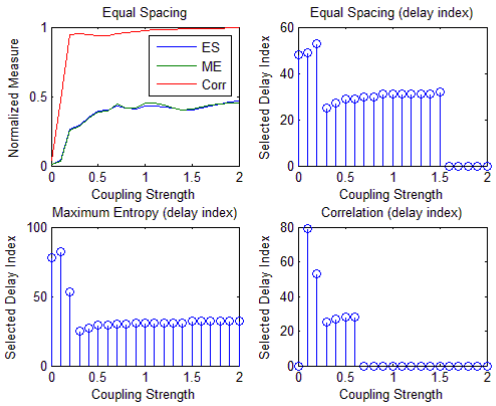


Figure 2.23: Estimation of mutual information in Shannon sense for Rössler Systems with different coupling strengths using the maximum value based on different time-delays

Note first that, in general, the value of the correlation coefficient is higher than that of the normalized mutual information for both binning methods. In a few instances, restricted to application to the Lorenz systems, the mutual information takes higher values than the corresponding correlation coefficient, see Figure 2.22 (g) and (h). For the Rössler system, shown in Figure 2.23, the 20 initial conditions examined produce nearly identical results and exhibit an increscent trend with respect to coupling strength. Although, the trend is clearly nonlinear, both correlation and mutual information can be used to quantify the amount of connection between the two components of the coupled system. For the Hénon Map, the trend is nearly non-decreasing, the only exceptions occurring a low coupling levels where some moderate fluctuation occurs in all cases, depicted in Figure 2.21 (a), (b), (c), and (d), and at high coupling strengths when equal space binning is applied, shown in Figure 2.21(a) and (b), the maximum entropy partitions provide slightly better performance than that offered by equal space binning. Figure 2.21 (d), however, shows that the equal space partition offers slightly better performance at high coupling strength.

Correlation and maximum entropy binning both display positive attributes with respect to binning resolution while, arguably, maximum entropy binning displays the smoothest behavior for these specific systems and as we have seen, these methods display marked sensitivity to initial conditions when applied to Lorenz attractors. However, as shown in Figure 2.22 (a)-(j), while the correlation coefficient displays a primarily

increscent behavior with respect to coupling strength, it also displays significant fluctuations over the range of coupling strengths examined. In particular, the correlation coefficient measure can overestimate the coupling on low coupling strength. In cases where mutual information approaches underperform (i.e. under estimate coupling), shown in Figure 28(i) and (j), the correlation coefficient measure essentially amplifies the associated fluctuations in estimated "correlation strength." Further examining the performance of these approaches applied to the Lorenz system, two of the cases selected show that equal space binning outperforms maximum entropy binning, as depicted in Figure 2.22 (e) and (f). In general, mutual information seems to be applicable to Lorenz systems though, clearly, pathologies can arise in some cases, Figure 2.22 (i) and (j). These pathological cases warrant further investigation. Pending further investigation of these anomalous estimation behaviors, the mutual information measures based on quantized data appears to be more robust than the correlation coefficient measure and more robust and computationally tractable than direct mutual information estimation approaches though further refinement of the quantization methods is needed.

Information Network Discovery and Self-organization: Here, we consider a system consisting of multiple systems (i.e. a System of Systems (SoS)), each possessing multiple sensors, with the individual elements of the system communicating with one another. That is, the disparate components and subsystems influence one another through physical interactions, e.g., heat, vibration, current, and the behaviors of individual elements can be observed in the behaviors of other, physically communicating system elements. It is worth noting that this phenomenon is precisely the core notion underlying observer theory. The key difference here is that, rather than designing an observer to interact with a system of interest, the observers are the other system elements in the overall system.

As noted above, the information-theoretic framework under development has three principle components, illustrated in Figure 2.24. Each of these components is aligned with a stage of the condition/health monitoring process (or, more generally, management and control). The first stage consists of discovering the information geometry of the system, which is discovering the how elements of the system communicate through the physical infrastructure. The key component under this element is focused on developing the basic building blocks for discovering inter-element transmission of information and this effort is addressed in the prequel. In the next stage, these building blocks are used to discover the actual topology of the system's intrinsic communication structure (i.e. the organization of the physically mediated communication between system elements). Note that this intrinsic structure may change with system operating condition as well as with the physical condition of its elements. Further, this element must also reconcile the available data streams with the intrinsic communications structure and the additional observation capabilities that this communication engenders.

It must also be noted that this reconciliation must embed the necessary context. That is, the observations (both directly sensed and indirectly observed through communicating systems) must be valuated, which can only be accomplished within the context of desired operational objectives. Specifically, these observations only have value in so far as they furthers the operation objective of performing health and condition monitoring of system elements. Thus the reconciliation process must properly identify the particular system elements and their associated fault dynamics and include the development of all classification and logic systems for associating data streams with phenomena (e.g., failure modes) of interest.

The final stage focuses on extracting the information from the relevant data streams (as identified during the second stage) and includes the development of all necessary signal processing, estimation, system identification, and feature extraction algorithms as well as all logic for fault detection, diagnosis, and prognosis.

Figure 2.24 (a) depicts a set of four black box elements comprising a generic system. Further, no a priori information is assumed for any element. Each element has multiple sensors associated with it that provide, through the available observations, information pertaining to each of these elements. Physical interactions (communication pathways), e.g., heat, current, vibration transmission paths, between these disparate

elements are represented by the arrow-headed lines connecting the boxes. Information is assumed to flow between system elements and the associated transmission paths define the communication topology where each element is a node as shown in Figure 2.24 (b). The information transmission between these elements is identified and characterized via information measures such as mutual information (sometimes known as information transmission). A principle component of the work is the development of a self-organizing swarm algorithm system for discovering the intrinsic communications topology. This system is based on swarm algorithms for shortest path discovery such as Ant Colony Optimization (ACO) (Bonabeau, Dorigo, & Theraulaz, 1999).

As noted earlier, the system elements also have sensors associated with them. These sensors and their associated observation processes provide the data for inferring the intrinsic communication structure where the mutual information between data streams can be used to illuminate the inter-element communications. With respect to adapting shortest path discovery to network discovery, mutual information relative to a particular data stream forms the basis of a food analog for applying swarm insect foraging behavior a la ACO.

It should also be noted that the communications topology associated with the sensing/instrumentation infrastructure is in all probability not the same as the intrinsic communication topology. It may be advantageous, however, to construct a virtual sensor network with does mirror this intrinsic structure. This is analogous to the so-called topology paradox in computer networking. The physical network is typically a star network (or a hierarchy of star networks) as this reflects the physical reality of building construction and thus the physical reality of network infrastructure. However, this topology is not conducive to providing necessary network capabilities such as, redundancy, fail-over, or the requisite peer-to-peer connectivity. Thus a virtual network is constructed to provide a logical topology that provides the necessary buses, meshes, and peer-to-peer connections.

Further, the potential existence of multiple observers (i.e. other communicating system elements) also provides observational redundancy. Thus a mechanism for verifying observations, detecting failed sensors, or even reconstituting lost sensing capabilities is present. Extrapolating further, these observers also provide a mechanism for constructing “virtual sensors” for observing phenomena that cannot be directly instrumented. The second component of this element is the identification of redundant communications and developing the requisite logic for associating the information sources with the phenomena of interest and for reconfiguring these associations on the fly in response to changing operational conditions, faults, and sensing failures.

Finally, the fault detection and diagnosis is performed at the last stage, Figure 2.24 (c), by incorporating the data streams and information source/phenomena associations identified in the previous stage.

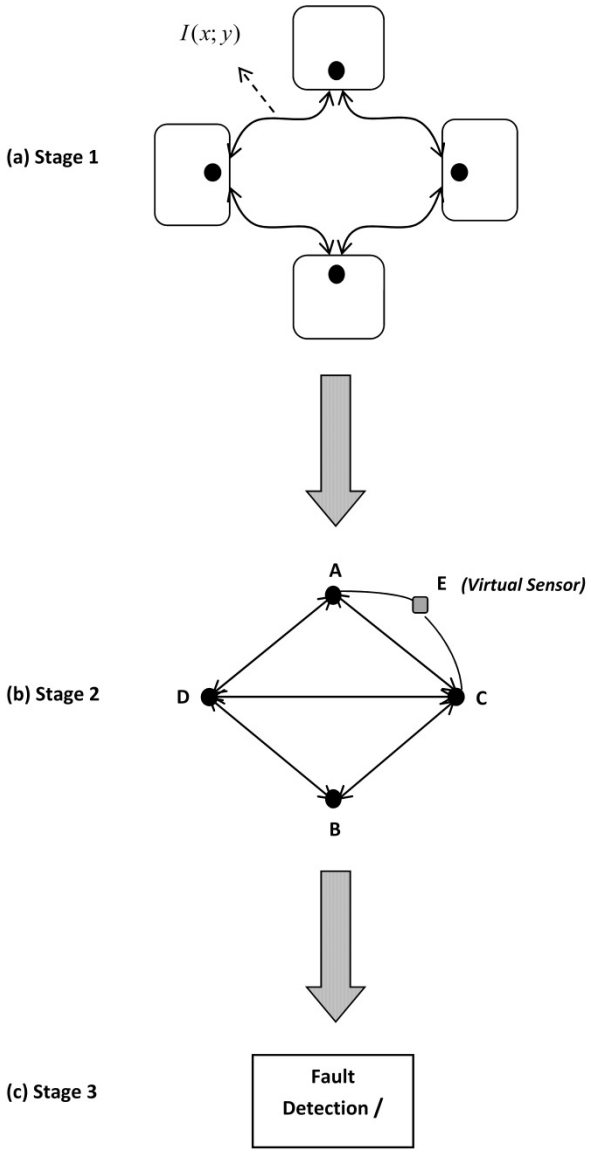


Figure 2.24: Relationships between three main project elements

Biologically Inspired Approaches: The key notion undergirding biologically inspired design is that biological systems display many behaviors, processes, and structures that address issues analogous to those in engineering design. While this approach has had its share of successes it has also had more than its fair share of instances where the underlying analogies have been stretched far beyond their limits. This effort directly addresses this shortcoming by casting a wider net while seeking appropriate analogs in biological systems when appropriate, and relying on existing engineering approaches that are equivalent in function but not in implementation. Thus we adopt a more general notion of what constitutes a biological analog and, if simpler classical design methods are available, they will be employed.

The study of self-organizing systems is based on a broad variety of theories and techniques, however the research on emergent algorithms has largely been restricted to computer science and does not adequately address continuous system. Because much of the work on self-organizing systems has not been available to most computer scientists studying emergent algorithms, the majority of research in emergent algorithms is focused on a discrete representation; swarm algorithms, EAs, and Petri Nets. Artificial neural networks (ANNs), however, have successfully incorporated continuous dynamics as well as introduced design

techniques such as simulated annealing based on results from the more general body of research on self-organizing systems.

The ongoing effort to develop biologically-inspired, distributed algorithms for health and condition monitoring for advanced power plant equipment incorporates a broader class of biological systems, including continuous systems, for inspiration within the context of stochastic systems theory and includes reexamining the underlying principles of emergent system development through a careful study of emergent system design, most notably stigmergy and local autonomy. Stigmergy is a class of sematectonic information, and will be addressed within the context of information theory. Specifically, the substrate upon which these systems live is itself a communications channel and stigmergy is best understood as encoded communication.

This work is based upon reexamining the basic principles identified over the last three decades of research into self-organizing/complex adaptive systems through the lens of stochastic systems theory. The proposed approach seeks inspiration from a more general class of biological systems than are typically considered within the context of biologically-inspired design while retaining the valuable progress made in the study and application of emergent algorithms, such as evolutionary algorithms (EAs), swarm algorithms, artificial neural networks (ANNs), and Petri networks, while augmenting both the basis for inspiration and the collection of analytical and design methodologies available for their implementation.

Swarm Intelligence: Collective intelligence approaches offer an intriguing alternative for communications architecture design. The robustness and flexibility of these approaches offer many advantages over traditional design methodologies including; distributed action, inherent parallelism, simpler agent logic than required for centralized, monolithic solution, inherent task decomposition, flexibility, redundancy, and economies of scale. By applying these approaches, an unprecedented level of configurability and range of capabilities can be obtained through the effective collaboration of intelligent, autonomous software agents.

A large number of multi-agent behaviors have been identified based upon observed behaviors in markets, swarms of social animals (Wilson, 1971; Colorni, Dorigo, & Maniezzo, 1992), human organizations, and evolution. The collection of agent influence algorithms that encompass emergent (self-organizing) behaviors can be classified into three categories (Palmer, Hantak, & Kovacina, 1999): ***indifferent***, (agents interact only indirectly through their effect on the environment, a "hidden" feedback loop known as *stigmergy*), ***agent-critical*** (collection of agents is divided into different "specialist" groups than can have different types and levels of influence over other agents) and ***uniform*** (all agents are identical but able to alter their behaviors in response to changing conditions) algorithms.

A key property of collective intelligence approaches is their ability to emerge the desired global behavior on the basis of local information. Of particular note are synchronization behaviors (Mirollo & Strogatz, 1990; Tyrell, Auer, & Bettstetter, 2006), task allocation behaviors (Bonabeau, Sobkowski, Theraulaz, & Deneubourg, 1996), optimization behaviors (Wodrich, 1996), exploration/mapping behaviors (Bilchev & Parmee, 1995; Dorigo, Maniezzo, & Colorni, 1992), pattern recognition and rule discovery (Zhou & Franklin, 1994; Parpinelli, Lopes, & Freitas, An Ant Colony Algorithm for Classification Rule Discovery, 2002), and sorting/clustering behaviors (Deneubourg j.-L. , Goss, Franks, Sendova-Franks, Detrain, & Chretien, 1991; Parpinelli, Lopes, & Freitas, 2002; Oprisan, Holban, & Moldoveanu, 1996; Kuntz, Layzell, & Snyers, 1997; Lumer & Faieta, 1994).

In order for data to be efficiently distributed throughout the network, nodes must trust and cooperate with each other. This type of environment is ideal for agent-based approaches to information (as opposed to data) routing. Of particular interest here are biologically inspired approaches such as swarm intelligence-based approaches. A significant amount of the work performed in this area is focused upon routing for wireless ad hoc networks wherein the network topology varies in response to changes in the physical organization of network nodes (Di Carlo & Dorigo, 1998; Kolacinski R. M., 2003). The current work offers an

interesting variation on agent-based approaches to self-organizing networks in that the network topology is also self-organizing but evolves based upon the information structure of the data streams and the information processing mandated by the operational needs of the condition monitoring system. The interactions between these disparate self-organizing techniques and the expansion of agent roles from routing to information processing in general offer a rich vein of design approaches to explore.

One of the primary difficulties in the design of information processing algorithms, in particular for ad-hoc sensor networks, is the fact that multiple, often contradictory, objectives must be achieved. For instance, the ability to use stigmergy to deduce much of the desired information can permit much of the network surveillance to be accomplished without increasing communications between nodes. In addition, the different behaviors provide a natural framework for encoding hybrid proactive/reactive information processing algorithms.

The challenges implicit in developing robust and provably reliable agent-based algorithms are daunting. A crucial innovation being developed under the auspices of the current effort is the extension of standard analysis and synthesis tools (Palmer, Hantak, & Kovacina, 1999; Kolacinski R. , 2003) to provide stability, robustness, and reliability analyses within a stochastic context via the thermodynamic formalism (Kolacinski, Kanchanaharuthai, & Loparo, 2011; Kolacinski R. , 2011).

Bounded Autonomy: A crucial element of the ongoing work is a close examination of local autonomy from the perspective of the use of bounded autonomy in complex system design. While stigmergy has deservedly received wide attention in the literature, the focus on this type of sematectonic information has occluded the critical importance of bounded autonomy within complex adaptive systems. Local behaviors are typically examined within the context of trying to understand how they emerge global behaviors of interest, overlooking the crucial nature of bounded autonomy to the organization of complex systems. Bounded autonomy is perhaps more clearly seen by examining physiological behaviors in biological systems (i.e. not swarms). Consider cardio-pulmonary interaction in homeostasis. Heart rate and respiratory rate are not specified as set points, the heart and the lungs are free, within certain operating bounds, to let these rates vary as based on local conditions. The role of the central nervous system is more akin to specifying a goal e.g., blood oxygen level, and the heart and lungs perform as they see fit to accomplish that goal. The autonomy bounds are altered based on operating state, and the variability inherent in these organ systems is purposeful and critical to the health of the overall system. Too little (cardiac arrest) or too much (atrial or ventricular fibrillation) variability are both signs of disease, the variability must be bounded.

The notion of bounded autonomy has been of some interest to the larger computer science/software engineering community, particularly in the context of robotics and multi-agent systems, but has rarely been considered within the context of emergent algorithms, typically restricted to constraints on relationships between agents and between agents in the environment. This approach is formalized in contract programming or design-by-contract programming where “contracts,” consisting of formal interface specifications for software components, are used to cast the definition of abstract data types, preconditions, post conditions, and invariants of software behavior (Mitchell & McKim, 2002). The approach may be valuable for obtaining verifiable software, but has virtually no utility for the design and implementation of emergent algorithms because it lacks any mechanism for adequately considering global behaviors much less ensuring global performance.

Statistical Mechanics and Bounded Autonomy: Autonomy bounds are typically changed in response to varying operating conditions. The local systems are then left to optimize their behavior as they see fit on the basis of local information. There is no guarantee, however, that this piecewise optimization achieves a global optimum with respect to current operating conditions. In fact, the performance of biological organisms strongly suggests that, even if global optima could be achieved, it would be "suboptimal" with respect to the organism's performance over evolving operating conditions. That is, the criteria for optimality lie more in the direction of robustness and adaptability rather than point-wise optimality. This is akin to an athlete "staying on their toes," that is, maintaining a dynamic posture rather than a static one as playing "back on their heels" will likely leave them "flat-footed" and unable to react to game play.

Furthermore, achieving a local optimum may have deleterious effects on a system's ability to respond to changes in operating conditions. To understand why this is so, consider a spin glass (Fischer & Hertz, 1993; Stein & Newman, 2013) which is a type of discrete spin system that is used to model ferromagnetism and forms the basis for many models of social behavior (Barrat, Barthelemy, & Vespignani, 2008). A classical discrete spin system consists of a number of "bodies" arranged on a *d*-dimensional lattice. With each body is associated a spin σ_i that takes the values ± 1 corresponding to the up (↑) and down (↓) states, respectively. The Hamiltonian describing the energy associated with each configuration is given by $\mathcal{H} = -\sum_{i,j} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i$, where J_{ij} is the coupling constant between the *i*th and *j*th spin representing the energy reduction if they are aligned and h_i is the strength of an external magnetic field. When the interaction between spins is restricted to nearest neighbors, this model is known as the *Ising-Lenz model* (Huang, 1987). A spin glass is a spin system that behaves like a glass, that is a spin system that is unable to attain a stable ordered configuration and instead moves between, potentially, infinite numbers of metastable configurations. In order for a spin system to become glassy it must be "frustrated (Toulouse, 1977)," that is, its configuration must be such that a body's nearest neighbor interactions are opposites. As each body strives to be aligned with its neighbors, it is frustrated as it can only align with one and will be out of alignment with the other neighbor. Recalling that the equilibrium configuration at $T = 0$ corresponds to the minimum energy configuration (i.e. an optima), the optimum (minimum energy) configuration for a one-dimensional lattice is the periodic chain ↑↑↓↓↑↑↓↓ ⋯. Clearly this configuration represents the minimum energy configuration for aligned nearest neighbor pairs but not for misaligned nearest neighbor pairs.

As this configuration clearly has a regular structure, frustration is not sufficient to engender glassy behavior. In addition, the spin system must be disordered (or at least ordered in a specific way (Marinari, Parisi, & Ritort, Replica field theoy for deterministic models: binary sequences with low autocorrelation, 1994; Marinari, Parisi, & Ritort, Replica field theory for deterministic models (II): a non random spin glass with glassy behaviour, 1994; Bouchaud & Mezard, 1994). Specifically, the system must be (dis)ordered in such a way that, as temperature is decreased, the minimum cannot be achieved (or only with probability 0) so that, rather than achieve the minimum, the system reaches one of a set of metastable state. At a given temperature, the configuration will fluctuate about this frozen disordered state, occasionally transitioning to another, similar state. These transitions occur over a wide range of time scales and it has been shown that this transition behavior results from a collection of infinitely many hierarchically organized metastable states (Palmer R. G., 1982).

This inability to descend into the potential well associated with the minimum energy configuration is precisely analogous to the situation described above wherein a system resides at local minima for a particular operating condition. The various metastable local minima are separated from one another by "potential walls' though, as they are not as high as those associated with that of the global minima and are therefore more easily surmounted. As noted above, these local minima devolve from a hierarchy of metastable minima, thus it is far easier to move from one configuration to another within the set of metastable local minima, both at a given temperature (operating condition) and at different temperatures (operating conditions) than it is to move from the global minima to a local minima, or, more pertinently, between global minima at different temperatures (operating conditions).

The connection between the stochastic behavior of spin glasses and optimization is well known and has in fact produced a well-known technique for optimizing complex systems such as artificial neural networks, namely the technique of *simulated annealing* (Kirkpatrick, Gelatt Jr., & Vecchi, 1983). In simulated annealing, Monte Carlo dynamics are combined with "slow cooling," reflexive of the fact that, in spin glasses, if the temperature is not zero, the system will explore multiple minima of the energy surface. If the cooling rate is sufficiently slow, the system should find itself in the minimum energy "ground state" as the temperature approaches zero. The success of this approach is highly dependent on the time scales associated with the transitions between the local minima. This is germane to the current consideration of determining optima with respect to multiple objects for multiple operating conditions. In cases where the time scales are too large (relative to the cooling rate), the system may become trapped in a local minima whenever the potential well associated with the global minima is too narrow to be discovered in a realistic computational time. However, for many systems, it can be shown that "equally good" ground states exist (Mezard, Parisi, & Virasoro, 1986) and that good approximations to global minima can readily be found.

This demonstrates two critical aspects of the problem at hand: 1.) the determination of the absolute minima for a given operating condition may well be beyond computation in a reasonable time frame but may be well approximated by local minima, and 2.) the transition between minima, particularly at different operating conditions, may be greatly simplified by not steering the system into the absolute minima associated disparate operating conditions thus improving the flexibility and robustness of the system. Evidence suggests that glassy behavior is precisely the strategy that biological systems have taken to endow the organisms, collectives, and subsystems in order to provide the flexibility and adaptability necessary to survive in complex, evolving, multi-objective environments and the restriction to something very much like the set of metastable local minima is the form of bounded autonomy favored by biological systems.

This strongly suggests the utility of disorder (randomness) in system "optimization" over a wide range of operating conditions. The analysis of spin glasses also offers another practical lesson for the analysis of disordered systems that will be exploited in the example to follow. Different types of disorder are more tractable for analysis and thus provide a better basis for the design and analysis of complex adaptive systems such as emergent algorithm systems. Broadly speaking, disorder can be *annealed* wherein the relationships between system elements, described by J_{ij} above, are time varying, random variables and *quenched* wherein these relationships are constant. Somewhat counter-intuitively, annealed disorder is more tractable than quenched disorder. Annealed disorder can be analyzed in a much simpler fashion as the average over disorder can be exchanged with the statistical average (Badi & Politi, 1997). This suggests that, for practical purposes, it may be advantageous to specify certain system parameters probabilistically rather than as fixed constants.

Exemplary System Analysis: The underlying design philosophy and its application to the analysis and design methodologies are elucidated via an exemplary application to the development of a distributed, agent-based coordination system for a coupled system of oscillators. The oscillators are simple analogs of cyclic processes that, in the context of an overall system, must be synchronized. Cyclic processes are ubiquitous in power generation plants, and could represent, for example, turbine generator dynamics, pump dynamics, bearing orbits, or circadian generation rhythms. The need to synchronize these systems is also critical within the context of power generation: chilled water supply, pulverized coal supply, and steam production rates must be synchronized with the power generation cycle and chemical processing cycles must be synchronized for the control of chemical byproducts of generation.

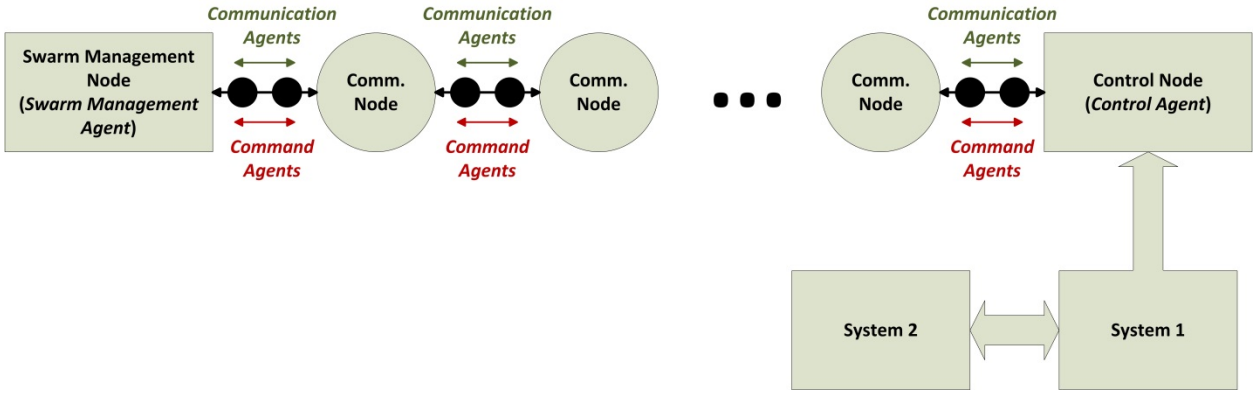


Figure 2.25: Schematic of Distributed Swarm Agent-based Coordination System

The exemplary system is depicted schematically in Figure 2.25. The system is composed of two coupled oscillators, a control node, a collection of communications nodes, a (swarm) agent management node, a swarm management agent, a control agent, a collection (swarm) of communication agents, and a collection (swarm) of command agents. The agents reside on a substrate provided by the nodes (that possess the computation/processing hardware). The Swarm Management Agent and Control Agent are permanently resident at the Swarm Management and Control Nodes, respectively. Similarly, Node Agents are resident at each of the communication nodes. The communication and command agents are allowed to "traverse" the network and may, at any given time, be resident at any of the nodes. From an implementation perspective, the Swarm Management Agent, Control Agent, and Node Agents are the software (at least a portion thereof) running on their respective nodes that are assumed to possess processing capabilities commensurate with their roles. The Communication and Command Agents are specialized message packets requiring limited processing capability and thus may "visit" any node possessing a modicum of processing (and an attendant agent) such as that typically resident on switches and routers.

The two linear oscillators are compliantly coupled and the Control Agent is able to observe the response of System 1 and, based upon this observation, must deduce the state of synchrony between systems 1 and 2 and, if necessary, provide a control input to maintain this synchrony. This control input is introduced through the inter-oscillator coupling. The linear system, assuming unit injection of the control input is:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -(1+\alpha) & \alpha & 0 & 0 \\ \alpha & -(1+\alpha+\Delta) & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},$$

$$y = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix},$$

where α is the stiffness coupling rate and Δ is a coupling asymmetry. Initially, $\alpha = 5$ and $\Delta = 0$. At a given instant, $\tau = 35$ seconds, Δ is set equal to an i.i.d. Gaussian noise process with zero mean and a variance of 2.5, representative of a fault induced behavior. Note system realization is controllable and observable.

As the purpose of this system is coordination rather than low-level control, a simple strategy is pursued here. Rather than formulate a feedback control law, a feedback mechanism patterned after the coordination

mechanism used to model cardiac pacemaker synchronization and neuronal recruitment (Pikovsky, Rosenblum, & Kurths, 2001) will be used, namely a feedback transformation of the oscillators into a pair of coupled Van der Pol Oscillators. The transformed state equation takes the form:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -(1+\alpha) & \alpha & 0 & 0 \\ \alpha & -(1+\alpha+\Delta) & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \varepsilon \begin{pmatrix} 0 \\ 0 \\ (1-x_1^2)x_3 \\ (1-x_2^2)x_4 \end{pmatrix},$$

Where ε is the weight of the nonlinear perturbation. The parameter ε is determined by number of Command Agents present at the Control Node and thus is controlled by the swarm of Command Agents. The role of the Control Agent here is restricted to computing an estimate of the differential entropy of the observed output and broadcasting it to any Communication Agents that may be resident at the Control Node, determining the number of Command Agents resident at the control node and computing a value for ε that is proportional to this number and, if in 'active' mode, passing this value on to the feedback mechanism.

The Control Agent is initially in 'passive' mode, where no synchronization mechanism beyond that resident in the mechanical coupling of the oscillators is active, and will stay there until it receives a message, via a Communications Agent from the Swarm Management Agent instructing it to switch into active mode. The overall swarm logic is managed by the Swarm Management Agent which accepts messages from any Communications Agents resident at the Swarm Management Node and, based upon the information obtained from these measurements, implements appropriate changes to the swarms (of Communications Agents and Command Agents) size (by spawning or absorbing agents of a given type) or their behavior by sending instructions (via the Communications Agents) to the other agents.

In the current implementation, the Control and Node Agents have preset modes. In keeping with the simple example, these modes are simply 'passive' and 'active' and the corresponding change to their behaviors devolves from their current mode. The assignment of behavior devolving from their mode is not, however, predicated upon a direct mapping to a fixed set of parameters determining behavior. Rather, the identification of a particular mode is mapped to bounds on autonomy. Specifically, the behaviors of the agents are specified probabilistically. For example, the Communications and Command Agents traverse the communications path via random walks and the behavior is specified by the corresponding transition (move left or right) or sojourn (stay) probabilities. Mode switching does not direct the selection of a particular set of transition or sojourn probabilities. Instead, the ability of the individual agents (Swarm Management, Control, or Node) to select their own set of probabilities is bounded, either by restricting the support of the probability density functions over the transition and sojourn probabilities or by specifying conditioning on these probabilities based upon the transition and sojourn probabilities of the neighbors of each agent.

Several aspects of this implementation of bounded autonomy warrant further discussion. First, as no transition or sojourn probabilities are hard coded, the individual agents retain far more flexibility in responding to their own local condition. This is critical as a real-world implementation will be subject to many localized phenomena including communications traffic congestion that will likely vary from node to node. While the example above assumes a very simple and fixed topology, any realization of a self-organizing health and condition monitoring system will have a varying topology, if for no other reason than faults can also occur in communications networks. Under the information theoretic paradigm employed by this sensor network, the communication topology is adapted in response to the geometry of the information sources and flows that vary with changes in plant operating conditions (e.g., environment, demand) and equipment (e.g., balance of plant, generating, and instrumentation equipment) health and condition. These agents can thus be endowed with a simple kernel for generating variates and then left to their own devices to select appropriate probabilities.

Second, the general definition of bounded autonomy, particularly that of defining bounds in terms of neighboring transition/sojourn probabilities, provides a straightforward mechanism for employing stigmergic communications. Classical implementations of swarm algorithms implement stigmergic effects through an explicit adaptation of transition probabilities based upon the quantity of sematectonic information (i.e., number of swarm members or concentration of pheromone) (Deneubourg J.-L. , Goss, Pasteels, Fresneau, & Lachaud, 1987; Dorigo M. , Learning by Probabilistic Boolean Networks, 1994; Bonabeau, Dorigo, & Theraulaz, 1999; Englebrecht, 2005). That is, the transition probabilities are a deterministic function of the relevant quantity of sematectonic information rather than the more general autonomy bounds considered here. In addition to enforcing a more stringent constraint on behavior, it also renders the stochastic behavior of the swarm system less tractable to analysis as it is a use of quenched disorder.

Finally, in contradistinction to the probabilistic description commonly employed for describing swarm behaviors, the design here utilizes annealed disorder that, as noted in the discussion of spin glasses above, makes the stochastic analysis of the swarm behavior far more tractable. In particular, the autonomy bounds (on specification of the transition/sojourn probabilities), are explicitly designed to provide the desired performance with respect to a stochastic performance criteria, namely the mean first passage time (MFPT). The MFPT is the mean time required for a trajectory (random walk here) to cross a set boundary. Using stochastic systems analysis, the MFPT for a randomly walking agent can be explicitly determined in terms of the distributions over transitions (Murthy & Kehr, 1989). In other words, the proposed methodology provides a means of ensuring an average "communication" lag from one end of the network to another over all agents.

This has several important consequences: 1.) swarm behaviors can be specified with respect to explicit performance bounds and adapted to accommodate changes in conditions and configuration, 2.) the specification of behaviors can be accomplished in an entirely decentralized way so that swarm management need not require a centralized controller, and 3.) the stochastic analysis of diffusion, the ensemble behavior of random walk trajectories, on a one dimensional lattice has been thoroughly examined and significant results connecting random walk behaviors to diffusion constants, conductivity, etc. are available (Haus & Kehr, 1987; Havlin & Ben-Avraham, 1987; Alexander, Bernasconi, Schneider, & Orbach, 1989; Derrida & Pomeau, Classical diffusion on a random chain, 1982; Derrida, Velocity and diffusion constant of a periodic one-dimensional hopping model, 1983; Bernasconi & Schneider, 1982). This connection to physical constants such as conductivity is critical as it provides an immediate mechanism for designing algorithms for more general topologies. For example, through the connection to conductivity, the design of swarm behaviors over a three-dimensional network can be accomplished via the standard theory of resistive circuits.

In this example, the behavior of the communications agents does not change with the switch from active to passive. Rather, the desired communication rates are held constant. The transition/sojourn probabilities associated with each node are assumed to be independent at any node j, and we denote the right transition probability by p_j and the left transition probability by q_j. The only operant constraint on transition probabilities is that $p_j + q_j = 1$. Defining $\alpha_j = (1 - p_j)/p_j$ and letting $\langle \alpha_j \rangle$ denote the average of α_j over $\rho(p)$, the probability density function over right transition probabilities, we have the following expression for the MFPT;

$$\left\langle \bar{t}_{0,N} \right\rangle = \frac{1+\langle \alpha \rangle}{1-\langle \alpha \rangle} N + \frac{\langle \alpha \rangle^2 + \langle \alpha \rangle}{\left(1-\langle \alpha \rangle\right)^2}\left(\langle \alpha \rangle^N - 1\right)$$

Where N is the number of nodes in the communication path. In the passive mode, Command Agents behave similarly, where the general idea is to have some Command Agents diffused throughout the network for immediate availability. When a deleterious change is detected, in this case by an increase in differential entropy beyond an acceptable threshold, the Swarm Management Agent begins to generate Command Agents via a Proportional Derivative rule and passes a message via the Communication Agents to the node

agents to switch into active mode which causes the Node Agents to condition their associated left transition probabilities so that they are correlated to their left neighbor's right transition probabilities via $\xi_i = q_j / p_{j-1}$. Letting $\langle \xi \rangle$ denote the average correlation coefficient over $\rho(p)$, we obtain the following expression for the MFPT;

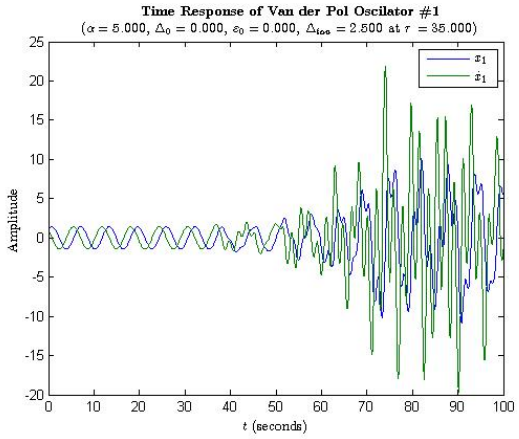
$$\left\langle \bar{t}_{0,N} \right\rangle = \begin{cases} \left\langle p^{-1} \right\rangle \left\{ 1 - \left\langle \xi \right\rangle \right\}^{-1} N, & \left\langle \xi \right\rangle < 1 \\ \left\langle p^{-1} \right\rangle N^2 / 2, & \left\langle \xi \right\rangle = 1 \\ \dfrac{\left\langle p^{-1} \right\rangle \left\langle \xi \right\rangle}{\left(1 - \left\langle \xi \right\rangle \right)^2} \left\langle \xi \right\rangle^N, & \left\langle \xi \right\rangle > 1 \end{cases}$$

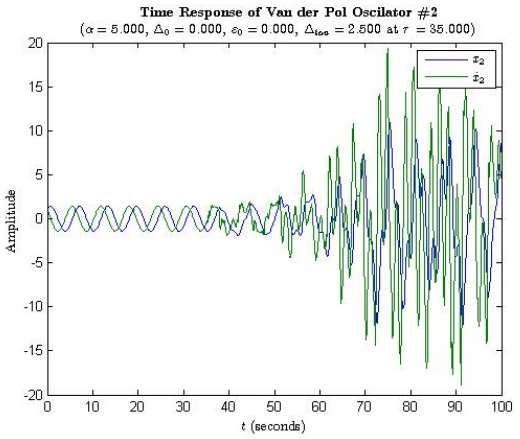
With the MPFT specified, the swarm size can be determined using a confidence level type of analysis.

Illustrative Results: The system described above was simulated in Matlab over a period of time of 100 seconds. The discrete dynamics of the swarm system is updated every tenth of a second. At $\tau = 35$, a noise disturbance is injected into the system as an asymmetry in the coupling term that, in addition to significantly perturbing their oscillatory behavior as shown in Figure 2.26 for the case where the swarm coordination system is inactive. In addition to perturbing the oscillatory behavior, this disturbance also destroys the synchronization between the two systems as shown Figure 2.27 (a). The loss of regularity in the systems' responses also manifests as an increase in entropy of the observed signal, as shown in Figure 2.29 (b) when compared with the differential entropy associated with the unperturbed system shown in Figure 2.29 (a). Also note that the differential entropy estimator begins to settle at approximately 10 seconds and has settled within 30 seconds. It should be noted that, in this context, differential entropy might not be the best information measure to examine in this context as quantization and sampling effects pollute the estimate. While other measures such as Kullback-Leibler divergence may offer a better measure in this case, differential entropy is sufficient for our purposes, which is to detect a significant change in synchrony between the two systems.

The addition of a swarm agent-based coordination system provides immediate improvements in both the time response of the system, shown in Figure 2.30 (a) and (b), and their synchrony Figure 2.31 (a). These improvements are reflected in the estimated differential entropy, shown in Figure 2.31 (b).

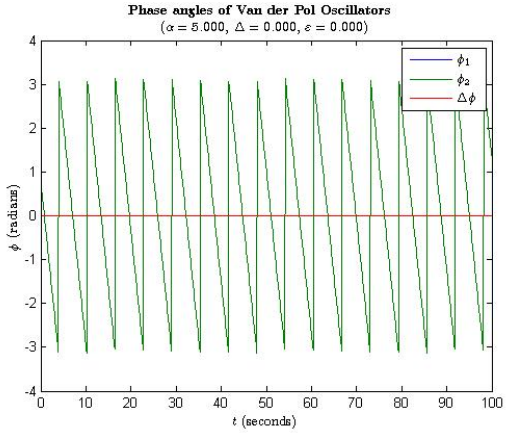
The Communication Agent and Command Agent distributions for the inactive and active coordination cases are shown in Figure 2.32 and Figure 2.33, respectively. As can be seen, the distribution of Communication Agents is evenly distributed across all nodes in both cases. While the collection of Communication Agents is initially concentrated at node 1, the Swarm Management Node, it rapidly disperses throughout the network. In comparison, the behavior of the Command Agents is very different. In the inactive case, the Command Agents behave very similarly to that of the Communication Agents, initially concentrated at the Swarm Management Node. However, in the active case, the Command Agents (and the Swarm Management and Control Agents) behave very differently. At approximately 40 seconds, the behavior clearly shifts as the node agents switch into active mode. The Swarm Management Agent begins to spawn more Command Agents, reflected by the peak appearing at this time at node 1. The agents quickly begin to traverse to the Control Node, reflected in the ridge extending from node 1 to node 12 (Control Node) and the sojourn probability at the Control Node increases as reflected in the peak forming at Node 12.

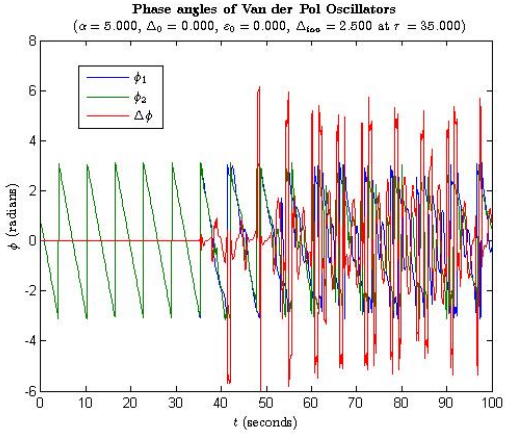




(a) (b)

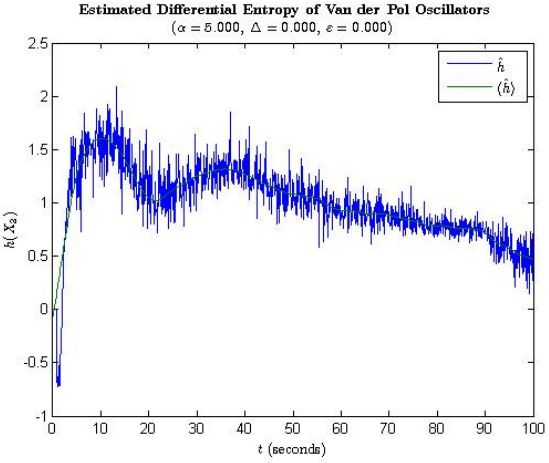
Figure 2.27: Time response of coupled oscillators: (a) System 1 and (b) System 2

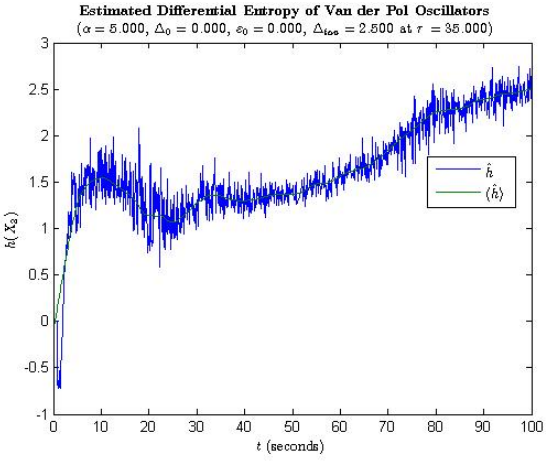




(a) (b)

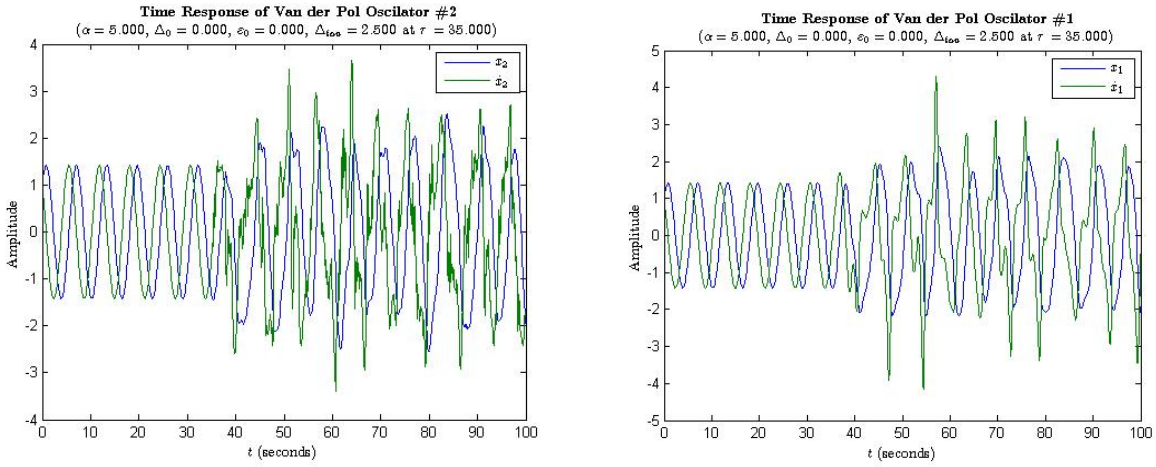
Figure 2.28: System Phase angles and relative phase angle for (a) unperturbed system and (b) perturbed system





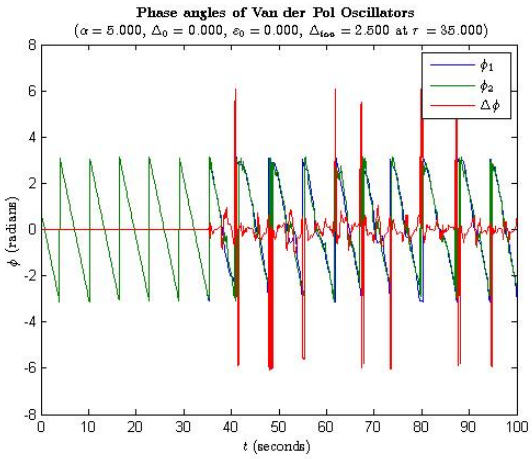
(a) (b)

Figure 2.29: Differential entropy estimate for (a) unperturbed system and (b) perturbed system

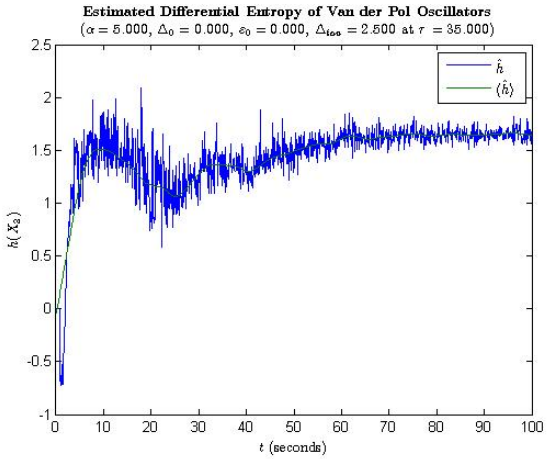


(a) (b)

Figure 2.30: Time response of oscillators with active coordination system: (a) System 1 and (b) System 2

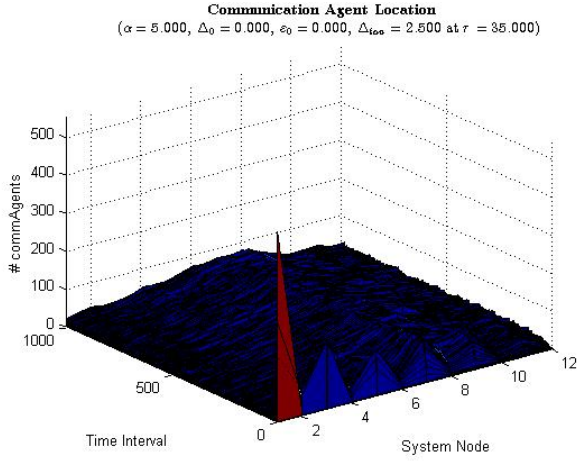


(a)

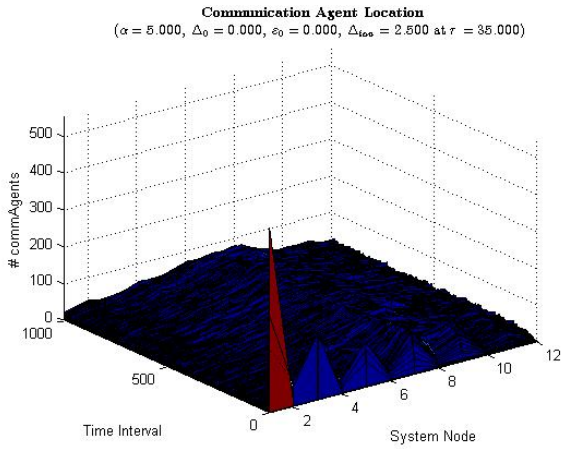


(b)

Figure 2.31: System performance measures: (a) Phase angle and relative phase and (b) Estimated differential entropy

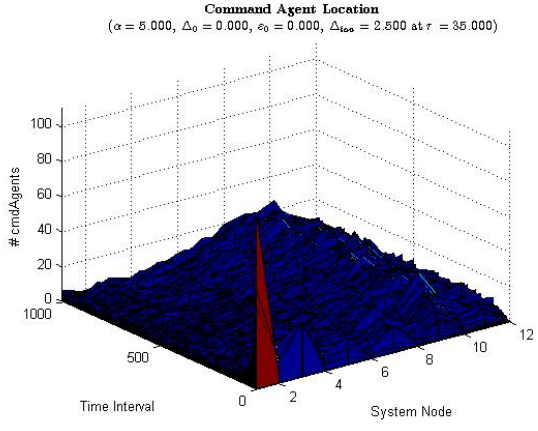


(a)

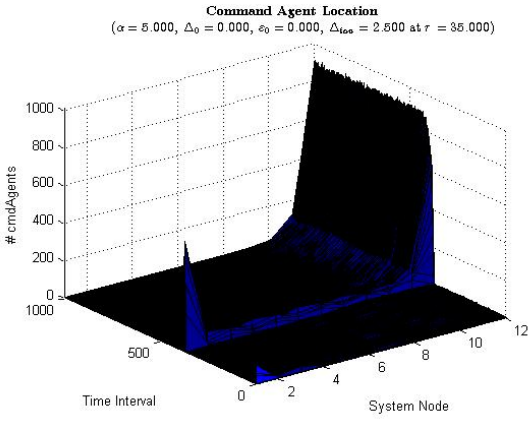


(b)

Figure 2.32: Communication Agent Distribution for (a) Inactive coordination system and (b) Active coordination system

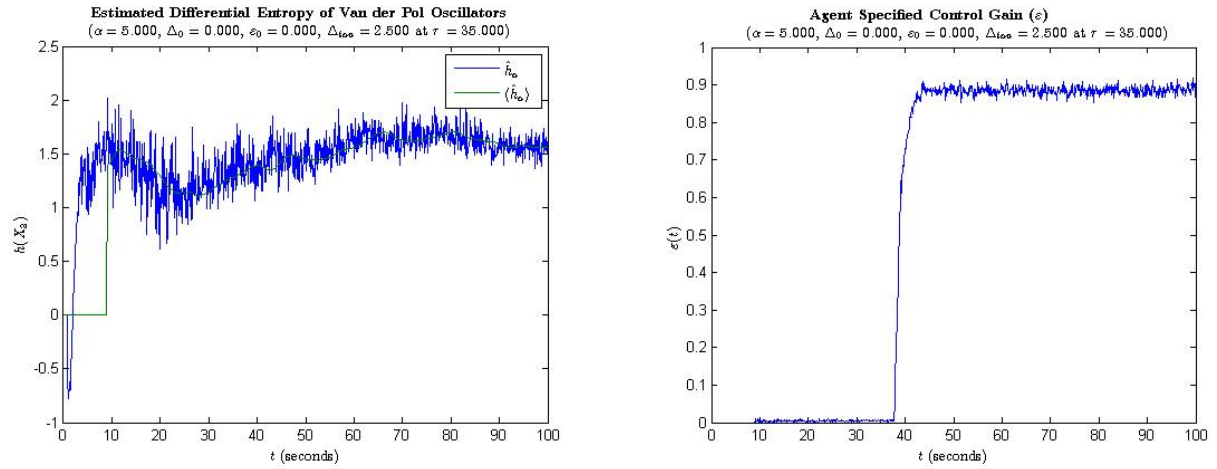


(a)



(b)

Figure 2.33: Command Agent distributions for (a) Inactive coordination system and (b) Active coordination system



(a) (b)

Figure 2.34: (a) Control Agent control input and (b) Swarm Management Agent differential entropy estimate

Figure 2.34 (a) shows the control input produced by the Control Agent and Figure 2.34 (b) shows the estimated entropy as reconstructed by the Swarm Management Agent from available messages transmitted by Communications Agents. As the agent can only transit from one node to a neighboring node, the minimum traversal time for the communication path is one (1) second. The latency due to the swarm behaviors is clearly negligible, as was intended by the design.

This example clearly demonstrates the viability of the design approach but, more importantly, it demonstrates a first principle design for a distributed swarm-agent based algorithm for coordination that displays the following features: 1) the use of simple, computationally inexpensive models of desirable behaviors/processes in place of more faithful reconstructions of biological analogs, i.e. Van der Pol oscillators instead of a network of pacemaker cells, 2) the effective use of stochastic systems and statistical analysis techniques to effect a first principles design of emergent behaviors, 3) the use of information-based measures to characterize system behaviors of interest, 4) the use of bounded autonomy to effectively coordinate system elements in a useful manner and to use stigmergic communications in a robust and effective manner, and 5) the distributed specification of swarm behaviors via a stand-alone kernel for generating random variates. The proposed methodology thus offers substantial advantages over other approaches for both meeting current operational needs and for providing a basis for scaling and accommodating new system configurations. The theoretical basis for this design methodology also offers a compelling basis for the design and implementation of complex networks of emergent algorithms via classical circuit design theory.

Alternative connectivity measures and their properties: Next, we introduce an alternative measure and discuss its potential for not only detecting coupling but also identifying the direction of the coupling. A key property of information measures is that they are invariant under a permutation. That is, an information measure is not dependent upon the order of the data considered. Depending upon context, this may be either a boon or a bane. In the current context, as has been noted, information measures do not embed historical relationships in the data and thus must be augmented using entropy rates or generalizations of correlation functions like correntropies to capture dynamics. A by-product of this lack is an inability to infer the direction (causality) of a coupling between variables. The current work focusing on an alternative information measure offers a potential means of addressing this shortcoming.

Consider two discrete-value random variables $X \in \{1,2,\ldots,n_X\}$ and $Y \in \{1,2,\ldots,n_Y\}$ with probability mass function $p_{XY}(x,y)$. An information measure based on the values of X and Y may be defined as follows:

$$M_I(x,y) = I\big(Z_X(x); Z_Y(y)\big)$$

where

- $Z_X(x)$ is a Bernoulli random variable such that $Prob(Z_X(x) = 1) = p_X(x)$
- $Z_Y(y)$ is a Bernoulli random variable such that $Prob(Z_Y(y) = 1) = p_Y(y)$

The total connectivity from the perspective of the value space can be computed by

$$M_{total} = \sum_{x=1}^{n_X} \sum_{y=1}^{n_Y} M_I(x, y)$$

The information measure M_{total} is closely related to the measure of mutual information between X and Y:

$$M_{total} = \sum_{x=1}^{n_X} \sum_{y=1}^{n_Y} Prob(X = x, Y = y) \log \frac{Prob(X = x, Y = y)}{Prob(X = x) Prob(Y = y)}$$

$$+ \sum_{x=1}^{n_X} \sum_{y=1}^{n_Y} Prob(X = x, Y \neq y) \log \frac{Prob(X = x, Y \neq y)}{Prob(X = x) Prob(Y \neq y)}$$

$$+ \sum_{x=1}^{n_X} \sum_{y=1}^{n_Y} Prob(X \neq x, Y = y) \log \frac{Prob(X \neq x, Y = y)}{Prob(X \neq x) Prob(Y = y)}$$

$$+ \sum_{x=1}^{n_X} \sum_{y=1}^{n_Y} Prob(X \neq x, Y \neq y) \log \frac{Prob(X \neq x, Y \neq y)}{Prob(X \neq x) Prob(Y \neq y)} .$$

Defining

$$C_2 = \sum_{x=1}^{n_X} \sum_{y=1}^{n_Y} Prob(X = x, Y \neq y) \log \frac{Prob(X = x, Y \neq y)}{Prob(X = x) Prob(Y \neq y)}$$

$$C_3 = \sum_{x=1}^{n_X} \sum_{y=1}^{n_Y} Prob(X \neq x, Y = y) \log \frac{Prob(X \neq x, Y = y)}{Prob(X \neq x) Prob(Y = y)}$$

$$C_4 = \sum_{x=1}^{n_X} \sum_{y=1}^{n_Y} Prob(X \neq x, Y \neq y) \log \frac{Prob(X \neq x, Y \neq y)}{Prob(X \neq x) Prob(Y \neq y)}$$

we obtain

$$M_{total} = I(X; Y) + C_2 + C_3 + C_4 .$$

Note that,

$$M_{total} \geq 0$$

and as can be sees from M_{total}, this new measure is tightly connected to mutual information. Further investigation of M_{total} could prove useful in the context of identifying alternative and, potentially, more robust measures of communication between system elements than those examined thus far. However, initial examination of this alternative measure has revealed an additional attribute that is absent in the previously considered measures. Specifically, we have found that the relationship between C_2 and C_3 has a potential use in detecting the direction of influence between two random variables. Since, this result is additive to mutual information, the focus of the work presented herein is on C_2 and C_3 rather than of M_{total}.

Direction of interconnection and alternative measures (Theoretical Result): Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be discrete random variables and assume that

$$Y \in \{1, \dots, M\} \triangleq D_Y$$

$$X \in \{(1,1), \dots, (1,N_1), \dots, (M,1), \dots, (M,N_M)\} \triangleq D_X.$$

In addition, we assume that

$$|D_X| = N \text{ or } \sum_{i=1}^{M} N_i = N$$

$$f: X \mapsto i \quad \forall X \in \{(i,1), \dots, (i,N_i)\}.$$

Rewriting C_2 as

$$\begin{aligned} C_2 &= \sum_{i=1}^{M}\sum_{j=1}^{N_i}\sum_{k=1}^{M} \text{Prob}\left(X=(i,j), Y\neq k\right) \log \frac{\text{Prob}\left(X=(i,j), Y\neq k\right)}{\text{Prob}\left(X=(i,j)\right)\text{Prob}\left(Y\neq k\right)} \\ &= \sum_{i=1}^{M}\sum_{j=1}^{N_i}\sum_{\substack{k=1 \\ k\neq i}}^{M} \text{Prob}\left(X=(i,j), Y\neq k\right) \log \frac{\text{Prob}\left(X=(i,j), Y\neq k\right)}{\text{Prob}\left(X=(i,j)\right)\text{Prob}\left(Y\neq k\right)} \\ C_2 &= \sum_{i=1}^{M}\sum_{j=1}^{N_i}\sum_{\substack{k=1 \\ k\neq i}}^{M} \text{Prob}\left(X=(i,j)\right) \log \frac{1}{\text{Prob}\left(Y\neq k\right)}. \end{aligned}$$

Similarly, C_3 is rewritten as

$$\begin{aligned} C_3 &= \sum_{i=1}^{M}\sum_{j=1}^{N_i}\sum_{k=1}^{M} \text{Prob}\left(X\neq(i,j), Y=k\right) \log \frac{\text{Prob}\left(X\neq(i,j), Y=k\right)}{\text{Prob}\left(X\neq(i,j)\right)\text{Prob}\left(Y=k\right)} \\ &= \sum_{i=1}^{M}\sum_{j=1}^{N_i}\sum_{\substack{k=1 \\ k\neq i}}^{M} \text{Prob}\left(X\neq(i,j), Y=k\right) \log \frac{\text{Prob}\left(X\neq(i,j), Y=k\right)}{\text{Prob}\left(X\neq(i,j)\right)\text{Prob}\left(Y=k\right)} \\ &\quad + \sum_{i=1}^{M}\sum_{j=1}^{N_i} \text{Prob}\left(X\neq(i,j), Y=i\right) \log \frac{\text{Prob}\left(X\neq(i,j), Y=i\right)}{\text{Prob}\left(X\neq(i,j)\right)\text{Prob}\left(Y=i\right)} \\ &= \sum_{i=1}^{M}\sum_{j=1}^{N_i}\sum_{\substack{k=1 \\ k\neq i}}^{M} \text{Prob}\left(Y=k\right) \log \frac{1}{\text{Prob}\left(X\neq(i,j)\right)} \\ &\quad + \sum_{i=1}^{M}\sum_{j=1}^{N_i} \text{Prob}\left(X\neq(i,j), Y=i\right) \log \frac{\text{Prob}\left(X\neq(i,j), Y=i\right)}{\text{Prob}\left(X\neq(i,j)\right)\text{Prob}\left(Y=i\right)} \\ C_3 &= C_{3,1} + C_{3,2}. \end{aligned}$$

Where

$$C_{3,1} = \sum_{i=1}^{M}\sum_{j=1}^{N_i}\sum_{\substack{k=1 \\ k\neq i}}^{M} \text{Prob}\left(Y=k\right) \log \frac{1}{\text{Prob}\left(X\neq(i,j)\right)},$$

$$C_{3,2} = \sum_{i=1}^{M}\sum_{j=1}^{N_i}\text{Prob}\left(X \neq (i,j), Y = i\right)\log\frac{\text{Prob}\left(X \neq (i,j), Y = i\right)}{\text{Prob}\left(X \neq (i,j)\right)\text{Prob}\left(Y = i\right)}$$

Note that

$$C_3 - C_2 = C_{3,2} + \left(C_{3,1} - C_2\right)$$

First, we consider

$$\begin{aligned}\frac{\text{Prob}\left(X \neq (i,j), Y = i\right)}{\text{Prob}\left(X \neq (i,j)\right)\text{Prob}\left(Y = i\right)} &= \frac{\text{Prob}\left(Y = i\right) - \text{Prob}\left(X = (i,j), Y = i\right)}{\left(1 - \text{Prob}\left(X(i,j)\right)\right)\text{Prob}\left(Y = i\right)} \\ &= \frac{\text{Prob}\left(Y = i\right) - \text{Prob}\left(X = (i,j)\right)}{\text{Prob}\left(Y = i\right) - \text{Prob}\left(Y = i\right)\text{Pr}\,ob\left(X(i,j)\right)} \\ &\leq 1\end{aligned}$$

This inequality is a direct consequence of the fact that a probability measure is bounded above by 1 and hence,

$$\log\frac{\text{Prob}\left(X \neq (i,j), Y = i\right)}{\text{Prob}\left(X \neq (i,j)\right)\text{Prob}\left(Y = i\right)} \leq 0$$

Therefore, by the non-negativity we obtain

$$C_{3,2} \leq 0$$

Next, $C_{3,1}$ is rewritten

$$\begin{aligned}C_{3,1} &= \sum_{i=1}^{M}\sum_{j=1}^{N_i}\sum_{\substack{k=1 \\ k\neq i}}^{M}\sum_{p=1}^{N_k}\text{Prob}\left(X = (k,p)\right)\log\frac{1}{\text{Prob}\left(X \neq (i,j)\right)} \\ &= \sum_{k=1}^{M}\sum_{j=1}^{N_k}\sum_{\substack{i=1 \\ k\neq i}}^{M}\sum_{p=1}^{N_i}\text{Prob}\left(X = (\text{i},p)\right)\log\frac{1}{\text{Prob}\left(X \neq (\text{k},j)\right)} \\ &= \sum_{i=1}^{M}\sum_{\substack{k=1 \\ k\neq i}}^{M}\sum_{j=1}^{N_k}\sum_{p=1}^{N_i}\text{Prob}\left(X = (i,p)\right)\log\frac{1}{\text{Prob}\left(X \neq (k,j)\right)} \\ C_{3,1} &= \sum_{i=1}^{M}\sum_{\substack{k=1 \\ k\neq i}}^{M}\sum_{p=1}^{N_k}\sum_{j=1}^{N_i}\text{Prob}\left(X = (\text{i},\text{j})\right)\log\frac{1}{\text{Prob}\left(X \neq (k,p)\right)}.\end{aligned}$$

and we have

$$C_{3,1}-C_2=\sum_{i=1}^{M}\sum_{j=1}^{N_i}\mathrm{Prob}\left(X=(i,j)\right)\left(\sum_{\substack{k=1\\k\neq i}}^{M}\left(\sum_{p=1}^{N_k}\log\frac{1}{\mathrm{Prob}\left(X\neq(k,p)\right)}-\log\frac{1}{\mathrm{Prob}\left(Y\neq k\right)}\right)\right)$$

Before proceeding to the proof, we introduce the following lemma and its corollary.

Lemma 1: Let $\boldsymbol{\alpha,\beta,\gamma}\geq\mathbf{0}$ and $\boldsymbol{\alpha}=\boldsymbol{\beta}+\boldsymbol{\gamma}$. The following inequality holds

$$\left(1-\gamma\right)\left(1-\beta\right)\geq\left(1-\alpha\right)$$

Proof:

$$\begin{aligned}\left(1-\gamma\right)\left(1-\beta\right)&=1-\left(\gamma+\beta\right)+\gamma\beta\\&=1-\alpha+\gamma\beta.\end{aligned}$$

By the non-negativity of both β and γ, the desired inequality results.

Corollary 1: Let $\boldsymbol{\alpha,\gamma_1,\ldots,\gamma_N}\geq\mathbf{0}$ **and** $\boldsymbol{\alpha}=\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}\boldsymbol{\gamma_i}$. The following inequality holds

$$\prod\nolimits_{i=1}^{N}\left(1-\gamma_i\right)\geq\left(1-\alpha\right)$$

Proof: Denote the partial sums $\sum_{\boldsymbol{j=i+1}}^{\boldsymbol{N}}\boldsymbol{\gamma_j}$ by $\boldsymbol{\alpha_i}$. Thus $\boldsymbol{\alpha_{i-1}}=\boldsymbol{\alpha_i}+\boldsymbol{\gamma_i}$ and the result follows.

By definition, $\sum_{p=1}^{N_k}Prob\left(X=(k,p)\right)=Prob(Y=k)$. By the non-negativity of probability measures we obtain

$$\begin{aligned}\prod\nolimits_{p=1}^{N_k}\left(1-\mathrm{Prob}\left(X=(k,p)\right)\right)&\geq\left(1-\mathrm{Prob}\left(Y=k\right)\right)\\\prod\nolimits_{p=1}^{N_k}\mathrm{Prob}\left(X\neq(k,p)\right)&\geq\mathrm{Prob}\left(Y\neq k\right)\\\log\prod\nolimits_{p=1}^{N_k}\mathrm{Prob}\left(X\neq(k,p)\right)&\geq\log\mathrm{Prob}\left(Y\neq k\right)\\\sum_{p=1}^{N_k}\log\mathrm{Prob}\left(X\neq(k,p)\right)&\geq\log\mathrm{Prob}\left(Y\neq k\right)\\-\sum_{p=1}^{N_k}\log\mathrm{Prob}\left(X\neq(k,p)\right)&\leq-\log\mathrm{Prob}\left(Y\neq k\right)\\\sum_{p=1}^{N_k}\log\frac{1}{\mathrm{Prob}\left(X\neq(k,p)\right)}&\leq\log\frac{1}{\mathrm{Prob}\left(Y\neq k\right)}.\end{aligned}$$

Thus, we have

$$C_{3,1}-C_2\leq0$$

and thus

$$C_2 \geq C_3$$

The result provides a potential mechanism for detecting the direction of coupling via this alternative information measure. For the simplest case where one random variable is a function of another random variable, this result may be directly applied. However, if one of the random variables is a function of more than one random variable, no conclusions may be drawn. As the latter case is more realistic in the context of real world observation processes, a stronger result is needed. Extending the application of information measures to detect the direction of coupling is highly desirable as it addresses a shortcoming of information measures and permits the elicitation of more structural information without the additional computational burden of computing information rates or correntropies.

Detection of Abnormality in a Power Plant using Machine Learning Approach: A power plant, as a system, may be modeled using a set of mathematical equations by representing

- x_t as the vector of internal variables of the power plant,
- y_t as the vector of observations from sensors measuring phenomena associated with equipment in the power plant,
- u_t as a set of control inputs which are manually input to the power plant,
- v_t as the process noise vector,
- w_t as the observation noise vector.

We denote the normal operation of a power plant by

$$\dot{x}_t = f_n(x_t, u_t, v_t)$$

$$y_t = g_n(x_t, w_t).$$

Clearly, if any abnormality (e.g., fault, environmental change) occurs in the plant, the dynamics of the plant will no longer be given by this model. In addition, observations from the sensors may also differ from the description given in the model. This latter condition could occur due to either the fault experienced by the plant or sensor degradation/failure. Mathematically, we represent the abnormally operating plant by

$$\dot{x}_t = f_{a_i}(x_t, u_t, v_t)$$

$$y_t = g_{a_i}(x_t, w_t)$$

where $\mathcal{A} \triangleq \{a_1, a_2, \dots, a_M\}$ are the set of possible faults in the power plant.

In general, it is not possible enumerate all elements of $\mathcal{A}$ due to the complexity of the plant. Therefore, the pair (f_{a_i}, g_{a_i}) cannot be modeled for all possible combination of failures, i.

Regardless, we wish to determine the condition of the plant from the set $\{n, a_1, a_2, \dots, a_M\}$ or at least differentiate between n and $\mathcal{A}$. In previous reports, the work focused on constructing the information theoretic framework for this purpose. In particular, the construction developed to date is based on the mutual information between two time-series generated from the two models (normal and faulted), respectively, using observation from each model, respectively.

First, each piece of equipment in the plant is considered separately. We denote them by $x_t^{(l)}$ where l indexes the individual components. Mathematically, each component is treated as possessing its own underlying dynamics and available observation processes as

$$\dot{x}_t^{(l)} = f_{m_i}^{(l)}(x_t, u_t, v_t)$$

$$y_t^{(l)} = g_{m_i}^{(l)}(x_t, w_t)$$

where $m_i \in \left\{n^{(l)}, a_1^{(l)}, a_2^{(l)}, \dots, a_{M_l}^{(l)}\right\}$ are the possible operational modes of the individual components and correspond to either normal or the faulted modes specific to that component, and $l = 1,2, \dots, L$.

Multiple components may operate in abnormal (faulted) conditions contemporaneously. Though, in general, only a small subset of components operates in a faulted mode while the rest operate normally. If one were able to determine the operating condition of each component at each instant, one could construct the elements composing the information theoretic framework directly from this information. However, it is not always the case that this a priori information is available. As a result, the elements of the information theoretic framework must be inferred from the raw data, as has been addressed in our previous work.

Mathematically, we define the operating condition of l^{th} component as $c_t^{(l)}$. We would like to estimate these operating conditions from observations, $\left\{y_{1:t}^{(1)}, y_{1:t}^{(2)}, \dots, y_{1:t}^{(L)}\right\}$. This problem is non-trivial even if the functional forms of $f_{m_i}^{(l)}$ and $g_{m_i}^{(l)}$ are known. This problem is impossible to solve analytically when $f_{m_i}^{(l)}$ and/or $g_{m_i}^{(l)}$ are unknown. If it is possible to classify or estimate $c_t^{(l)}$ by $\hat{c}_t^{(l)}$ such that

$\hat{c}_t^{(l)} \in \left\{n^{(l)}, a_1^{(l,1)}, \dots, a_1^{(l,N_{l,1})}, a_2^{(l)}, \dots, a_2^{(l,N_{l,2})}, \dots, a_{M_l}^{(l)}, \dots, a_{M_l}^{(l,N_{l,M})}\right\}$,

then the set $\left\{a_p^{(l,p)}, \dots, a_p^{(l,N_{l,p})}\right\}$ represents $a_p^{(l)}$.

Generally speaking, if the operational modes of the plant may be estimated in a manner such that can subsets of estimated conditions can be collapsed into a single operating condition, the requisite analyses can be performed using these estimates. This observation provides the motivation for introducing machine learning into our approach.

Machine Learning Approach on Time-series data for Equipment Condition and Monitoring in Power Plant: We emphasize that in general, the plant models are not known a priori. Therefore, it is not an easy task to label abnormal operational modes directly from sensor data. That is, an oracle is needed to label the relevant features of time-series data in such a way that this label will have utility for detecting abnormal plant behaviors. For this reason, standard classification methods do not obtain. Hence, our focus will be placed on clustering algorithms.

One of the key aspects in applying classification and/or clustering algorithm is feature selection. It is inefficient and commonly computationally prohibitive to use entire time-series as feature sets for these algorithms. In addition, it is desirable to capture important characteristics that are sufficient for describing the information content of the time-series and can provide a means for differentiating between the time-series associated with different behaviors. In particular, these features can be thought of as information rich bases for describing the time-series and, thus, provide the appropriate basis for considering time-series within an information-theoretic framework.

Feature Selection and Distance measure: A crucial attribute of the features sought is that it be able to capture the difference in motions, e.g. trajectories, of the signals. The emphasis placed on these attributes reflects the hypothesis that abnormal equipment will, on the whole, display a greatly different range of motion in its internal variables (as reflected in the trajectories of the associated phenomena being observed) than healthy equipment will. To this end, we examine a feature set that is invariant to both scaling and shift. More specifically, we propose the use of the empirical amplitude distribution (i.e. normalized histogram) as a feature set. Note also, that if scaling and shift magnitude are found to be necessary for detection and diagnostics, they can be easily added into the feature set. Mathematically, for a time-series data, $x_{1:T}$, and a

level of quantization, Q, we construct a histogram using the variables $h_1, h_2, \dots, h_Q$ such that for $X_{max} = \max_{t=1,\dots,T} x_t$ and $X_{min} = \min_{t=1,\dots,T} x_t$, $h_i = \frac{1}{T}\sum_{t=1}^{T} \delta_{\left(x_t - (X_{max}-X_{min})\frac{i}{Q} + X_{min}\right)}$ and δ is Kronecker delta.

At first glance, h_i appears sufficient to capture the motion of $x_{1:T}$, however an empirical study reveals that this is not the case in practice. The information provided by h_i must be augmented in order to achieve the desired detection goals. The simplest methods for capturing the underlying dynamics are time-delay embedding techniques that seek to discover the intrinsic "embedding dimension" of a system (i.e. the dimension of the embedded manifold in the state space on which a system is constrained to evolve) and adding time constraints to the original signal. For the time series data $x_{1:T}$, we construct the variables

$$Z^{(1)}_{1:T-1} \text{ where } Z^{(1)}_i = [X_{i+1}, X_i]^T$$

$$Z^{(2)}_{1:T-1} \text{ where } Z^{(2)}_i = [X_{i+1}, X_{i+1} - X_i]^T$$

The new variable $Z^{(1)}_{1:T-1}$ captures the embedded dimension of the original signal by adding a time-delayed copy of the data to increase the dimension. The second variable $Z^{(2)}_{1:T-1}$ is a combination of the original signal and its "derivative" information. Either approach provides a mechanism for capturing both the value space and the temporal evolution of the original signal. Using these higher dimensional variables, a feature set is defined using two-dimensional histograms:

- $h^{(1)}_{ij} = \frac{1}{T}\sum_{t=1}^{T} \delta_{\left(x_{t+1} - (X_{max}-X_{min})\frac{i}{Q} + X_{min}\right)} \delta_{\left(x_t - (X_{max}-X_{min})\frac{j}{Q} + X_{min}\right)}$,
- $h^{(2)}_{ij} = \frac{1}{T}\sum_{t=1}^{T} \delta_{\left(x_{t+1} - (X_{max}-X_{min})\frac{i}{Q} + X_{min}\right)} \delta_{\left(x_t - (Y_{max}-Y_{min})\frac{j}{Q} + Y_{min}\right)}$,

where $Y_{max} = \max_{t=1,\dots,T-1}(x_{t+1} - x_t)$, $Y_{min} = \min_{t=1,\dots,T-1}(x_{t+1} - x_t)$ and δ is the Kronecker delta.

One obvious way to measure the distance between two histograms with the same binning is that, for two sets of 2-D histogram $\{h^A_{ij}\}$ and $\{h^B_{ij}\}$, the distance between the histogram of A and the histogram of B may be computed using the n^{th} norm by

$$d^n\left(h^A_{ij}, h^B_{ij}\right) = \sum_{i=1}^{N}\sum_{j=1}^{M} \left\| h^A_{ij} - h^B_{ij} \right\|^{\frac{1}{n}}$$

where N is the number of bins in the first axis and M is the number of bins in the second axis. In the absence of a priori information that provides insight into an appropriate norm, the 2-norm is commonly applied. In general, it is desired only to match the shape of the distribution, e.g. the histogram. If there is additional noise or perturbations that causes a change in the histogram for a given specific quantization, a slight modification of the distance metric has been developed. First, align a_i and a_j such that the correlation coefficient between the two histograms is maximum, i.e.:

$$\left(a_i, a_j\right) = \arg\max_{x,y} \rho\left(\{\tilde{h}^A_{ij}(x,y)\}, \{\check{h}^B_{ij}(x,y)\}\right)$$

where

- $\tilde{h}_{ij}(x,y) = h_{ij} \qquad \forall i = 1, \dots, N-x,\ \forall j = 1, \dots, M-y,$
- $\check{h}_{ij}(x,y) = h_{(i+x)(j+y)} \qquad \forall i = 1, \dots, N-x,\ \forall j = 1, \dots, M-y,$
- $\rho(X,Y) = \frac{cov(X,Y)}{var(X)var(Y)}$.

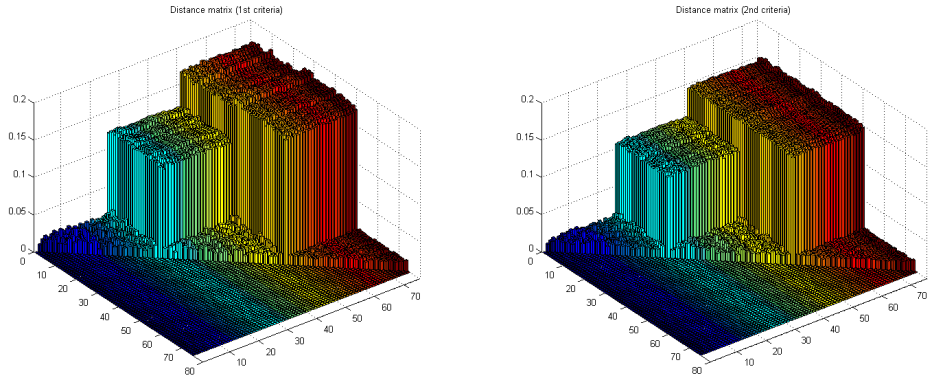
The distance between two histograms, $\{\tilde{h}^A_{ij}(a_i, a_j)\}$ and $\{\check{h}^B_{ij}(a_i, a_j)\}$, is then computed.

Testing the proposed feature set: Two scenarios have been simulated to test the effectiveness of the proposed feature set. The first scenario is based on the simulation of three different linear systems driven by noise. The second scenario focuses on the simulation of coupled and uncoupled chaotic systems. In this preliminary stage of algorithm development, the investigation is focused on examining the distances computed between data from the same systems and that computed between data from different systems. It is expected that the distances between data from the same systems will always be smaller than the distances between data from different systems. The number of bins used for quantization is fixed at 15 for all simulations.

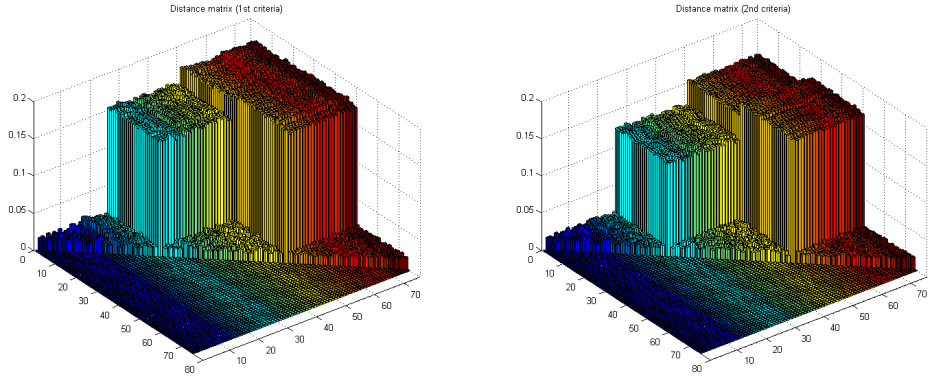
Linear system simulations driven by noise: In these simulations, three different linear systems are examined. The linear systems considered are :

$$Y_t^{(1)} = -1.05Y_{t-1}^{(1)} - 0.135Y_{t-2}^{(1)} + 0.85X_t$$
$$Y_t^{(2)} = -8.5Y_{t-1}^{(2)} + 0.15X_t - 0.3X_{t-1}$$
$$Y_t^{(3)} = 0.8Y_{t-1}^{(3)} - 0.0975Y_{t-2}^{(3)} + 0.6X_t - 0.1X_{t-1} - 0.3X_{t-2}$$

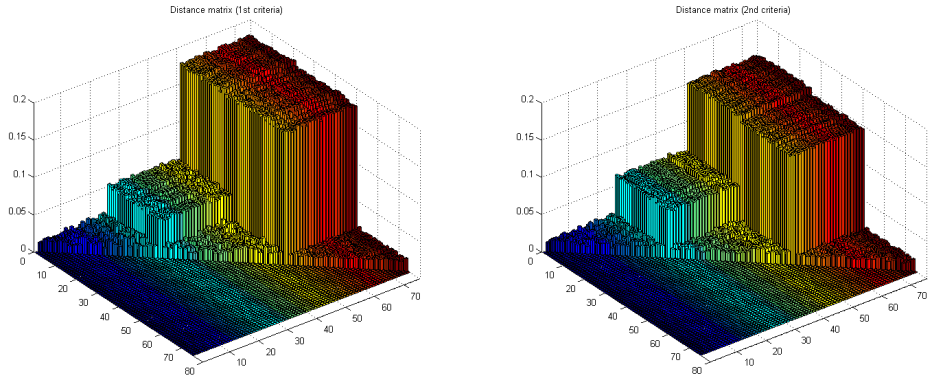
For the first simulation, a long sequence of length 250,000 is generated where X_t is white noise with an underlying probability distribution $N(0,1)$. In addition, all initial conditions for $Y_t^{(i)}$ are $N(0,1)$ and independent of X_t. A histogram is computed using non-overlapping windows of size 10,000 and, therefore, each system is examined using a collection of 25 feature sets for a total of 75 feature sets. The distance between the 75 feature sets is computed and the result is represented as a (symmetric) distance matrix, presented as a bar chart where redundant (subdiagonal) entries are zeroed (for visualization purposes). In all figures presented, the subfigure on the left-hand side is the histogram for $Z_{1:T-1}^{(2)}$ while the subfigure on the right-hand side is the histogram for $Z_{1:T-1}^{(1)}$, respectively. The results are shown in Figure .35.


Distance matrix (1st criteria)
Distance matrix (2nd criteria)

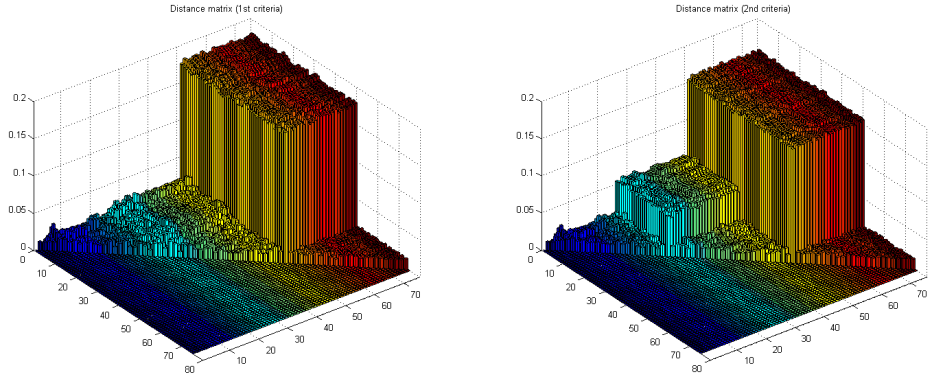
(a)


Distance matrix (1st criteria)
Distance matrix (2nd criteria)

(b)



(c)

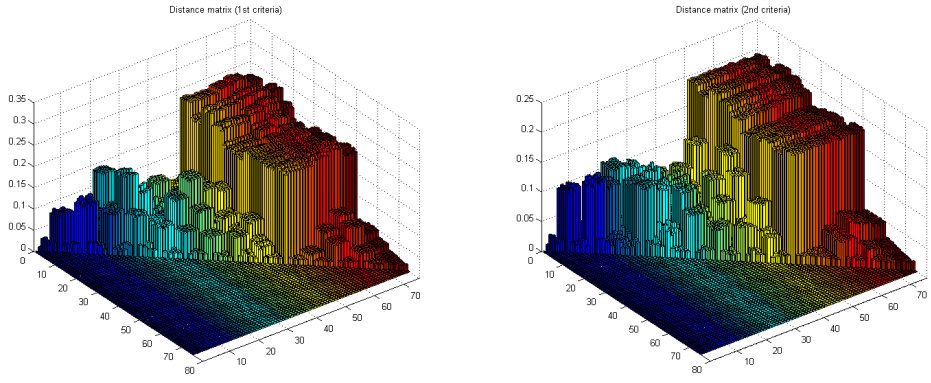


(d)

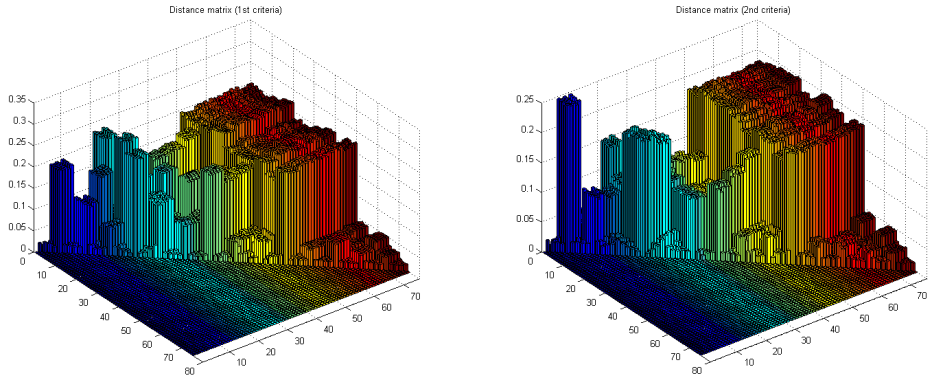
Figure 2.35: Results from long sequence of length 250,000 with non-overlapping windows of size 10,000 for three linear systems. All extracted features are treated as individual data. Thus, 75 total data are presented where first 25 data are from the system #1, next 25 data are from the system #2; and last 25 data are from the system #3

Figure 2.35 (a) and (b) presents a case where the proposed features are based on the distances computed between windowed data segments for the two augmented variables $Z_{1:T-1}^{(1)}$ and $Z_{1:T-1}^{(2)}$ can resolve the three systems. The results in Figure 2.35 (c), however, show that this feature description does not clearly resolve system #1 from system #2, though system #3 can be resolved from the other two systems. Notice that the distance measures computed using data from the same trajectory produce measures showing that the trajectories are relatively close. This holds for the data from all three systems. In Figure 2.35 (d), $Z_{1:T-1}^{(1)}$ can resolve system #1 from system #2 while $Z_{1:T-1}^{(2)}$ is unable to do so. This may indicate that $Z_{1:T-1}^{(1)}$ might provide a better augmentation of X_t than $Z_{1:T-1}^{(2)}$ in this case. In Figure 2.35 (d), $Z_{1:T-1}^{(1)}$ can resolve system #1 from system #2 while $Z_{1:T-1}^{(2)}$ is unable to do so. This may indicate that $Z_{1:T-1}^{(1)}$ might provide a better augmentation of X_t than $Z_{1:T-1}^{(2)}$ in this case.

Encountering two or more systems that are, for all intents and purposes, “identical” is not uncommon in practice. More precisely, it is very possible that two similar systems at different locations provide the same functionality and possess the same underlying dynamics. To examine this situation, shorter sequences of length 50,000 were generated and sampled using non-overlapping windows of size 10,000, as in previous case. For each system, 5 different realizations were generated. As in the previous case, there are 75 data sets, total, with 25 data sets associated with each system. The only difference is that, contained within the 25 data sets associated with each system are five (5) collections of five (5) data sets, each collection associated with a different realization. As before, the subfigure on the left-hand side shows the histogram for $Z_{1:T-1}^{(2)}$ while the subfigure on the right-hand side is the histogram for $Z_{1:T-1}^{(1)}$,. The results are shown in Figure 2.36 below.



(a)



(b)

Figure 2.36: Results from set of 5 realizations each contained sequence of length 50,000 with non-overlapping windows of size 10,000 for three linear systems. All the extracted features are treated as individual data. Thus, 75 total data are presented where first 25 data are from the system #1, next 25 data are from the system #2; and last 25 data are from the system #3

The results obtained using the realizations featured in Figure 2.36 (a) and (b) demonstrate that the proposed approach cannot be used to group data from system #1 and system #2 if they come from different realizations. For these systems, the distribution over the sample paths are such that it is almost as likely that a pair of realizations is close as it is that they are far apart. In addition, data from system #1 and system #2 cannot be separated by the proposed feature set. However, data from system #3 are relatively close to each other despite the fact that they come from different realizations, reflecting the fact that the distribution over the sample paths for system #3 has a much smaller variance. Also, data from system #3 are well separated from system #1 and system #2. This could indicate that if the trajectories of any data from different systems are sufficiently different and the variances of the associated sample paths are sufficiently small, our method can be used to distinguish them.

To refine this analysis, an additional simulation was performed. In this simulation, data sets of size 10,000 samples were generated for each realization and for each system. Five (5) ensemble data sets were then constructed for each system by averaging 100 realizations for each ensemble data set. Thus, for each system, five (5) ensemble data sets are constructed. The intent here is to investigate if it is possible to collect data from different realization and use average features to resolve the three systems. The results are shown in Figure 2.36 where the histogram of the augmented variable $\boldsymbol{Z}^{(2)}_{1:T-1}$ is shown on the left-hand side and that for $\boldsymbol{Z}^{(1)}_{1:T-1}$ is shown on the right-hand side.

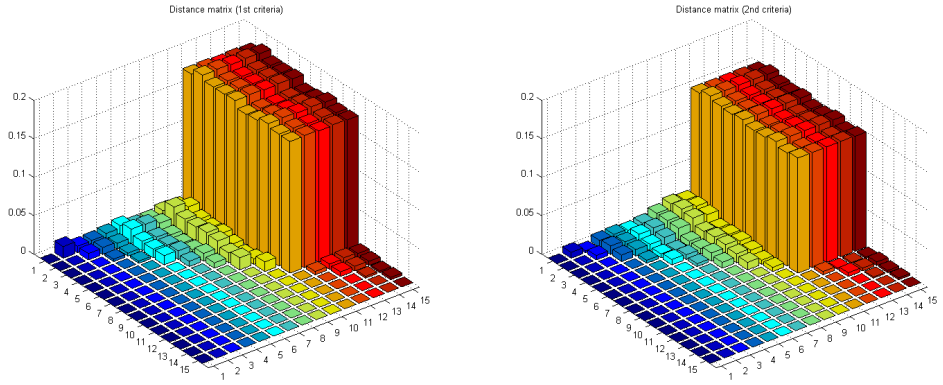


Figure 2.37: Results from set of 500 realizations, each containing a long sequence of length 10,000 for each of the three linear systems. All extracted features are averaged over 100 realizations to form single ensemble data set. Thus, 15 total data sets are presented where first 5 data are from the system #1, the next 5 data sets are from system #2; and the last 5 data sets are from system #3

Figure 2.37 indicates that, by averaging over 100 realizations, system #1 and system #2 can no longer be resolved. Conversely, system #3 remains well separated from the other two systems. These results show that this approach, particularly as implemented here, has significant limitations. These limitations are similar to those of clustering methods that cannot resolve clusters whose mean values and variances are such that significant overlap of cluster membership occurs. One possible approach to improve the robustness of this approach is to investigate clustering based on the distributions over the underlying distributions via Dirichlet and Beta processes.

Simulations for Coupled and Uncoupled Chaotic Systems As discussed in prior reports, the necessary focus of this work is not restricted to stochastic systems but must also consider deterministic systems whose behavior, while deterministic, is sufficiently complicated that it may approach the unpredictability of stochastic systems. In order to examine this aspect, three coupled chaotic systems are considered:

1. Coupled Hénon Maps

$$\begin{cases} x_{n+1}^{(1)} = 1.4 - {x_n^{(1)}}^2 + 0.3x_n^{(2)} \\ x_{n+1}^{(2)} = x_n^{(1)} \end{cases}$$

$$\begin{cases} x_{n+1}^{(3)} = 1.4 - \left(\boldsymbol{C}x_n^{(1)}x_n^{(3)} + (1-\boldsymbol{C}){x_n^{(3)}}^2\right) \\ \qquad\qquad +0.3x_n^{(4)} \\ x_{n+1}^{(4)} = x_n^{(3)} \end{cases}$$

where

- $C = 0,0.04,0.08,\ldots,0.8$
- Observations: $x_{n+1}^{(2)}$ and $x_{n+1}^{(4)}$

- Number of time steps: 10,000

2. Coupled Lorenz Systems

$$\begin{cases} \dot{x}_t^{(1)} = 10\left(x_t^{(2)} - x_t^{(1)}\right) \\ \dot{x}_t^{(2)} = x_t^{(1)}\left(28 - x_t^{(3)}\right) - x_t^{(2)} \\ \dot{x}_t^{(3)} = x_t^{(1)}x_t^{(2)} - \frac{8}{3}x_t^{(3)} \end{cases}$$
$$\begin{cases} \dot{x}_t^{(4)} = 10\left(x_t^{(5)} - x_t^{(4)}\right) \\ \dot{x}_t^{(5)} = x_t^{(1)}\left(28.001 - x_t^{(6)}\right) - x_t^{(5)} \\ \dot{x}_t^{(6)} = x_t^{(4)}x_t^{(5)} - \frac{8}{3}x_t^{(6)} + \boldsymbol{C}\left(x_t^{(3)} - x_t^{(6)}\right) \end{cases}$$

where

- $C = 0,0.1,0.2,\ldots,2$
- Observations: $x_{n+1}^{(1)}$ and $x_{n+1}^{(4)}$
- Number of data points: 10,000
- Numerical Integration: Runge-Kutta with step size 0.01 seconds

3. Coupled Rössler Systems

$$\begin{cases} \dot{x}_t^{(1)} = -0.95x_t^{(2)} - x_t^{(3)} \\ \dot{x}_t^{(2)} = 0.95x_t^{(1)} + 0.15x_t^{(2)} \\ \dot{x}_t^{(3)} = 0.2 + x_t^{(3)}\left(x_t^{(1)} - 10\right) \end{cases}$$
$$\begin{cases} \dot{x}_t^{(4)} = -0.95x_t^{(5)} - x_t^{(6)} + \boldsymbol{C}\left(x_t^{(1)} - x_t^{(4)}\right) \\ \dot{x}_t^{(5)} = 0.95x_t^{(4)} + 0.15x_t^{(5)} \\ \dot{x}_t^{(6)} = 0.2 + x_t^{(6)}\left(x_t^{(4)} - 10\right) \end{cases}$$

where

- $C = 0,0.1,0.2,\ldots,2$
- Observations: $□_{n+1}^{(1)}$ and $x_{n+1}^{(4)}$
- Number of data points: 10,000
- Numerical Integration: Runge-Kutta with step size 0.1 seconds

Uncoupled systems, e.g. $C = 0$, are examined first with respect to the efficacy of the proposed feature representations for resolving the three chaotic systems and to test this approach, a realization of length 50,000 generated for each system is used to construct the histograms corresponding to the two augmented variables. For each realization, the histogram is constructed using non-overlapping segments of length 10,000 for a total of 30 data sets. The resulting histograms are presented in Figure 2.38 where the histogram corresponding to $Z_{1:T-1}^{(2)}$ is shown on the left-hand side and that corresponding to $Z_{1:T-1}^{(1)}$ is shown on the right-hand side.

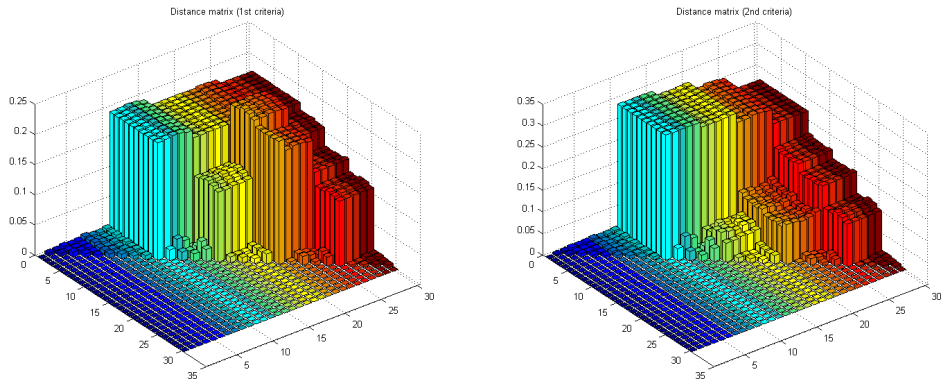


Figure 2.38: Results from set of 5 non-overlapped segments from sequence of length 50,000 for three coupling chaos systems. All the extracted features are treated as individual data. Thus, 30 total data are presented where first 10 data are from Hénon maps, next 10 data are from the Lorenz systems; and last 25 data are from the Rössler systems. For each system, two sets of 5 data are from the different uncoupled subsystems

Figure 2.38 shows that not only can $Z_{1:T-1}^{(2)}$ resolve the three chaotic systems it can also resolve the two subsystems of the Lorenz and Rössler systems. Note that the two subsystems of Lorenz and Rössler systems have different underlying dynamics while the two subsystems of Hénon maps have the same underlying dynamics. This demonstrates that $Z_{1:T-1}^{(2)}$ provides an effective basis for comparing uncoupled chaotic systems. $Z_{1:T-1}^{(1)}$ is also an effective basis for resolving these different systems but is less effective with Lorenz systems. This reflects variations in distance between different segments obtained from the same realization. Although, the results obtained in this scenario seem satisfactory, it is also necessary to investigate the effect of coupling strength on the ability to resolve differences between the disparate systems.

To examine the importance of coupling strength to the performance of the proposed algorithms, separate simulations were be run for Hénon maps, Lorenz systems and Rössler systems. Also, attention is restricted to the use of $Z_{1:T-1}^{(2)}$ as a basis for comparing the various systems since it outperforms $Z_{1:T-1}^{(1)}$. For each system and coupling level, realizations of length 50,000 are used. Further, only the data obtained from one of the coupled systems (the second or driven subsystem) is considered here. As the first subsystem drives the second subsystem and the underlying dynamics of the first (driving) subsystems remain the same for all chaotic systems considered here. The histograms are generated from data sets of size 10,000 samples using non-overlapping segments of the corresponding time-series data. Hence, total of 105 data points for each coupled chaotic system pair. The results over the set of coupling strengths $C = \{0,0.04,0.08,\ldots,0.8\}$ are presented below. As the total number of coupled system pairs considered here is significantly greater than that considered above, the results are presented via heat maps as bar plots are not effective for visualizing the results for such a large number of cases.

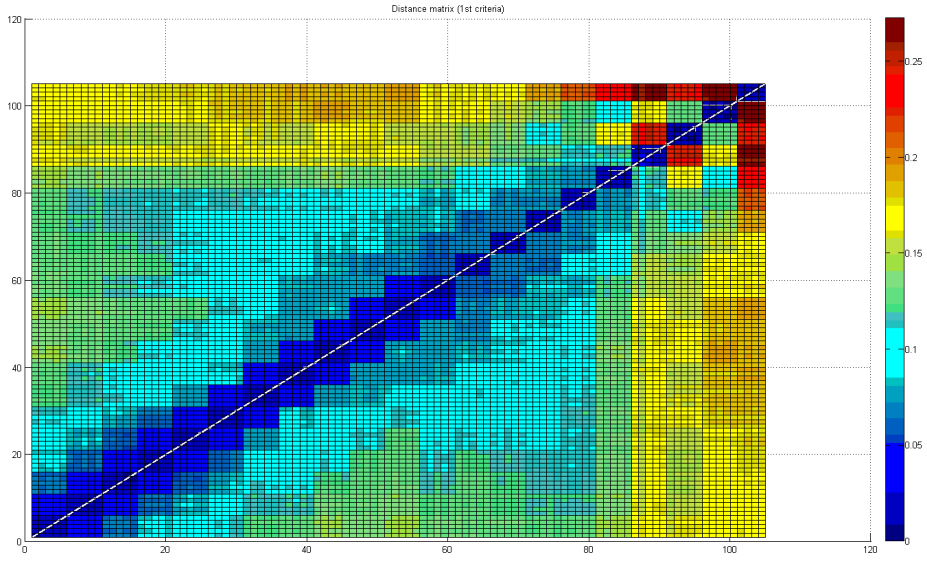


Figure 2.39: Results from set of 5 non-overlapping segments from sequence of length 50,000 for Hénon maps with coupling strengths $\boldsymbol{C = 0, 0.04, 0.08, \ldots, 0.8}$. All extracted features are treated as individual data points (i.e., 105 total data points are presented).

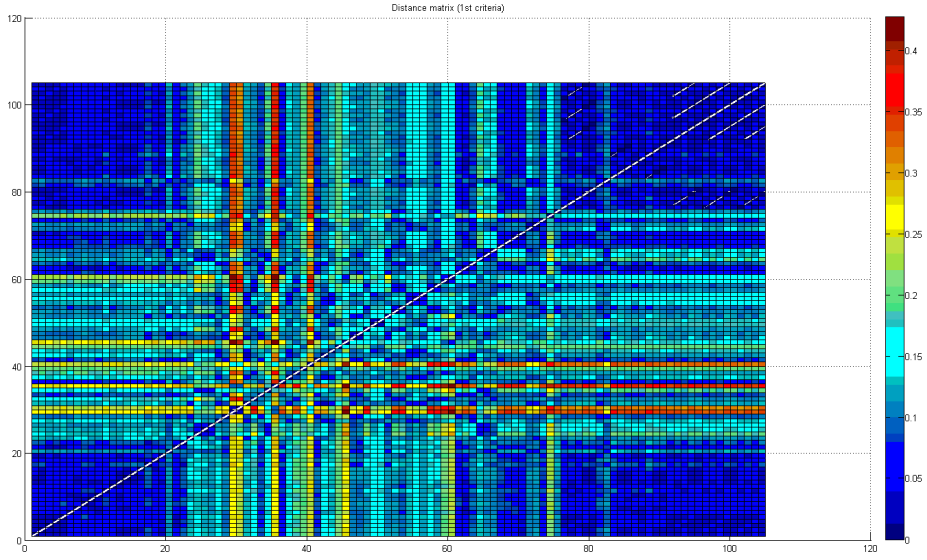


Figure 2.40: Results from set of 5 non-overlapping segments from sequence of length 50,000 for Lorenz systems with coupling strengths $\boldsymbol{C = 0, 0.1, 0.2, \ldots, 2}$. All extracted features are treated as individual data points (i.e., 105 total data are presented)

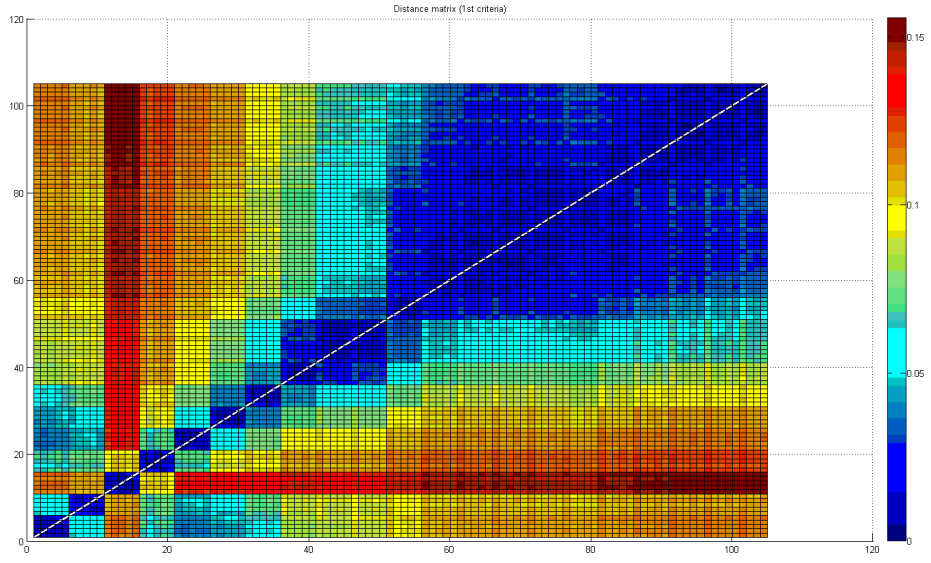


Figure 2.41: Results from set of 5 non-overlapping segments from sequence of length 50,000 for Rössler systems with coupling strengths $\boldsymbol{C = 0, 0.1, 0.2, \ldots, 2}$. All extracted features are treated as individual data (i.e., 105 total data are presented)

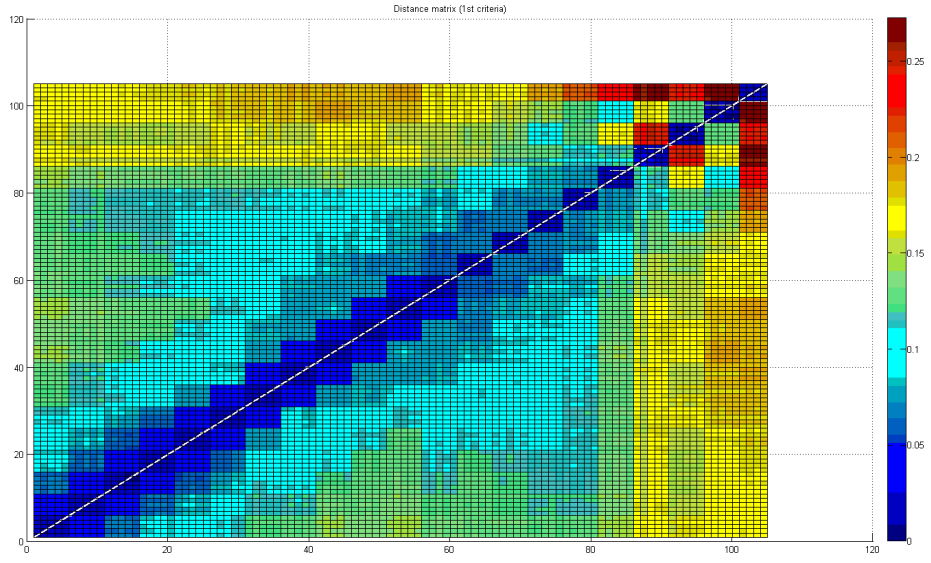


Figure 2.39 and reveal that the distances between data sets obtained from the same trajectory are relatively close. On coupled Hénon maps, for the low to moderate coupling level, the proposed features may potentially reflect the coupling strength. Specifically, the distances between the data from the different realizations possessing the same coupling level are relatively small. In addition, the distances between the data from different systems with different coupling levels are proportional to the differences in coupling level. At higher coupling levels, $\boldsymbol{C \geq 0.68}$, this proportionality does not hold as all of the data are close to each other at these levels. Regardless, the results indicate that this feature set can be used to resolve coupled

Hénon maps with different coupling levels. Similarly, as shown in

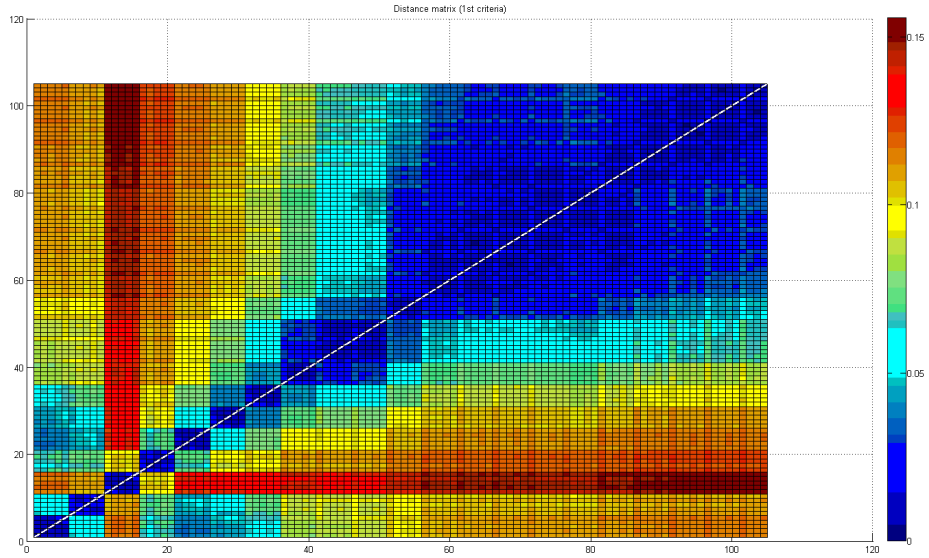


Figure 2.41, the distances between data sets from Rössler systems with different levels of coupling can be resolved for $\boldsymbol{C \leq 1}$. In contrast, when $\boldsymbol{C > 1}$, the proposed featural description cannot be used to distinguish between data from systems having different coupling levels. Finally, for Lorenz systems, the proposed features are unable resolve between different coupling levels, as shown in Figure 2.40. Also noteworthy here is that, under the proposed feature description, some segments of data from the same realization appear to be far one another.

We observe the proposed feature set based on the underlying distributions (estimated via histograms) are able to resolve data from systems possessing different underlying dynamics under some conditions analogous to those seen in typical clustering approaches (i.e., sufficient separation between distributions). This approach can be used to roughly group linear systems with different parameters and, more importantly, it can clearly resolve the three different coupled chaotic systems and their subsystems examined here provided that their underlying dynamics are different. In addition, it can also distinguish between the data from Hénon maps at different levels of coupling. This result also holds for coupled Rössler systems for coupling parameter less than or equal to 1 but fails entirely at distinguishing data from coupled Lorenz systems with different coupling levels.

Depending on the characteristics of the data sets examined going forward, this approach may be sufficient for detection and analysis needs though it is likely that this approach will need to be refined or augmented with additional approaches. This refinement and investigation of additional techniques is a key focus of the results to be presented in the next section. In parallel with the effort to improve the ability of the detection algorithms to resolve between different systems is a companion effort focusing on identifying features of time-series data that intrinsically provide better separation and thus improve the effectiveness of the current set of detection algorithms. Finally, it is desirable to extend the capability of these algorithms to include diagnosis. That is, in addition to being able to resolve differences between time series data generated by different systems, it is also desirable that the algorithms be able to effectively classify these data. To this end, a thorough investigation of clustering algorithms is necessary and appropriate techniques and algorithms must be selected and implemented to achieve the project goals.

References for Connection between Control Theory and Information theory:

[1] J. Zaborszky, “An information theory viewpoint for the general identification problem,” IEEE Transactions on Automatic Control, vol. 11, no. 1, pp. 130-131, 1966.
[2] H. L. Weidemann and E. B. Stear, “Entropy Analysis of Estimating Systems,” IEEE Transactions on Information Theory, vol. IT-16, no. 3, pp. 264-270 , May 1970.
[3] Y. Tomita, S. Omatu, and T. Soeda, “An application of the information theory to filtering problem,” Information Sciences, vol. 11, pp. 13-27, 1976.
[4] P. Kalata and R. Priemer, “Linear prediction, filtering and smoothing: An information theoretic approach,” Information Sciences, vol. 17, pp. 1–14, 1979.
[5] G. N. Saridis, “Entropy formulation of optimal and adaptive control,” IEEE Transactions on Automatic Control, vol. 33, no. 8, pp. 713-721, 1988.
[6] Y. A Tsai, F. A. Casiello, and K. A. Loparo, “Discrete time entropy formulation of optimal and adaptive control problem,” IEEE Transactions on Automatic Control, vol. 37, no. 7, pp. 1083-1087, 1992.
[7] X. Feng, K. A. Loparo, and Y. Fang, “Optimal State Estimation for Stochastic Systems: An Information Theoretic Approach,” IEEE Transactions on Automatic Control, vol. 42, no. 6, pp. 771-785, June 1997.
[8] X. Feng and K. A. Loparo, “Active Probing for Information in Control Systems with Quantized State Measurements: A Minimum Entropy Approach,” IEEE Transactions on Automatic Control, vol. 42, no. 2, pp. 216-238, February 1997.
[9] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition”, Proceedings of the IEEE, vol. 77, issue 2, pp. 257-286, Feb 1989.
[10] Alan B. Poritz, “Linear predictive hidden Markov models and the speech signal”, Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82, vol. 7, p.p. 1291-1294, May 1982.
[11] Louis A. Liporace, “Maximum likelihood estimation for multivariate Observations of Markov Sources”, IEEE Transactions on Information Theory, vol. 28, issue 5, p.p. 729-734, Sep 1982.
[12]B. H. Juang, Stephen E. Levinson and M. M. Sondhi, “Maximum likelihood estimation for multivariate mixture observation of markov chain”, IEEE Transactions on Information Theory, vol. 32, no. 2, p.p. 307-309, Mar 1986.
[13] Marco F. Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D. Hanebeck, “On Entropy Approxilllation for Gaussian Mixture Randolll Vectors”, Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Seoul, Korea, August 20-22, 2008.
[14] Andrew R. Runnalls, “Kullback-Leibler Approach to Gaussian Mixture Reduction”, IEEE Transactions on Aerospace and Electronic Systems, vol. 43, no. 3, p.p. 989-999, July 2007.
[15] P. Grassberger, “Estimating the Information Content of Symbol Sequences and Efficient Codes”, IEEE Transactions on Information Theory, vol. 35, no. 3, p.p. 669-675, May 1989.
[16] A. D. Wyner and J. Ziv, “Some Asymptotic Properties of the Entropy of a Stationary Ergodic Data Source with Applications to Data Compression”, IEEE Transactions on Information Theory, vol. 35, no. 6, p.p. 1250-1258, Nov 1989.
[17] D. S. Ornstein and B. Weiss, “Entrop nformation Theory, IEEE Transactions on Information Theory, vol. 39, no. 1, p.p. 78-83, Jan 1993.
[18] A. N. Quas, “An entropy estimator for a class of infinite alphabet processes”, Theory of Probability & Its Applications, vol.43, no. 3, p.p. 496–507, 1999.
[19] I. Kontoyiannis, P. H. Algoet, Yu. M. Suhov, and A. J. Wyner, “Nonparametric Entropy Estimation for Stationary Processesand Random Fields, with Applications to English Text”, IEEE Transactions on Information Theory, vol. 44, no. 3, p.p. 1319-1327, May 1998
[20] A. Kraskov, H. Stögbauer, and P. Grassberger, “Estimating mutual information”, Phys. Rev. E, vol. 69, no. 6, 2004.
[21] Steven M. Pincus, “Approximate entropy as a measure of system complexity”, Proc. Nati. Acad. Sci. USA, vol. 88, pp. 2297-2301, March 1991

References for Estimation of Mutual Information Measures:

Akyol, B., Haack, J., Ciraci, S., Carpenter, B., Vlachopoulou, M., & Tews, C. (2012). VOLTTRON: An Agent Execution Platform for the Electric Power System. *In Proceedings of the 3rd International Workshop on Agent Technologies for Energy Systems.* Valencia, Spain.

Alexander, S., Bernasconi, J., Schneider, W. R., & Orbach, R. (1989). Excitation dynamics in random one-dimensional systems. *Reviews of Modern Physics , 53* (2), 175-198.

Badi, R., & Politi, A. (1997). *Complexity: Hierarchical structures and scaling in physics* (Vol. 6). Cambridge, UK: Cambridge Universiy Press.

Barrat, A., Barthelemy, M., & Vespignani, A. (2008). *Dynamical Processes on Complex Networks.* Cambridge, U.K.: Cambridge University Press.

Bentley, P. J., Gordon, T., Kim, J., & Kumar, S. (2001). New Trends in Evolutionary Computation. *Proceedings of the 2001 Congress on Evolutionary Computation*, (pp. 162-169).

Bernasconi, J., & Schneider, W. R. (1982). Diffusion in a one-dimensional lattice with random asymmetric transition rates. *J. Physics A , 15*, L729.

Bilchev, G., & Parmee, I. C. (1995). The Ant Colony Metaphor for Searching Contiunous Design Spaces. In T. C. Fogarty (Ed.), *AISB Workshop on Evolutionary Computing, Lecture Notes in Computer Science. 993*, pp. 25-39. Berlin: Springer-Verlag.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems.* New York, NY: Oxford University Press.

Bonabeau, E., Sobkowski, A., Theraulaz, G., & Deneubourg, J.-L. (1996). Quantitative Study of the Fixed Threshold Model for the Regulation of Division of Labour in Insect Societies. *Proc. Royal Society of London B , 263*, 1565-1569.

Bouchaud, J.-P., & Mezard, M. (1994). Self induced quenched disorder: a model for the spin glass transition. *J. Phys. I , 4*, 1109.

Colorni, A., Dorigo, M., & Maniezzo, V. (1992). An Investigation of Some Properties of an Ant Algorithm. In R. Manner, & B. Manderick (Ed.), *Proc. 1992 Parallel Problem Solving From Nature Conf.* (pp. 509-520). Amsterdam: Elsevier.

Conant, R. C. (1975). Laws of Information which Govern Systems. *IEEE Transactions on Systems, Man and Cybernetics , SMC-6* (4), 240-255.

Conant, R. C. (1976). Laws of Information which Govern Systems. *IEEE Trans. Sys., Man, & Cyber. , SMC-6* (4), 240-255.

Cover, T. M., & Thomas, J. A. (2012). *Elements of Information Theory* (Second ed.). Hoboken, NJ: John Wiley & Sons.

Deneubourg, j.-L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., & Chretien, L. (1991). The Dynamics of Collective Sorting: Robot-Like Ant and Ant-Like Robot. In J. A. Meyer, & S. W. Wilson (Ed.),

Proc. 1st Conference on Simulation of Adaptive Behavior: From Animals to Animats (pp. 356-365). Cambridge, MA: MIT Press.

Deneubourg, J.-L., Goss, S., Pasteels, J. M., Fresneau, D., & Lachaud, J.-P. (1987). Self-Organization Mechanisms in Ant Societies (II): Learning in Foraging and Division of Labour. *Experimentia Suppl. , 54*, 177-196.

Derrida, B. (1983). Velocity and diffusion constant of a periodic one-dimensional hopping model. *J. Statistical Physics , 31*, 433-450.

Derrida, B., & Pomeau, Y. (1982). Classical diffusion on a random chain. *Physical Review Letters , 48*, 627-630.

Di Carlo, G., & Dorigo, M. (1998). AntNet: distributed stigmergic control for communications networks. *J. Art. Int. Res. , 9*, 317-365.

Dorigo, M. (1994). Learning by Probabilistic Boolean Networks. *IEEE International Confernece on Neural Networks*, *2*, pp. 887-891.

Dorigo, M., Maniezzo, V., & Colorni, A. (1992). *Positive Feedback as a Search Strategy.* Politecnico di Milano, Milan, IT.

Dorigo, M., Trianni, V., Sahin, E., Gross, R., Labella, T. H., Baldassarre, G., et al. (2004). Evolving Self-Organizing Behaviors for a Swarm-bot. *Autonomous Robots , 17*, 223-245.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. *Proceedings of the 12th International Conference on Machine Learning.* San Francisco, CA: Morgan Kaufmann Publishers.

Englebrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence.* West Sussex, U.K.: John Wiley and Sons, Ltd.

Feng, X., & Loparo, K. A. (1997). Active Probing for Information in Control Systems with Quantized State Measurements: A Minimum Entropy Approach. *IEEE Transactions on Automatic Control , 42* (2), 216-238.

Feng, X., Loparo, K. A., & Fang, Y. (1997). Optimal state estimation for stochastic systems: an information theoretic approach. *IEEE Transactions on Automatic Control , 42* (6), 771-785.

Fischer, K. H., & Hertz, J. A. (1993). *Spin Glasses.* Cambridge, U.K.: Cambridge University Press.

Haack, J., Akyol, B., Carpenter, B., Tews, C., & Foglesong, L. (2013). Volttron: An Agent Platform for the Smart Grid. *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems.* St. Paul, MN, USA.

Haus, J. W., & Kehr, K. W. (1987). Diffusion in regular and disordered lattices. *Physics Reports , 150* (5-6), 263.

Havlin, S., & Ben-Avraham, D. (1987). Diffusion in disordered media. *Advances in Physics , 36*, 695.

He, Z., Chiang, H. D., Li, C., & Zeng, Q. (2009). Fault-section Estimation in Power Systems Based on Improved Optimization Model and Binary Particle Swarm Optimization. *IEEE Power and Energy Society General Meeting*, (pp. 1-8).

Huang, K. (1987). *Statistical Mechanics* (Second ed.). Singapore: Wiley.

Janjarasjitt, S., & Loparo, K. A. (2008). An approach for characterizing coupling in dynamical systems. *Physica D: Nonlinear Phenomena , 237* (19), 2482-2486.

Janjarasjitta, S., & Loparo, K. A. (2008). An approach for characterizing coupling in dynamical systems. *Physica D: Nonlinear Phenomena , 237* (19), 2482-2486.

Kirkpatrick, S., Gelatt Jr., C. D., & Vecchi, M. P. (1983). Optimizing by simulated annealing. *Science , 220*, 671.

Kolacinski, R. (2003). A Behavioral Model-Based Approach to Constructing Models of Swarm Behaviors. *Workshop on Agent/Swarm Programming 2003 (WASP'03).* Cleveland, OH.

Kolacinski, R. M. (2003). *Swarming Algorithms to Enhance Connectivity and Performance of Mobile Ad hoc Networks (MANETs).* DARPA Final Report.

Kolacinski, R. (2011). Power System Model Identification via the Thermodynamic Formalism. *Proc. IFCA World Congress.* Milan, IT.

Kolacinski, R., Kanchanaharuthai, A., & Loparo, K. (2011). A General Mathematical Framework for Power System Security and Control. *Proc. IEEE EnergyTech 2011.* Cleveland, OH.

Kotsianis, S., & Kanellopoulos, D. (2006). Discretization Techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering , 32* (1), 47-58.

Kuntz, P., Layzell, P., & Snyers, D. (1997). A Colony of Ant-Like Agents for Partitioning in VLSI Technology. In P. Husbands, & I. Harvey (Ed.), *Proc. 4th European Conference on Artificial Life* (pp. 417-424). Cambridge, MA: MIT Press.

Liui, L., & Cartes, D. A. (2006). Particle Swarm Optimization for Automatic Diagnosis of PMSM Stator Fault. *Proceedings of the American Control Conference*, (pp. 3026-3031).

Lumer, E., & Faieta, B. (1994). Diversity and Adaptation in Populations of Clustering Ants. *Proc. 3rd International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3* (pp. 499-508). Cambridge, MA: MIT Press.

Mann, T. P. (2006). Numerically stable hidden Markov model implementation. 1-8.

Marinari, E., Parisi, G., & Ritort, F. (1994). Replica field theory for deterministic models (II): a non random spin glass with glassy behaviour. *J. Phys. A , 27*, 7647.

Marinari, E., Parisi, G., & Ritort, F. (1994). Replica field theoy for deterministic models: binary sequences with low autocorrelation. *J. Phys. A , 27*, 7615.

Mezard, M., Parisi, G., & Virasoro, M. (1986). *Spin glass theory and beyond.* Singapore: World Scientific.

Mirollo, R. E., & Strogatz, S. H. (1990). Synchronization of Pulse-coupled Biological Oscillators. *SIAM J. App. Math. , 50* (6), 1645-1662.

Mitchell, R., & McKim, J. (2002). *Design by Contract: by example.* Addison-Wesley.

Murthy, K. P., & Kehr, K. W. (1989). Mean first-passage time of random walks on a random lattice. *Physical Review A , 40* (4), 2082.

Oprisan, S. A., Holban, V., & Moldoveanu, B. (1996). Functional Self-Organization Performing Wide-Sense Stochastic Processes. *Phys. Lett. A , 216*, 303-306.

Palmer, D. W., Hantak, C. M., & Kovacina, M. A. (1999). Impact of Behavior Influence on Decentralized Control Strategies for Swarms of Simple, Autonomous Mobile Agents. *Proc. Workshop: Biomechanics Meets Robotics Modeling and Simulation of Motion.* Heidelberg, Germany.

Palmer, R. G. (1982). Broken ergodicity. *J. Adv. Phys. , 31*, 669.

Parpinelli, R. S., Lopes, H. S., & Freitas, A. A. (2002). An Ant Colony Algorithm for Classification Rule Discovery. In H. Abass, & R. Sarker (Ed.), *In Data Mining: A Heuristic Approach* (pp. 191-208). London, UK: Idea Group Publishing.

Parpinelli, R. S., Lopes, H. S., & Freitas, A. A. (2002). Data Mining with an Ant Colony Optimization Algorithm. *IEEE Trans. On Evolutionary Computation , 6* (4).

Pikovsky, A., Rosenblum, M., & Kurths, J. (2001). *Synchronization: A universal concept in nonlinear sciences* (Vol. 12). Cambridge, U.K.: Cambridge University Press.

Scott, D. W. (1979). On Optimal and Data-based Histograms. *Biometrika , 66* (3), 605-610.

Stein, D. L., & Newman, C. M. (2013). *Spin Glasses and Complexity.* Princeton, NJ: Princeton University Press.

Toulouse, G. (1977). Theory of the frustration effect in spin glasses I. *Communication Physics , 2*, 15.

Tyrell, A., Auer, G., & Bettstetter, C. (2006). Fireflies as Role Models for Synchronization in Ad hoc Networks. *Proc. 1st int. conf. on Bio inspired models of network, information and computing systems (BIONETICS 2006).* Cavalese, Italy: ACM, New York, NY.

Wilson, E. O. (1971). *The Insect Societies.* The Belkap Press of Harvard University Press.

Wodrich, M. (1996). *B. Sc. Thesis, Ant Colony Optimization.* South Africa: University of Capetown, Dept. of Electrical and Electronic Engineering.

Zhou, L., & Franklin, S. (1994). Character Recognition Agents. *rtificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems* (pp. 301-306). MIT Press.

3.0 Discovering the Communication Topology of Physical Systems

Network Routing: The core function of the information measures examined herein is to enable the innate structure of large-scale complex systems to be elicited. The underlying metaphor applied here, that these systems can be viewed as communication networks, suggests that the functions necessary for this elicitation may have analogs within the set of functions that must be performed for the effective operation of communication networks. Indeed, network discovery is a key function that must be performed for network routing. The correspondence between the elicitation of system structure and network discovery provides a solid foundation for the design and implementation of robust and reliable architecture and software components for system structure elicitation.

Network routing is the functioning core of every network. It implements the strategies used by network nodes to discover paths to send information/data from sources to destinations. Routing protocols specify which information to use for taking routing decisions, how to communicate the information between the nodes and how to build the routing table, which is the local database of routing information. Of particular interest are Mobile Ad-hoc NETworks (MANETs) that have a constantly evolving topology and thus a particular interest in network discovery.
In this report, we review the research on routing protocols and algorithms that have been specifically inspired from *collective behavior of insect societies*. Some of the collective behaviors of social insects are foraging, labor division, nest building/maintenance, cemetery organization and larval sorting.

Biologically inspired Algorithms: Biological systems consist of a set of distributed and autonomous units, such as ants, that produce system-level behaviors, looking for food, through local interactions which makes them adaptive, resilient, robust and scalable. These characteristics meet most of the necessary criteria for routing protocols in networks such as being robust, autonomous, adaptive, distributed and scalable. A large majority of research has been on the social behaviors observed in ants, termites and bee colonies. These algorithms are composed of autonomous distributed agents that follow a bottom-up approach based on the self-organizing abilities of the system. These characteristics are the basics of the *Swarm Intelligence*.

Ant and bee colonies have inspired a relatively large volume of research in network routing area. Specifically, the ability of ants to find the shortest path to the food source using pheromone trails has contributed to the development of *Ant Colony Optimization*.
Given the interest in many domains for MANETs, the literature in this domain is extensive. Therefore, we only focus on the most popular and effective routing protocols. The first notable algorithms were introduced in late 90's. From that point, the number of biologically inspired algorithms for MANET routing has grown tremendously.

Classification Features of Network Routing Protocols: The following is a classification of network routing protocols capturing the distinctive characteristics of the algorithms.

- Static vs. Dynamic
    - Static: using routing tables which are defined offline by network administrators according to some prior knowledge of the network.
    - Dynamic: updating routing tables online to reflect changes in the network state.
- Single-Path vs. Alternate- and Multi-Path
    - Single path: discover multiple routs, select the best
    - Multipath: discover, maintain and use multiple paths
- Flat vs. Hierarchical Organization
    - same hierarchical level vs. zone/cluster head
- Global vs. Local Representation
    - Local: using only local traffic and topology models. Swarm intelligence protocols usually use this topology because of its simplicity.

    - Global: each node maintains a complete database of the network to construct a network graph and apply shortest path algorithm on it. They converge quicker and scale better but require more power and memory.
- Deterministic vs. Probabilistic Decisions
    - Deterministic: using a deterministic rule selection algorithm to find the next node.
    - Probabilistic: using probabilistic selection rule which requires more computations and memory. However, they provide a certain level of randomness in rout selection which adds robustness and flexibility to the routing system.
- Constructive vs. Destructive Routing Table Making
    - Constructive: starting with empty set of routes and gradually adding routes until all the routing tables are constructed.
    - Deconstructive: starting by the assumption that the network is a fully connected graph and continuing by cutting the nonexistent path in the physical network.
- Proactive vs. Reactive Behavior
    - Reactive: paths are searched when required.
    - Proactive: information is maintained up-to-date all the time.
- Distributed vs. centralized routing
    - Centralized: discovery and maintenance of routing information controlled by a single node.
    - Distributed: more robust to network variations.
- Best-effort vs. QoS-aware routing
    - QoS-aware: protocols that can provide to the application routing services with quality guarantees.

General Framework for Swarm Intelligence based Routing Protocols: The general framework for swarm intelligence based routing protocols consists of five top-level modules and some submodules that implement the operation at the node router. The characteristics of these modules and their relationship are illustrated in Figure 3.1.
The routing protocol framework in this figure is the standard framework for mobile AdHoc networks. The main purpose of our project is to find the intrinsic communication topology of the network. Although network discovery is one of the goals in this framework, network routing is the main purpose. This difference shows itself in some circumstances such as losing a sensor. In our project, not only we would like to find an alternate path through, but also would like to add reconstructing information for the lost sensor which is different from the standard framework.

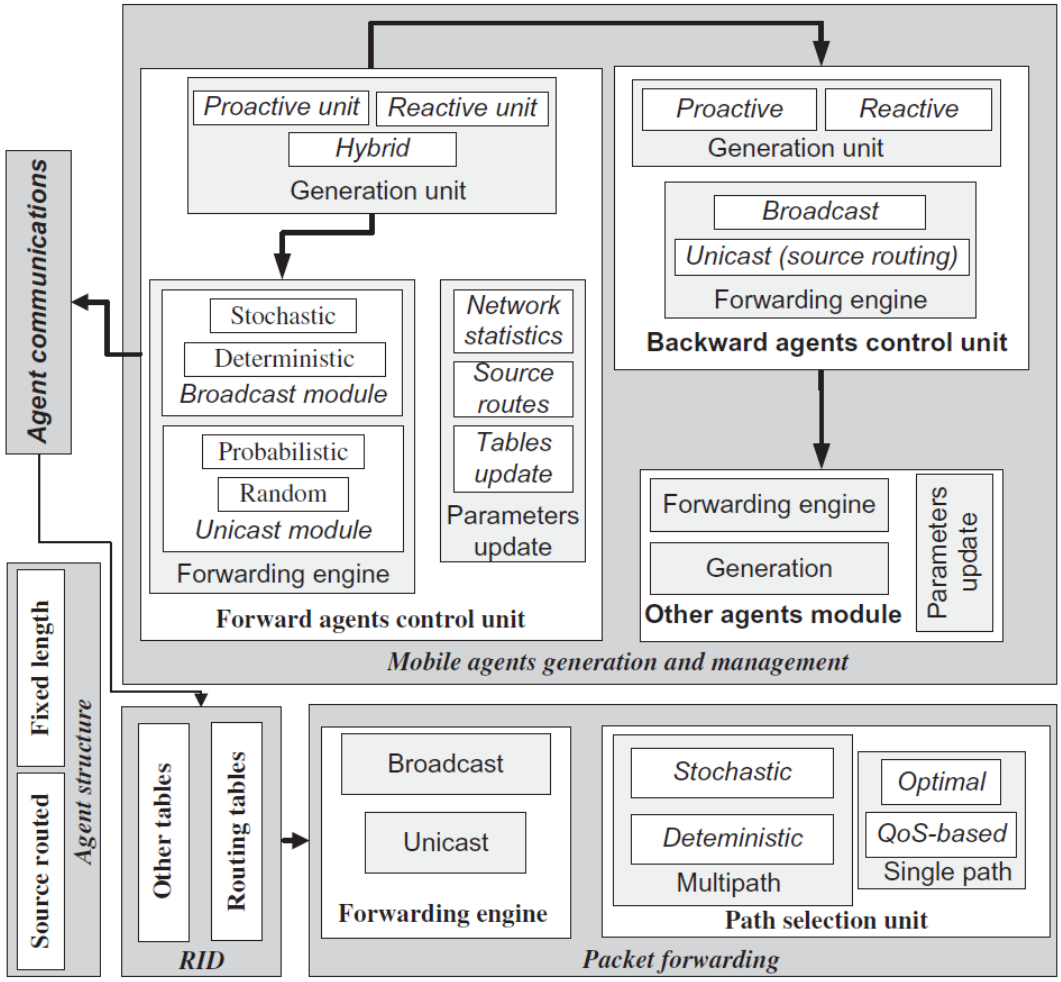


Figure 3.1: Swarm Intelligence based routing protocols framework

These top modules are:

- Mobile Agents Generation And Management
  - Forward agents: Launched by the source nodes to find the path to specified destination and collect routing information while travelling.
  - Backward Agents: The forward agents that have reached the destination. Travel back to the source node.
  - Other Agents Module: Additional agents used during discovery/update process such as sending agents *randomly* to explore the network.
- Routing Information Database (RID)
  - Routing Information Database; set of locally maintained structures.
- Agent Structure
- Agent Communications
  - The mobile agents share network data and collected path information using a stigmergic approach as in ACO. They have direct access to the data in the RID and implement the rule to update routing tables and statistics.
- Packet Forwarding:
  - Local forwarding of data packets

This framework provides a practical template for our problem. We may require expanding or modifying some of the above-mentioned modules to accommodate our project problem. Quality of Service (QoS) is one of the modules that require our attention. QoS is the ability to provide different priority to different applications

or users to guarantee a certain level of performance; for example, dedicating more communication network bandwidth for a particular process to achieve better channel capacity. Another module is the routing information database (RID). As we use different information structure in our network, maintaining the information tables may not be in our interest.

The other difference is the fact that most of these modules are based on Transmission Control Protocol (TCP) protocol. In this protocol, first the connections are established in a multi-step handshaking process and then data will be transferred. However, in our project, the connection is already established and we would like to discover it.
At last but not least, the network discovery in our project will be done in a wired network. This framework is the standard routing protocol for MANETs, which are a class of wireless mobile networks. They intend to minimize the transmission by minimizing the bandwidth. Therefore, this framework has a handful of constraints on discovery protocol that are not operable on what we are conducting in our project.

Routing Algorithms for Mobile AdHoc Networks: Mobile AdHoc Networks (MANETs) are a class of wireless mobile networks. The nodes are all mobile and they can enter and leave the network at any time and change the network configuration. They communicate with each other via wireless connections that due to mobility can constantly be established and broken. Data packets are transmitted from node to node. Other characteristics of MANETs are their shared channel, short battery lifetime, low bandwidth and distributed multi-hop forwarding. The following is a list of some of the routing algorithms for MANETs.

- GPS/Ant-Like Algorithm (GPSAL)
  - Camara and Loureiro, 2000.
  - Location-based routing algorithm and is one of the first algorithms developed based on ACO for MANETs.
  - Assume that all the mobile hosts participating in the MANET are equipped with GPS and send their approximate position to the host to reduce the number of routing messages.
  - Updates in the routing tables are then exchanged between hosts.
- Accelerated Ants Routing (AAR)
  - Matsuo and Mori, 2001.
  - The ants are equipped with a stack where the last n visited nodes are stored.
- Ant Colony Based Routing Algorithm (ARA)
  - Gunes et al, 2002.
  - Reactive. Supports multiple path routing. There are forward ants and backward ants.
- Ant-AODV
  - Marwaha et al, 2002
  - Extended by proactive updating of the routing tables based on uniform ants.
- Probabilistic emergent Routing Algorithm (PERA)
  - Baras and Mehta, 2003.
  - Reactive. Capable of multiple paths discovery which is helpful in quick recovery from link failures. Data packets are routed over the single best path available.
- Mobile Ant-Based Routing (MABR)
  - Heissenbüttel and Braun, 2003
  - Proactive, for large-scale MANETs. It uses the geographical partitioning of the node area and also the pheromone exploiting geographical addressing.
- Termite
  - Roth and Wicker, 2003.
  - Hybrid, paths are discovered on-demand, i.e. reactive, but their goodness is implicitly sampled by data packets in a proactive fashion. Forward ants are unicast and follow a random walk. Backward ants are also routing stochastically.
- AntHocNet
  - Di Caro et al, 2004.

  - Hybrids, reactive because the nodes start gathering information when requested by a local traffic session while proactively keep the routing information up-to-date for the entire duration of communication.
- Ad-Hoc Networking with Swarm Intelligence (ANSI)
  - Rajagopalan and Shen, 2005.
  - Reactive. The ants are deterministically flooded towards the destination to find or repair a route.

Routing Algorithms for Wireless Sensor Networks: Wireless Sensor Networks (WSNs) are a set of autonomous nodes equipped with sensing capabilities, wireless communication interfaces, limited processing and energy resources. They are used for cooperative and distributed monitoring of physical or environmental phenomena such as pressure, sound, temperature, etc. The nodes are usually statistically distributed in the area. They can also be mobile and capable of interacting with the environment. In a WSN, individual nodes have limited communication range and form an ad hoc network over a shared wireless medium.

The routing protocol requirements for WSNs are similar to those of routing protocols for MANETs. However, in WSNs there are restrictions on energy efficiency, nodes are usually static and in general the networks are larger. A list of routing algorithms for WSNs is provided below.

- Energy Efficient Ant-Based Routing (EEABR)
  - Camilo et al, 2006.
  - Proactive. Extends the network lifetime by reducing the communication overhead in path discovery using fixed-size agents and introducing energy and number of hops in the pheromone update rule.
- ACO-based quality-of-service routing (ACO-QoSR)
  - Camilo et al, 2006.
  - Reactive. Find the appropriate paths to send data which has total end-to-end delay less than a bounding value and energy residual ratio above certain threshold.
- Self-organizing data gathering for multi-sink sensor networks (SDG)
  - Kiri et al, 2007.
  - Aimed to achieve reliability and scalability in WSNs. They propose a multi-sink WSN in which the nodes can use an alternate sink in case of failure. Agents are generated by sink nodes only as backward ants to minimize the routing overhead.
- Ant-based service-aware routing algorithm (ASAR)
  - Sun et al, 2008.
- Many-to-One Improved Ant Routing (MO-IAR)
  - Ghasemaghaei et al, 2008.
- AntChain
  - Ding and Xiaoping Liu, 2004.
  - Assumes that each sensor node can directly reach every other node in the network and can directly communicate with the sink.
- Jumping Ant Routing Algorithm (JARA)
  - Chen et al, 2007.
- Energy-Delay ant-based (E-D ANTS)
  - Chen et al, 2008.
  - Find the route with minimum energy-delay product in order to maximize network lifetime and to provide a real-time data delivery service.
- Probabilistic, Zonal and Swarm-inspired system for Wildfire Detection (PZSWiD)
  - Ramachandran et al, 2008.
  - Cluster-based ACO-inspired system for wildfire detection.
- Ant-aggregation
  - Misra and Mandal, 2006.

Discovering the Intrinsic Communication Topology of a Physical System: To formulate a distributed and self-organizing implementation of the basic sensor network functions, the physical systems and the available observations of their associated phenomena are considered from a foraging perspective. The principle focus of the effort is to elucidate the intrinsic communication topology within a physical system. The current activity investigates the efficacy of Ant Colony Optimization (ACO) techniques for discovering this information connectivity.

Next we discuss the application of ACO techniques to the discovery of the intrinsic information topology of systems. The mathematical underpinnings of this approach are presented first. Following the introduction of an appropriate mathematical framework, the application of this approach to an exemplary network of system for the purpose of elucidating its intrinsic communication topology is presented. Finally, conclusions and plans future work are discussed.

Physical System Observations: Consider the network in Figure 3.2 below. This figure illustrates the observation of a physical system from foraging perspective.

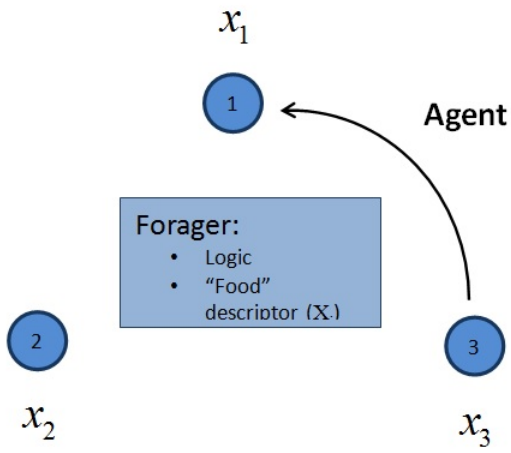


Figure 3.2: Observation of Physical System from a Foraging Perspective

In this figure, x_i is a time series that is the partial observation of the physical system at node i. When an agent goes from one node to another, it carries time series data x_i from its home node to its destination node. The food is defined as the "similarity" between these the data at the two nodes. When the agent carries information, it looks for data that contains the "same information". Moreover, it wants to preserve the dynamics of the system.

To capture this similarity, we calculate the "Cross Correlation" between the two time series at the home and destination nodes. Cross correlation is a measure of similarity between time series and is a standard method of estimating the degree to which two time series are (linearly) related. In large and complex systems, calculating cross correlation requires extensive memory and can be computationally inefficient. As the data of interest in the target application is typically nonstationary, these calculations can be performed in windows that are temporally related. Appropriate windows are identified for each time series and the data contained within these windows is used to calculate the cross correlation between these windows instead of the whole time series as shown in Figure 3.3.

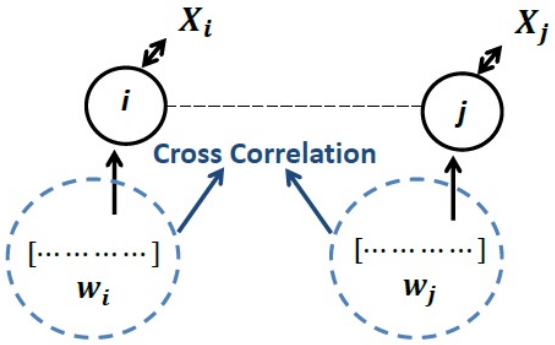


Figure 3.3: Calculating Cross Correlation between two nodes in the system

The cross correlation between two windows is related to the strength of the interconnection between them, i.e. the higher the cross correlation, the stronger the connectivity between two windows. In order to give a number to the strength of interconnection, we suggest the use of the "Pearson Product-Moment Correlation Coefficient", ρ. This is a measure of linear correlation between two variables and has a value between +1 and -1, with 1 being positive linear dependence, 0 as no correlation, and -1 as negative linear dependence. The absolute value of the correlation coefficient between the windows of two nodes is the strength of the connection that is attached to the path. The stronger this connection, the more agents will take this path.
In the ACO algorithm, ants start from the source node and take the shortest route to the food and bring it back to the source node. In an adaptation of the ACO algorithm in this project, the ants take the route with stronger connectivity, the route with higher $|\rho|$. As they move, they lay pheromone on the route. Therefore, more ants will take this path while discovering the network. After all the ants return to the source node, we will have a primary understanding of the intrinsic communication topology of the network.

Discovering Intrinsic Communication Topology: In this section, we describe our approach in more details for a simple linear system. We design an exemplary linear system to discover its intrinsic communication topology using our technique.

Exemplary Linear System: The exemplary linear system examined is defined in equation given below. This linear system has a noise process, w_k, with defined correlation structure, D_p.

$$X_{k+1} = AX_k + D_p w_k$$
$$Y_k = X_k$$

Here, $X_k \in \mathbb{R}^n$, n is the number of nodes/states, $D_p \in \mathbb{R}^{n \times n}$ is the correlation matrix, and $w_k \in \mathbb{R}^n$ is a white noise process. The exemplary network considered in this example has 10 nodes/states and is defined by the matrices A and D_p and noise sequence w_k given below.

Preliminary Results: We select different window sizes for Y_k, where $k = 1,...,10$, and calculate their correlation coefficient. We choose these windows to be aligned to give us the most accurate results comparing to selection of the entire times series. We then calculate the correlation coefficient between those windows. The correlation coefficients for windows shown in Figure are given in the $|\rho|$ matrix below.

$$|\rho| = \begin{bmatrix} 1 & 0.47 & 0.81 & 0.68 & 0.88 & 0.58 & 0.83 & 0.82 & 0.66 & 0.68 \\ 0.47 & 1 & 0.67 & 0.37 & 0.55 & 0.64 & 0.57 & 0.65 & 0.38 & 0.71 \\ 0.81 & 0.67 & 1 & 0.77 & 0.67 & 0.83 & 0.87 & 0.74 & 0.52 & 0.76 \\ 0.68 & 0.37 & 0.77 & 1 & 0.55 & 0.73 & 0.74 & 0.52 & 0.38 & 0.65 \\ 0.88 & 0.55 & 0.67 & 0.55 & 1 & 0.49 & 0.77 & 0.90 & 0.76 & 0.74 \\ 0.58 & 0.64 & 0.83 & 0.73 & 0.49 & 1 & 0.69 & 0.52 & 0.38 & 0.66 \\ 0.83 & 0.57 & 0.87 & 0.74 & 0.77 & 0.69 & 1 & 0.74 & 0.62 & 0.74 \\ 0.82 & 0.65 & 0.74 & 0.52 & 0.90 & 0.52 & 0.74 & 1 & 0.58 & 0.75 \\ 0.66 & 0.38 & 0.52 & 0.38 & 0.76 & 0.38 & 0.62 & 0.58 & 1 & 0.51 \\ 0.68 & 0.71 & 0.76 & 0.65 & 0.74 & 0.66 & 0.74 & 0.75 & 0.51 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 0.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4 \end{bmatrix}$$

$$D_p = \begin{bmatrix} 1 & 0.5 & 1.5 & 1 & 1.5 & 2 & 3.5 & 1 & 2 & 1 \\ 0.5 & 2.5 & 0.5 & 1 & 2.5 & 0.5 & 0.5 & 4 & 1 & 1.5 \\ 1.5 & 0.5 & 3 & 3.5 & 3 & 1.5 & 3.5 & 3.5 & 4 & 3.5 \\ 1 & 1 & 3.5 & 4 & 1 & 3.5 & 3 & 0.5 & 4 & 0.5 \\ 1.5 & 2.5 & 3 & 1 & 2.5 & 3 & 2.5 & 2.5 & 3 & 1.5 \\ 2 & 0.5 & 1.5 & 3.5 & 3 & 2 & 1.5 & 1.5 & 1.5 & 3 \\ 3.5 & 0.5 & 3.5 & 3 & 2.5 & 1.5 & 2.5 & 2.5 & 4 & 0.5 \\ 1 & 4 & 3.5 & 0.5 & 2.5 & 1.5 & 2.5 & 2.5 & 4 & 3.5 \\ 2 & 1 & 4 & 4 & 3 & 1.5 & 4 & 4 & 0.5 & 1 \\ 1 & 1.5 & 3.5 & 0.5 & 1.5 & 3 & 0.5 & 3.5 & 1 & 2 \end{bmatrix}$$

$$w_k \approx N(0,1)$$

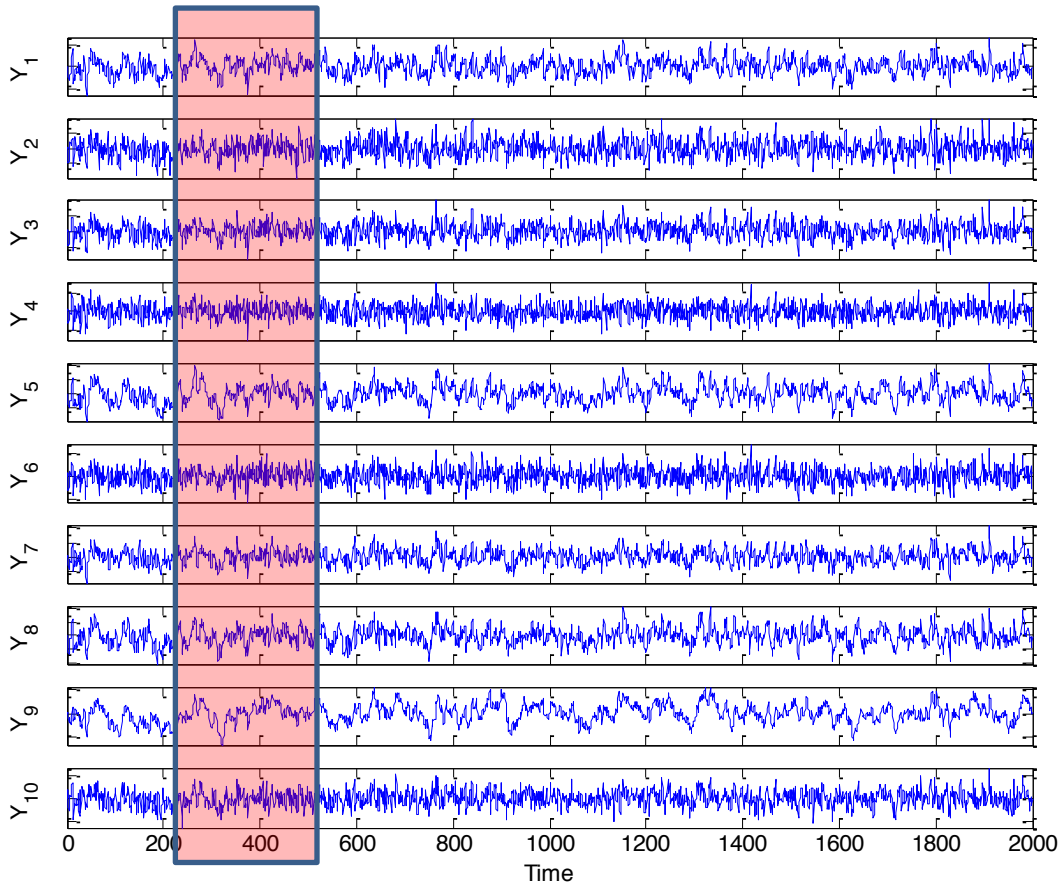
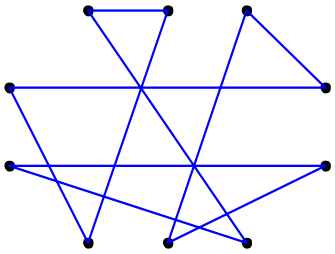


Figure 3.4: Aligned Windows with the size of 300 data points for each window

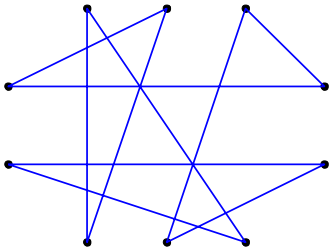
The ACO algorithm is simulated using MATLAB and we select 200 ants/agents and simulate the above network for 50 iterations. As mentioned in Section previously, the correlation coefficient between each state/node are selected as the food on that path connecting the states/nodes. In order to incorporate this assumption into the code, we set

$$d_{ij} = \frac{1}{\rho_{ij}}$$

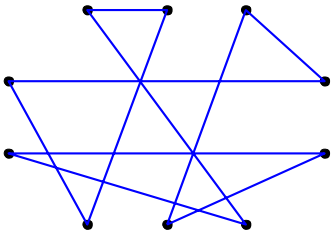
where d_{ij} is the distance between states/nodes (i, j) and ρ_{ij} is the correlation coefficient between the selected windows of those states/node. If the correlation coefficient, or food here, is high, the distance between the nodes is short. Therefore, more ants/agents will take this path to find the shortest path to the food. Please note that we should set $d_{ii} = 0$ as the distance of each node to itself is zero. Figure 3.5 shows the results for our exemplary network for aligned windows of the same size. The windows size is 300 and they are shifted to the right side as shown in Figure 3.6.



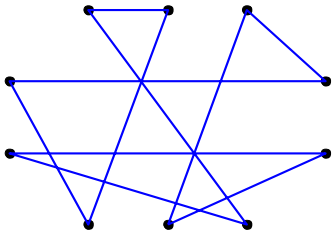
a) Far Left Windows



(b) Middle Left Windows



(c) Middle Right Windows



(d) Far Right Windows

Figure 3.5: Intrinsic Communication Topology of Exemplary Network for Shifted Aligned Windows of the same size

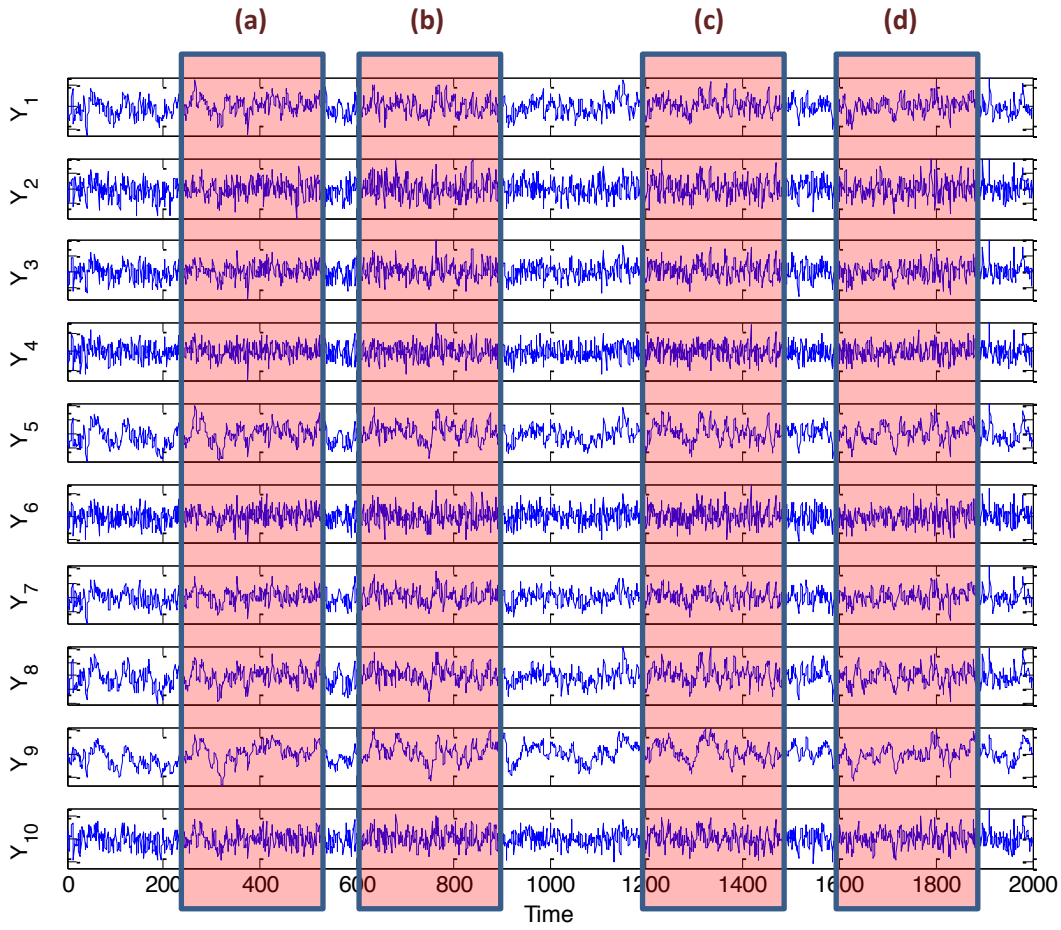


Figure 3.6: Aligned Windows with the same size of 300

As we can seen, the intrinsic topology is different for the same network with the same window size, but with different window locations. However, the information path between some nodes is the same for each network. This reinforces the existence of an *information dependency* between those nodes. Table 3.**4** lists the shortest information path for each of the above networks.

Table 3.4: Shortest Information Distance for the Networks of Figure

Networks	(a)	(b)	(c)	(d)
Shortest Information Distance	13.3390	12.9171	12.7819	13.0430

As we can see from Table 3.**4**, the shortest information distance is given when the windows are selected in the middle right of the time series.

Next we investigate what happens if we keep the position of the windows, but change their size. This is shown in Figure for six aligned windows with sizes from 100 to 2000.

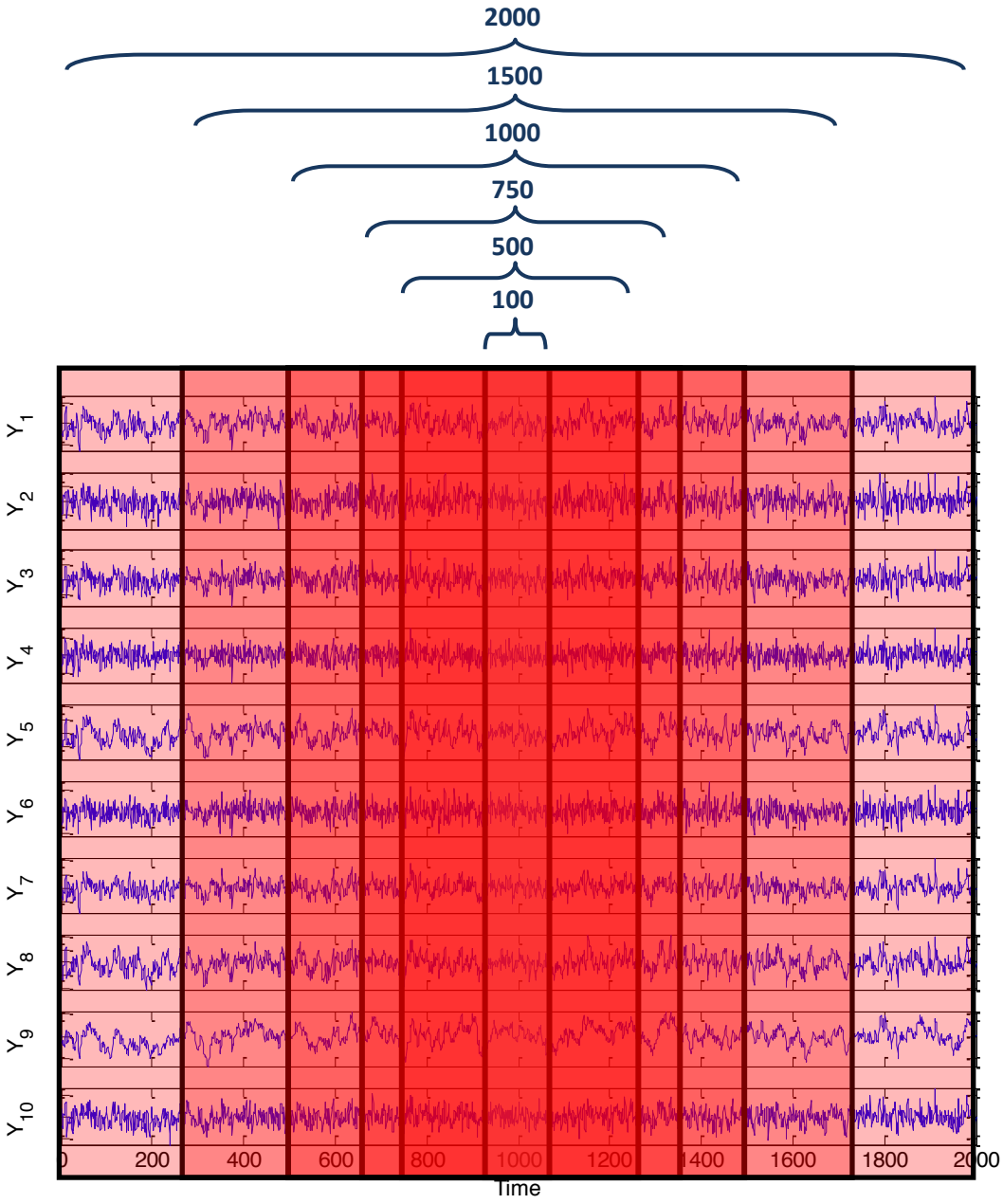
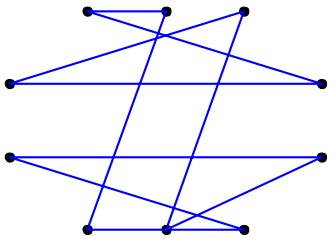
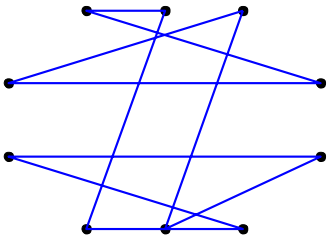


Figure 3.7: Aligned Windows with Different Sizes

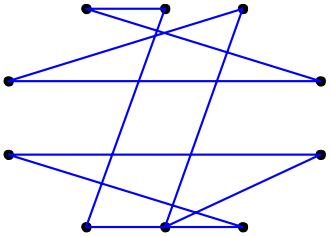
The simulation results for these networks are shown in Figure 3.8.



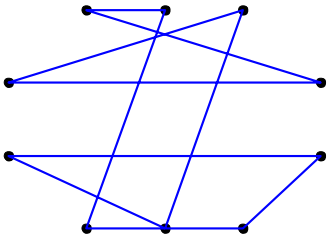
(a) Window Size = 100



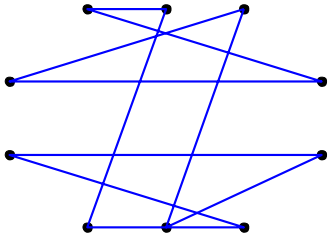
(b) Window Size = 500



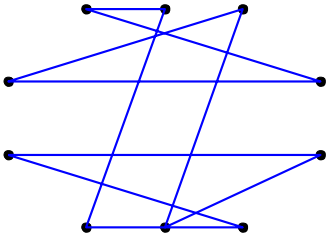
(c) Window Size = 750



(d) Window Size = 1000



(e) Window Size = 1500



(f) Window Size = 2000

Figure 3.8: Intrinsic Communication Topology of Exemplary Network for Aligned Centered Windows of Different Sizes

Table 3.5: Shortest Information Distance for Networks of Figure 3.8.

Window Size	100	500	750	1000	1500	2000
Shortest Information Distance	13.1409	13.0353	13.2026	13.0745	13.1384	13.0596

Although the intrinsic topology for all the networks is the same, their shortest information distance is different because the correlation coefficient for every network is different as the windows are not the same. The shortest information distance is for the window size of 500.

We repeated the simulation for different aligned windows with different sizes.

Figure 3.9 illustrates the information connectivity that is shared among the majority of the resulted topologies. We can say that this is the intrinsic communication topology for our exemplary network.

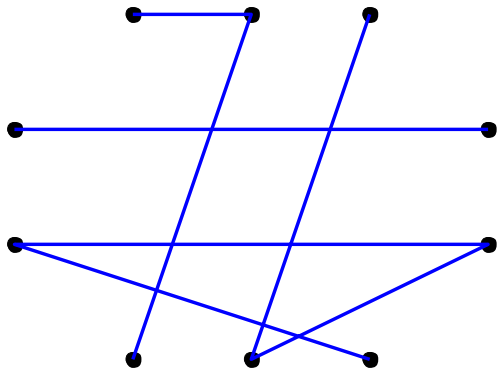


Figure 3.9: Intrinsic Communication Topology of Our Exemplary Network

Classification of the Operational State of Power Generation Plants: Because of the growing need to understand the data within diverse fields, there is a growing need for "classification" methods. Moreover, variation in needs across disciplines and application domains has created numerous bases for classification and a number of different theoretical approaches to classification exist. Moreover, the growing interest in "data analytics" and "Big Data" is driving the development of a rapidly increasing collection of techniques and approaches as part of a larger effort in developing machine learning techniques.

One of the most basic, and hence most generally applicable, approaches are known as *clustering algorithms*, wherein the basic organization of data is elicited by identifying subsets of data that belong together in clusters. In this report, we briefly review some algorithms along with their summarized details. The algorithms are categorized into 5 categories based on their nature, underlying concept and/or possible result clusters.

Centroid-based clustering: The main goal for this algorithm is to categorize T data into specified number of clusters, N about an appropriate centroidal value, typically a mean. It is well-known that the optimal solution of this goal is an NP-hard problem which cannot be solved in limited timeframe. As a result, heuristic algorithms to obtain near optimal solution have been developed. Of this class of approaches, the K-mean clustering algorithm is the most common and basic algorithm (Hartigan & WongReviewed , 1979; Lloyd, 1982). The K-mean clustering algorithm constructs a specific number of clusters N as follows:

Step 1. Initialized the centroids of cluster $\left\{m_1^{(0)}, m_2^{(0)}, \dots, m_N^{(0)}\right\}$.

Step 2. Assign each data into a cluster in which the distance between current data and the centroid is closest. Mathematically, in i^{th} iteration, data x_l will belong to the cluster $c_l^{(i)}$, which is computed by $c_l^{(i)} = \text{argmin}_{k=1,\dots,N}\, d\left(x_l, m_k^{(i-1)}\right)$. Note that $d(\cdot,\cdot)$ is some specified distance measure that is not necessarily a Euclidian distance.

Step 3. Recompute new centroids for all clusters based on new members per the formula $m_k^{(i)} = \frac{\sum_{l=1}^{T} x_l \square_{k,c_l^{(i)}}}{\sum_{l=1}^{T} \delta_{k,c_l^{(i)}}}$ where δ is Kronecker delta.

Step 4. Iterate Step 2 and Step 3 until algorithm converges.

There are two methods frequently used to pick the initial centroids (Hamerly & Elkan, 2002). The first method picks N centroids randomly from the data. In contrast, the other method randomly assigns each datum to a cluster then computes centroid of each cluster based on its members. A variation on this second approach extends the assignment of data to new clusters that can be generated on the basis of nonparametric Bayesian induction methods (Teh, 2010) and thus permits clustering without the a priori specification of the number of clusters.

Another variation on centroid-based clustering that considers the case where individual data can belong to multiple clusters with a differing degree rather than crisply classifying each datum as a member of a single cluster. A membership function that takes values between 0 and 1 are used to quantify the degree to which the data belong to each of the clusters. The degree of belonging (to a particular cluster) is proportional to a selected measure of nearness between the data and the cluster of interest. The most common algorithm embodying this idea is the C-mean clustering algorithm. This algorithm is similar to the K-mean clustering algorithm except that the C-mean clustering algorithm uses a fuzzy (as opposed to crisp) notion of clustering (Nock & Nielsen, 2006). The C-mean clustering algorithm constructs as specific number of clusters N for a given fuzzy parameter m as follows:

Step 1. Initialize the degree of belonging $\left\{w_1^{(0)}(l), w_1^{(0)}(l), \dots, w_1^{(0)}(l)\right\}$ $\forall l = 1, \dots, T$. Compute initial centroid of cluster by $m_k^{(0)} = \frac{\sum_{l=1}^{T}\left(w_k^{(0)}(x)\right)^m x_l}{\sum_{l=1}^{T}\left(w_k^{(0)}(x)\right)^m}$ for $k = 1, \dots, N$.

Step 2. For all $l = 1, \dots, T$ and $k = 1, \dots, N$, assign degree of belonging using $w_k^{(i+1)}(l) = \frac{1}{\sum_{p=1}^{N}\left(\frac{d\left(m_k^{(i)}, x_l\right)}{d\left(m_p^{(i)}, x_l\right)}\right)^{\frac{2}{m-1}}}$ where $d(\cdot,\cdot)$ is a selected distance measure.

Step 3. Reassign a new centroid for all cluster based on degree of belonging from all data using the same equation as in Step 1.

Step 4. Iterate over Step 2 and Step 3 until algorithm converges.

Observe that centroid-based approaches use centroids and a notion of distance, i.e. a “radius”. Both of these quantities are dependent on the cluster member. As a result, the shapes of clusters depend on the

operative distance measure and their locations depend on the centroid. The choice of distance measure is fraught. In many cases, the data may actually live in a higher dimensional space than is being observed. That is, the observations are commonly projections onto a lower dimensional space. Moreover, the observations may confound multiple dimensions. If the underlying manifold on which the data truly lie is nonlinear, this projection may manifest irregular structure (particularly if the underlying manifold is nonsmooth) or, if the dimensions are confounded in the observations, the true structure may be occluded. If the data have an irregular shape or organization, this class of clustering algorithm can have a difficulty capturing their hidden structure.

Hierarchical clustering: Unlike the K-means and C-means clustering algorithms that rely on a notion of a cluster centroid, hierarchical clustering approaches construct clusters on the basis of connectivity. The underlying concept is based on the idea that data that are relatively close (in some chosen sense) should belong to the same cluster while data that are relatively far should belong to different cluster. Note that the clusters formed in this fashion can have peculiar shape since it is not restricted by centroids and distance measure like the K-mean and C-mean clustering algorithms. Hierarchical clustering does not assign data to clusters but it rather it constructs a hierarchy as shown in the form of a dendrogram in Figure 3.10.

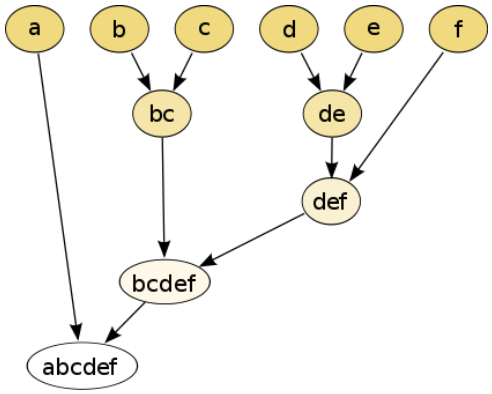


Figure 3.10: Dendrogram from data set $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}\}$ based on their distances. The bottom (root) node is where all data are in the same cluster while the top (leaf) node is where all are data are in their own cluster.

Each level of dendrogram can be considered a potential partition of the data into clusters. In general, the user manually selects the clusters (i.e. levels). The selection of clusters can be automated if suitable criteria and/or procedures can be specified a priori. Similar to the K-means (and C-means) algorithms, the result of the clustering procedure can vary with the distance measure applied. In addition, hierarchical clustering also defines the distance between two clusters, called the linkage. This linkage is used during the construction of dendrogram and, as a result, the linkage criterion used greatly affects the resultant clustering. Two most commonly applied linkage criteria are *single linkage* and *complete linkage* that are defined as follows:

- Single linkage determines the distance between two clusters by the closest distance. In other words, the distance is determined from two data, one from each cluster, that are nearest to one another. Mathematically, the distance between cluster X and cluster Y, $D(X,Y)$ is $D(X,Y) = \min_{x\in X, y\in Y} d(x,y)$ where $d(\cdot,\cdot)$ is a distance measure.
- Complete linkage is the opposite of single linkage. The distance between two clusters is determined from two data, one from each cluster, that are farthest from one another. Mathematically, the distance between cluster X and cluster Y, $D(X,Y)$ is $D(X,Y) = \max_{x\in X, y\in Y} d(x,y)$ where $d(\cdot,\cdot)$ is a distance measure. Clearly, inverse distance may used to determine the closeness between two clusters.

Furthermore, the dendrogram can be constructed in two ways, *agglomerative* and *divisive*. Divisive methods begin with one cluster and continually divide the cluster(s) until all data belong to their own cluster. In

contrast, agglomeration methods begin with singleton elements and continually merge clusters until all data belong to single cluster. In general, hierarchical clustering approaches are very slow for a large data set since its complexity is $O(n^3)$. There exist efficient algorithms including SLINK (Sibson, 1972) for single-linkage and CLINK (Defays, 1977) for complete-linkage clustering where both have a complexity of $O(n^2)$.

While this approach can provide more flexibility with respect to the shapes of individual clusters compared to centroid-based methods, the construction of clusters from dendrograms is non-trivial. Even if a criterion such as Davies–Bouldin index or Dunn index is applied (Dunn, 1973; Davies & Bouldin, 1979), choosing an appropriate threshold to obtain the desired clustering can be challenging. In addition, the dendrogram must be stored in memory before the clusters are finalized and this can be infeasible if the number of data to be clustered is large. Finally, the complexity of even the efficient algorithms noted above is still relatively high, e.g., $O(n^2)$ and thus may be computationally prohibitive when dealing with large data set.

Probability Distribution-based clustering: This class of algorithm uses statistics and probability theory for cluster construction. Distribution-based clustering assumes that data belong to a probability distribution that is the weighted sum of different probability distributions (Hastie, Tibshirani, & Friedman, 2001). These approaches apply expectation maximization (EM) algorithms to determine the parameters of this combined probability distribution. Finally, the data are assigned to a cluster based on the likelihood function for each sub-distribution with the identified parameters. Mathematically, This algorithm assumes that data belong to distribution of the form $\sum w_i f_i(X;\theta_i)$ where $\sum w_i = 1$ and $f_i(\cdot;\vec{\theta}_i)$ are probability distributions with associated parameters vectors, $\vec{\theta}_i$. Note that $f_i(\cdot;\vec{\theta}_i)$ is not necessarily restricted to having the same form and same number of parameters for all i. In general, however, it is typically assumed that each sub-distribution $f_i(\cdot;\vec{\theta}_i)$ has the same form for computational purposes. After the form of $\sum_{i=1}^{N} w_i f_i(X;\vec{\theta}_i)$ is determined the EM algorithm is applied to this mixture distribution using the data $\{x_1,\dots,x_T\}$ to determine w_i and $\vec{\theta}_i$ for all i. Finally, the cluster of the l^{th} datum is determined via maximum likelihood, i.e. $c_l = \text{argmax}_{k=1,\dots,N} f_{\square}(x_l,\vec{\theta}_i)$. The most common method is based on Gaussian mixture models since Gaussian mixtures can be used to approximate a wide-range of distributions. The main drawback to this assumption is that one may need to specify a large number component distributions (number of clusters) to the mixture and that can lead to an over-fitting problem. For the best performance, users need to manually specify the form of the distribution directly from the available data and this may not be feasible for high-dimensional data and, especially, data with unknown structure.

Density-based clustering: From the concept of probability distribution-based clustering, probability distribution can be sliced into horizontal layers (parallel to attributes plane) (Kriegel, Kröger, Sander, & Zimek, 2011) a la level sets. Under this clustering paradigm, a cluster can be seen as regions (in the attributes plane) of higher density while the sparse regions can be considered as outliers or noise. Thus, this idea is more robust to noise compared to probability distribution-based clustering since it does not need to incorporate outlier into any clusters. However, the sparse regions are necessary to distinguish between clusters and thus density-based clustering methods are not appropriate in cases where cluster overlap or share boundary points. Note also that density-based clustering is not limited to a probabilistic notion of density since dense regions can be defined relative to different measures of density in the attributes plane. The most common algorithm in this category is *Density Based Spatial Clustering of Applications with Noise* (*DBSCAN*) (Ester, Kriegel, Sander, & Xu, 1996). DBSCAN relies on the definition of ε-*neighborhoods* and the notion of *density-reachability* to construct the clusters. The following definitions and notions provide the core elements of the DBSCAN algorithm:

- The ε-*neighbourhood* of any point p is a set $N_\varepsilon(p) = \{y \in D : d(x,y) < \varepsilon, y \neq x\}$ where D is the set of all data and $d(\cdot,\cdot)$ is a distance measure.
- A point p is said to be *directly density-reachable* from a point q w.r.t. ε and minimum points, $n_{\min}$ if $p \in N_\varepsilon(q)$ and $\text{Card}(N_\varepsilon(q)) \geq n_{\min}$.

- A point p is *density-reachable* from a point q wrt. ε and $n_{\min}$ if there is a chain of points $p_1, \dots, p_n$, where $p_1 = q$ and $p_n = p$ such that each p_{i+1} is directly density-reachable from p_i.
- A point p is *density-connected* to a point q w.r.t. ε and $n_{\min}$ if there is a point o such that both, p and q are density-reachable from o w.r.t. ε and $n_{\min}$.
- A *cluster* C w.r.t. ε and $n_{\min}$ is a non-empty subset of D satisfying the following conditions:
  - *Maximality*: $\forall\, p, q$: if $p \in C$ and q is density-reachable from p w.r.t. ε and $n_{\min}$, then $q \in C$.
  - *Connectivity*: $\forall\, p, q \in C$: p is density-connected to q w.r.t. ε and $n_{\min}$.
- Let $C_1, \dots, C_k$ be the clusters of the data set D w.r.t. parameters ε and $n_{\min}$. Then we define the *noise* as the set of points in D not belonging to any cluster C_i.

With these definitions, pseudo code for DBSCAN is as follows:

```
FUNCTION DBSCAN(D, ε, n_min)
    C = 0
    FOR P = 1 to |D|
        IF P is not visited
            mark P as visited
            NeighborPts_P = regionQuery(P, ε)
            IF sizeof(NeighborPts_P) < n_min
                mark P as NOISE
            ELSE
                C = next cluster
                expandCluster(P, NeighborPts_P, C, ε, n_min)

FUNCTION expandCluster(P, NeighborPts_P, C, ε, n_min)
    add P to cluster C
    FOR Q in NeighborPts_P
        IF Q is not visited
            mark Q as visited
            NeighborPts_Q = regionQuery(Q, ε)
            IF sizeof(NeighborPts_Q) >= n_min
                NeighborPts_P = NeighborPts_P joined with NeighborPts_Q
        IF Q is not yet member of any cluster
            add Q to cluster C

FUNCTION regionQuery(P, ε)
    return all points within P's ε -neighborhood (including P)
```

The main advantage of DBSCAN is that the number of clusters need not to be specified a priori, the shape of clusters can also be arbitrary and it is robust to noise with selection of appropriate ε and $n_{\min}$. As noted above, DBSCAN has issues with overlapping clusters or shared *border points*. Also, if the clusters have different data densities, the parameterized offered by ε and $n_{\min}$ is not sufficient to construct the desired clusters. In addition, this method is subject to the curse of dimensionality in high dimensional applications during determination of appropriate values of ε. OPTICS (Ankerst, Breunig, Kriegel, & Sander, 1999) was developed specifically to address the issues of sensitivity to different data densities in different clusters. In addition, OPTICS also significantly reduces the criticality of properly selecting the parameter ε. The class of clustering algorithms has shown great potential in many applications and, as a result, is currently an area of active research that has produced significant developments recently (Campello, Moulavi, & Sander, 2013; Roy & Bhattacharyya, 2005; Breunig, Kriegel, Ng, & Sander, 1999; Achtert, Böhm, & Kröger, DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking, 2006; Achtert, Böhm, Kriegel, Kröger, Müller-Gorman, & Zimek, Finding hierarchies of subspace clusters, 2006; Achtert, Böhm, Kröger, & Zimek, Mining Hierarchies of Correlation Clusters, 2006; Achtert,

Böhm, Kriegel, Kröger, Müller-Gorman, & Zimek, Detection and Visualization of Subspace Cluster Hierarchies, 2007; Schneider & Vlachos, 2013).

Based on the underlying concept of clustering used by this approach, density-based clustering is demonstrably more flexible and robust than probability distribution-based clustering. This results from the fact that density-based clustering provides a mechanism to identify noise and to reject the associated data as outliers. Many implementations of this class of algorithm are relatively fast, $O(n \log n)$ and the number of cluster need not be specified a priori.

Biologically inspired clustering: These clustering algorithms are primarily based on the eusocial behavior of ants and provide decentralized algorithms that can be used to identify clusters within a given data set. Algorithms based on ant behaviors can be classified into two categories. First category consists of Ant Colony Optimization (ACO)-based approaches (Shelokar, Jayaraman, & Kulkarni, 2004; Kao & Cheng, 2006) and second category contains grid-based sorting approaches.

ACO-based methods are algorithms based on the foraging behaviors of ant swarms to solve optimization problems due to the tendency of ant swarms to find "shortest path" solutions between food sources and their nests. The introduction of ACO methods to clustering problems is due to the fact that clustering problems can be formulated as optimization problem. Generally, the clustering problem is formulated as an optimization problem within the ACO framework. That is, the clustering problem is restated as

$\min F(w, m) = \sum_{j=1}^{K} \sum_{i=1}^{T} w_{ij} \left\| x_i - m_j \right\|_2^2$

subject to

$$\sum_{j=1}^{K} w_{ij} = 1, \quad i = 1, \dots, T$$
$$\sum_{i=1}^{T} w_{ij} \geq 1, \quad j = 1, \dots, K$$

Where

- x_i is the i^{th} datum for $i = 1, \dots, T$
- m_j is a centroid of j^{th} cluster for $j = 1, \dots, K$
- $w_{ij} = \begin{cases} 1 & \text{if object } i \text{ is contained in cluster } j \\ 0 & \text{otherwise} \end{cases}$ for $i = 1, \dots, T$ and $j = 1, \dots, K$

These algorithms are agent-based and the basic algorithm underpinning this class of approaches is as follows (Shelokar, Jayaraman, & Kulkarni, 2004). A pheromone matrix is defined as $Z_{T \times K}$ where each element of $Z_{T \times K}$, τ_{ij}, represents the pheromone (weight) deposition associated with object i and cluster j. Initially, τ_{ij} is assigned a small random number. Each agent uses normalized l^{th} row of $Z_{T \times K}$ (i.e. $Z_{l \times K}$) along with uniform random variable to assign the cluster for l^{th} datum as its solution. Each agent will repeat the process over all of the data until it gets a full solution (i.e. cluster assignment). The best solutions, relative to the above cost, among all agents will be compared with best solution from the previous iteration and will replace the old solution if it is better. Finally, all agents update the pheromone matrix $Z_{T \times K}$, where the pheromone strength is proportional to the quality of their solutions. Also, at the end of each iteration, a pheromone evaporation rate may be applied to encourage exploration of new solutions. As can be seen, the main drawback of this approach is the same as that of the K-mean clustering algorithm. Namely, the number of clusters must be specified a priori in order to formulate valid optimization problem.

In contrast to ACO-based methods, alternative approaches uses sorting behaviors of eusocial insects (e.g., larval sorting, cemetery organization) in a 2-dimensional grid (Lumer & Faieta; Yang & Kamel, Clustering ensemble using swarm intelligence, 2003; Vizine, De Castro, Hruschka, & Gudwin, 2005; Yang & Kamel, An aggregated clustering approach using multi-ant colonies algorithms, 2006). The basic concept in this approach is fairly simple. First, these algorithms construct a grid, of user specified size, that has dimensions sufficiently larger than number of data. All data are then randomly assigned to points on the grid. Agents

then iteratively sort the objects on the grid into clusters. During each iteration, a specified number of agents will pick up and drop objects using a probability based on neighboring objects. In other words, dissimilar objects in the neighborhood of a given object increase the likelihood that it will be picked up by an agent (assuming that an agent is nearby) while objects similar to an object being held by an agent in the neighborhood of the agent will increase the likelihood that it will drop the object. The algorithm iterates until the solution converges or the maximum number of iterations is achieved. The advantage of this approach is that we do not need to specify the number of cluster a priori. However, the results of this class of algorithm typically need to be clustered by another algorithm to refine the results. Note that the algorithm developed in (Yang & Kamel, An aggregated clustering approach using multi-ant colonies algorithms, 2006) does not have this drawback because the agents label data during the process.

In addition to two types of approaches discussed above, researchers have developed a clustering algorithm based on the behavior of bees (Santos & Bazzan, March 30 2009-April 2 2009). This algorithm considers each datum as a single agent (bee). Each agent possesses following behaviors based on that of the recruitment behavior of bees. First, a bee can choose to perform the dance to recruit other bees into its group a la the manner in which bees recruit other bees to follow it to a pollen source. Alternatively, a bee can choose to visit other bees that are currently dancing. A bee who visits a dancing bee may decide to continue watching, leave the dancer, or abandon its group to join dancer's group. The decision to continue watching another bee is determined by local similarity (i.e. the inverse of some distance measure) between two bees while a decision to change group is based on the similarity of the bee and another group. The algorithm iterates until the solution converges or the number of maximum iterations is reached.

Clustering Algorithms for Power plant monitoring within an Information Theoretic Platform: The survey presented in the previous section provides the basis for selecting the clustering algorithm(s) best suited to our needs. Recognizing that one size does not fit all, different applications may warrant different approaches. Furthermore, defining proper selection criteria requires a close examination of the goals and objectives of power plant monitoring as well as the specific nature of the data under consideration. As noted in prior reports, a primary objective of the power plant monitoring system is to estimate operational modes of the plant so that estimation, detection, diagnosis and prognosis algorithms appropriate to the given operational state can be brought to bear. Further, the determination of the proper operational state must be done in near real time relative to the critical dynamics of the power plant's operational state and thus computational load and run time are crucial components of evaluating performance.

Consider a feature set extracted from available power plant observations in the first iteration of a clustering algorithm, denoted as $F_1 = \left\{f_1^{(1)}, f_2^{(1)}, \dots, f_P^{(1)}\right\}$ with associated sensor data $X_1 = \{x_1, x_2, \dots, x_T\}$. Note that the number of features P is not necessarily equal to the number of data T. Note also that, in real time operation, only data from the past up to current time is available. In practice, only recent historical data are typically needed since outdated data can bias or otherwise corrupt the result. The clustering algorithm will be used to obtain a clustering of the data $C_1 = \{c_1, c_2, \dots, c_P\}$ from F_1. This clustering, C_1, will be used for operational state analysis during the initial time step. In the subsequent iteration, a new set of data is obtained and, assuming that the design is appropriate, i.e. the feature set in the 2nd iteration, $F_2 = \left\{f_1^{(2)}, f_2^{(2)}, \dots, f_P^{(2)}\right\}$, may be assumed to be constrained by the condition $f_n^{(1)} = f_{n+d}^{(2)}\ \exists d$. In other words, two consecutive iterations do not produce different feature set. In general, it is expected that the clusters obtained in sequential iterations using any algorithm must be relatively close. For this reason, it is also expected that the clustering process in the second iteration will be significantly faster than that of the first. As the initial clustering proposed to the algorithm in the second iteration is that obtained from the first, which is close to that which will be generated during the second iteration, the algorithm can be assumed to converge quickly. This expectation holds in subsequent iterations until the end of operation/computation and thus the computational burden associated with these clustering algorithms can reasonably be assumed to be significantly less than their worst (or even average) performance case.

It can be seen from the literature review in previous section that, except in the case of centroid-based clustering or biologically inspired clustering, the expectation that subsequent iterations produce similar results is typically not met without the use of non-trivial implementation techniques – more sophisticated implementations of these algorithms do impose constraints in successive clustering processes and thus this is not an insurmountable obstacle. More importantly, however, is the fact that the expectation that computational loads will remain low due to quick convergence is not likely to be achieved by hierarchical clustering or density-based clustering techniques. The main reason for this conclusion is simply that these two approaches must be re-executed from scratch when new data is obtained new of stale data is discarded. For these reasons only centroid-based clustering and biologically inspired clustering can be considered for general use within this framework.

Biologically inspired clustering algorithms have the following advantages: 1) the shape of clusters in centroid-based approaches is limited by distance measure, 2) a larger number of clusters are necessary in centroid-based algorithms if the data from modes of interest are grouped in clusters having complex shapes, 3) the use of a large number of clusters imposes a requirement for a large amount of raw data and this greatly increases complexity, 4) it is well-known that biologically inspired algorithms are capable of finding good solutions relatively quickly and are, therefore, suitable for use where minor changes in the problem formulation and/or constraints can be expected (e.g., in response to changing operational conditions).

Thus, biologically inspired clustering algorithms are proposed for use within the information-theoretic framework for classification of the operational state of complex power plants. To this end, a representation of a notional power plant is considered and appropriate algorithms are applied to discover its information geometry. In addition to providing a basis for associating data sources with monitoring system functions, the geometry itself provides an indicator of changes, i.e., faults and failures, in the plant. For example, if a valve fails, new (existing) information channels corresponding to fluid flow in a pipe system may emerge (vanish) on the basis of the nature of the valve's failure (e.g., fully open, fully closed, stuck, …).

Biologically Inspired Routing Algorithms: Biologically inspired approaches for eliciting the structure of information flow are the focus of ongoing work due to the nature of power generation systems and operational environments. Next we examine two candidate routing algorithms for discovering the intrinsic communication topology of power generation systems.

This section is organized as follows: the first subsection describes the mathematical representation of a notional power generation system, and the candidate algorithms are reviewed in the second subsection.

Physical System Representation: Next we describe the representation of a power generation plant as the basis for algorithm development and testing. A schematic of a power generation plant, including relevant instrumentation infrastructure, is depicted in Figure 3.11 showing both physical and communication layers.

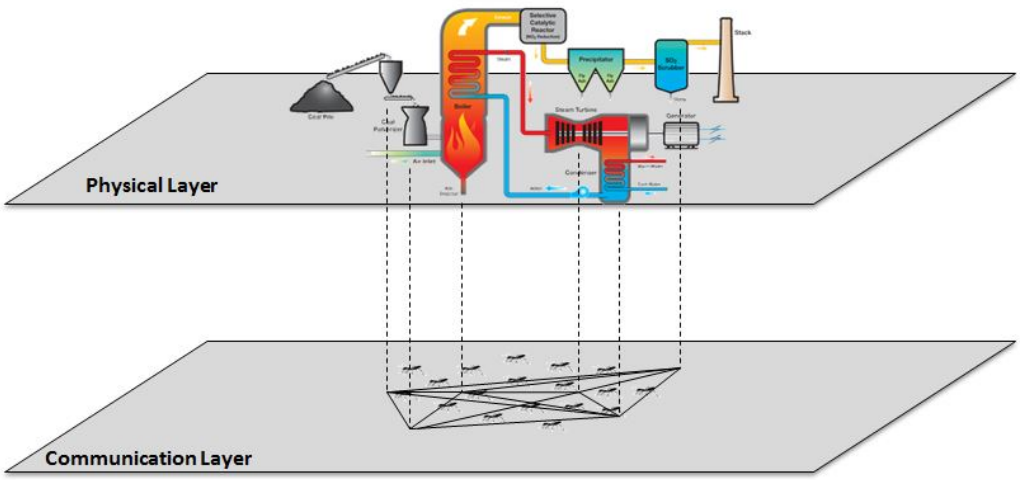


Figure 3.11: Representation of a Power Generation Plant

Equation Section (Next)The physical layer contains the power plant with all of its constituent elements, e.g., machines, pipes, sensors, devices, etc. The connections between these elements are implicitly mapped to the communication layer as a representation of the physical system as viewed through the available observation processes, i.e. instrumentation. This representation is distinct from both the physical plant itself and mathematical models that may be constructed of the plant and its constituent elements (including instrumentation). That is to say that the system elements, their physically mediated interconnections (viewed as communication via *physically-based communication channels*), sensors, sensor conditioning and processing elements, and communications infrastructure all play a role in determining the representation in the communication layer. Further, the environment and the status of the system play a role in defining this representation. Understanding this representation of the system and its relationship to the actual physical system is the necessary basis for any health and condition monitoring system. A crucial element of obtaining this understanding is an understanding of this implicit representation.

To this end, mobile software agents are deployed within the communication network to gather information from visited nodes. This information will then be used in a decentralized fashion by the agents to elucidate the *information geometry* of the network depicted in the communications layer in Figure 3.11**.** Note that this network does not necessarily possess a one-to-one correspondence with the communication system used for with instrumentation and control systems as the edges of the network graph include physically mediated communication between system elements. Nor is a one-to-one correspondence between the network's nodal organization and any physical or logical organization of the plant as a whole necessary as the various interconnections and system dynamics may couple elements (and their constituent components) in ways that do not reflect physical or logical organization, e.g., fluid elements may be coupled to mechanical elements or a vibration transmission path may couple two elements that are not in close physical proximity.

Routing Algorithms: Given the importance of networked communications, the literature in this domain is extensive. New algorithms are regularly introduced to comply with novel communication technologies and user demand. This section focuses on the definition and characterization of the core notions necessary to network discovery and biologically inspired algorithms for realization of the key functions that must be performed to elicit network structure. In particular, this section begins by defining the notions of "Telecommunication Network" and "Routing" (Di Caro, Ducatelle, & Gambardella, 2008). Next, the general behaviors required of routing algorithms are discussed and, finally, the first algorithms that applied Ant Colony Optimization (ACO) ideas to routing problems in networks (Farooq & Di Caro, 2008) are introduced.

Telecommunication Network: A telecommunication network is represented as a directed weighted graph $G = (V, \mathrm{E})$ where each node in V represents a processing and forwarding unit and each edge E is a transmission system with some characteristics such as bandwidth and capacity. Two nodes in a network are said to be neighbors if they communicate with each other directly. As we are more interested in general communication between system elements, e.g., physically, as opposed to electronically, mediated communication, or indirect communication due to feedback loops, we will retain the associated nomenclature and conventions, where appropriate, for general communication networks. In other words, the term telecommunication network, though technically an accurate description of physically mediated communication, is generally assumed to refer to electronic communication networks while the communication networks of interest here are not restricted to electronic communication networks. To this point, we will refer to this general case as simply a communication network and reserve telecommunication network (or other suitable terminology such as SCADA) to denote a restriction to electronic communications.

Routing: Routing is defined as directing data flow from source nodes to destination nodes while maximizing the network performance. Three cores of every routing function are:

- Acquisition, organization and distribution of data/information on user-generated traffic and network states,
- Applying above information to generate feasible routes while maximizing the performance,
- Forwarding traffic along the selected routes.

Routing tables are a key notion in routing algorithms. They are data structures that hold all the information that the algorithm uses to make the local forwarding decision. Routing tables are both a local database of information and also a local model of the global network status.

Every node in a network has a routing table for every possible destination in the network, and each table has an entry for every neighbor of that node. For example, consider the network in

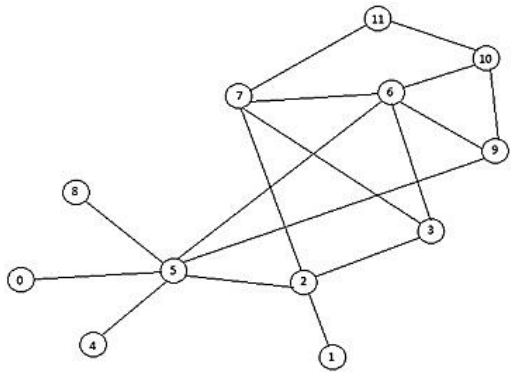


Figure 3. with 12 nodes. Node 3 has 3 neighboring nodes; therefore this node has 11 routing tables with 3 entries each. All the routing tables for node 3 are shown in one table in

Figure 3.13.

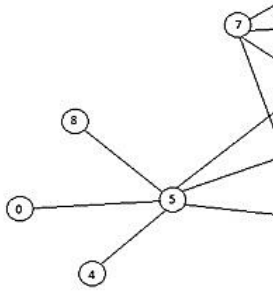


Figure 3.12: A Network with 12 nodes

Destination node	Next node	Pheromone value
0	7	0.437617
0	6	0.416205
0	2	0.146178
1	7	0.280460
1	6	0.015005
1	2	0.704535
2	7	0.291052
2	6	0.000703
2	2	0.708245
4	7	0.312407
4	6	0.602697
4	2	0.084896
5	7	0.390440
5	6	0.485925
5	2	0.123636
6	7	0.351311
6	6	0.627780
6	2	0.020908
7	7	0.950641
7	6	0.011946
7	2	0.037412
8	7	0.402235
8	6	0.206698
8	2	0.391067
9	7	0.578002
9	6	0.363279
9	2	0.058720
10	7	0.700919
10	6	0.271622
10	2	0.027459

Figure 3.13: Routing Table for Node 3

Routing Transmission Modes: There are two basic routing approaches: *Connection Oriented* and *Connectionless*. In connection-oriented algorithms, a path connection between two end-points must be established before sending a packet. The routing algorithm is required to find and use full end-to-end paths. On the other hand, in connectionless algorithms, connection establishment is not required. However, there is no guarantee that the packet will be delivered to the destination. In the work at hand, it is not necessary to explicitly enumerate all possible end-to-end paths between any two end points. This simply reflects the fact that the objective here is not to find a shortest route to push information but to discover the routes that are actually used. That is, the objective here is to "discover" the connections not to force a connection between two points. Therefore, only *connectionless algorithms* are considered here.

Delivered Service: Two delivered services are Best effort and Quality-of-Service (QoS). In QoS service the user can set constraints on the quality of the service such as bandwidth and end-to-end delay. However, in best effort service, there is no guarantee on the quality of delivered service. Again, as the objective here is to discover those routes in actual use, only best effort algorithms are considered as QoS algorithms introduce constraints on the network that may bias the discovery process. Note, it is important that all connections be discovered, regardless of bandwidth and delay, as all of them may provide information about the health and condition of the network. For example, an increase in noise transmitted between two nodes (and hence a decrease in bandwidth) may be indicative of a fault in the network. It is also likely that those communication channels with least bandwidth/greatest delay are the most sensitive to changes in the system.

Topology and Connectivity: Wired and Wireless mobile ad hoc networks are two types of topologies. In wired networks, the topology is fixed. All the elements, hosts and routers, are connected through one-to-one cables. In the case of addition/removal of elements of fault, a small, one-time, modification may occur. On the other hand, in wireless networks, all the nodes are mobile and can enter or leave the network at any time. They communicate with one another through wireless communications that can break or establish constantly due to the mobility nature of the network. A common, and an important, feature in these networks is their adaptability to changes in the network. Both of these topologies have their own challenges. Both types of networks are germane in the current effort.

There are two main reference algorithms for networks that use Ant Colony Optimization ideas to perform routing functions in networks: ABC for connection-oriented and AntNet for connectionless networks. A brief introduction on these two algorithms is presented next.

ABC: Ant-Based Control: ABC, proposed by Schoonderwoerd et al. at 1996, was the first notable algorithm to apply ACO ideas to routing and load-balancing problems in networks (Schoonderwoed, Holland, Bruten, & Rothkrantz, 1996). The authors considered a telephone network with established connections between sender and receiver. They proposed the ABC algorithm to distribute the calls in a telephone network, load balancing, and to minimize the rejected calls due to congestion as each node can only handle a limited number of simultaneous calls.

The cost used to define the underlying optimization problem is defined as the level of congestion over the network. Therefore, heavily congested nodes are to be avoided, and this can be accomplished by delaying ants at congested nodes. The ant's age is related to the length of the path taken by the ants, i.e. the time the ant has spent in the network. In this scenario, the younger ants have more influence on routing tables than older ones.

The authors have also incorporated noise into their algorithm. The noise is added to the random walk of ants via the specification of exploration probabilities. In this case, less used and useless routes will also be explored with a likelihood specified by the exploration probabilities. This provides the algorithm with a mechanism for rapid discovery newly occurring routes that may provide better network performance. Also, the information gathered by the ants can be used in the case of blocked routes to avoid route freezing.

The general framework of ABC algorithm is as follows. Ants are deployed regularly in the network with a random destination specified. Every node deploys an ant with a random destination at every time step of the simulation. The ants perform random walks based on their local routing tables. They then update their routing tables by increasing the probability for the routes they have just taken. To reduce the cost and level of congestion, if ants are delayed on congested routes, their age is increased by their presence in the network, thus decreasing their influence on defining network routing tables. Last but not least, ants are penalized/rewarded during their walks as a function of local system utilization to promote network balancing.

AntNet: An Adaptive Agent-Based Routing Algorithm: AntNet was proposed by Di Caro and Dorigo at 1998 (Di Caro & Dorigo, 1998). It provides traffic-adaptive routing for connectionless IP data networks. It is a distributed, mobile agent-based algorithm based on Monte Carlo methods. The authors have introduced indirect communication among agents, i.e. stigmergy, through local information readings from nodes and writing three data structures at the nodes; pheromone tables, a set of statistical model parameters, and a local data routing table. The algorithm provides a distributed method for spreading data traffic over the best available paths based on the updated routing information from agents.

The framework of the AntNet algorithm is shown in Figure. In this algorithm Routing tables are first initialized with uniform (equal) values for all the neighbors, i.e. a destructive approach. Agents, which are called forward ants, are deployed to randomly selected destination nodes and search for paths with minimum delay. At each node visited along their paths, they gather information about the end-to-end delay for that path. While forward ants are moving, the information on traversal time and congestion status of the path is stored locally. When they reach their destination, they become backward ants and move along the same path in opposite direction. They update the local routing table using the information obtained during their forward traversal of their path. Backward ants die when they return to their source node.

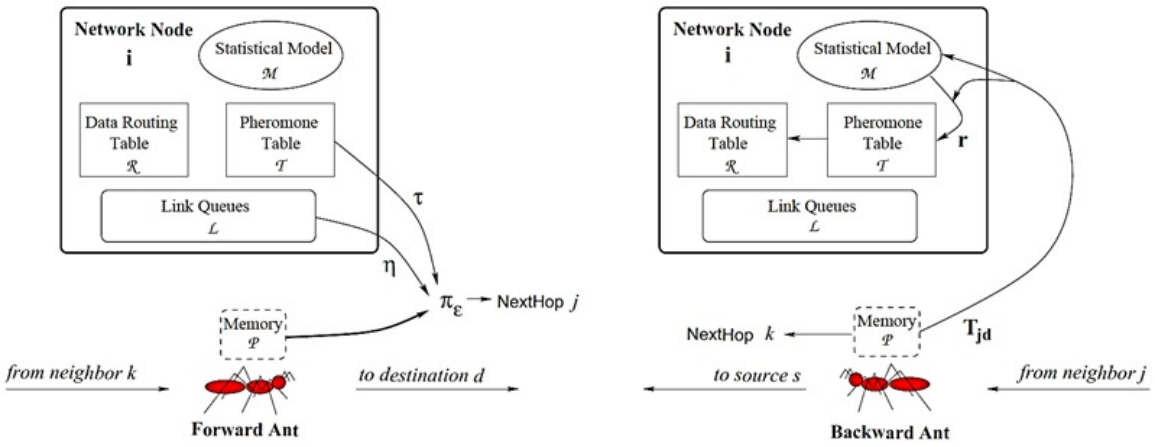


Figure 3.14: AntNet Algorithm

In the AntNet algorithm, if a cycle is detected, the cycle's nodes are removed from agent's memory stack. Also, if a cycle lasts longer than the lifetime of the ant before entering the cycle, the ant will be destroyed.

ABC and AntNet algorithms are two main algorithms for routing algorithm in networks. Several algorithms were proposed after ABC and AntNet that are modifications of these algorithms, each improving one specific weakness of ABC and AntNet. For example, Oida and Kataoka in (Oida & Kataoka, 1999) discussed that the earlier version of AntNet was prone to stagnation. This is because the heuristic term based on the instantaneous status of the data link queues was not included in the earlier version. They solved this problem by adding an evaporation mechanism to their proposed algorithms, DCY-AntNet and NFB-Ants.

The specific areas addressed by modifications that have been introduced to date are listed as below:

- **Updating Routine:** Update the pheromone values only for source node vs. updating for all nodes in the route. Also, all returning ants can update the table vs. only elitist ants can.
- **Next Node Selection:** heuristic, statistical, etc.
- **Initial Routing Table:** Uniform distribution, random, etc.
- **Ant colony Type:** Multiple colonies with distinct pheromones or only a single type colony.

Based on the characteristics of the network and our objectives, the above criteria can be investigated to ascertain their suitability to the network topology problem at hand. This investigation will be the focus of effort during the next quarter. Specific plans include completing a design test bed for evaluating route discovery algorithms, implementing the necessary computational infrastructure and agent emulation software, and implementing and testing route discovery algorithms.

Note: Choosing a simulation tool is very important and it should be able to perform a precise validation of the developed algorithms. Authors in (Saleem, Di Caro, & Farooq, 2011) have done an extensive review on Swarm Intelligence based routing algorithms for wireless sensor networks. They discussed that although all the algorithms had been evaluated in simulation, the simulation environment was not satisfactorily described. They provided a pie chart representation of different simulators used in the reviewed papers. This chart is shown in Figure 3.15. NS-2 simulator was one of the top choices (%29). In the following, we will give a brief introduction to this simulator.

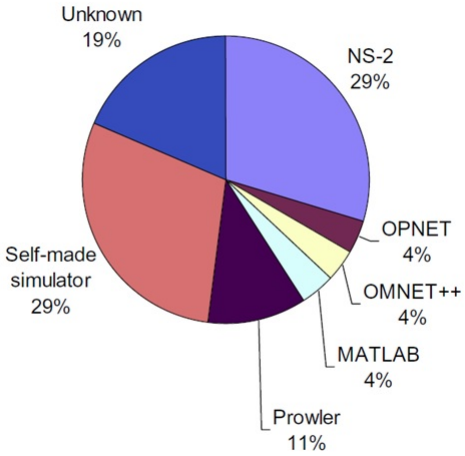


Figure 3.15: Simulator Usage

The Network Simulator (NS), is a series of discrete event networks (Fall & Varadhan, 2011). There are three version of this simulator: NS-1, NS-2 and NS-3. All of them are discrete-event network simulator, primarily used in research. They provide substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. NS-1 and NS-2 are no longer maintained. NS-3 is actively developed (but not compatible for work done on ns-2). It is built using C++ and Python with scripting capability. NS-3 is often criticized for its lack of support for protocols (like WSN, MANET etc.) that were supported in NS-2, as well as for the lack of backward compatibility with NS-2. This simulator does not provide graphics. Raw or processed data can be graphed using tools like Gnuplot, matplotlib or XGRAPH.

Simulated data from Alstom Power for their 1000 MWe ultra-supercritical coiler and steam plant has been provided to the CWRU team. The steam plant is simulated using a dynamic process simulator (Yang, Lou, Neuschaefer, Boisson, & Bories, 2013). The steam generator produces steam flow to a turbine generator with boiler outlet conditions of the main steam flow of 600^c at 58 bar g. The plant net heat rate is 9045 kJ/Kwh. Figure shows a schematic of the power plant. Extensions to this model are described in Section 4.0 of this report.

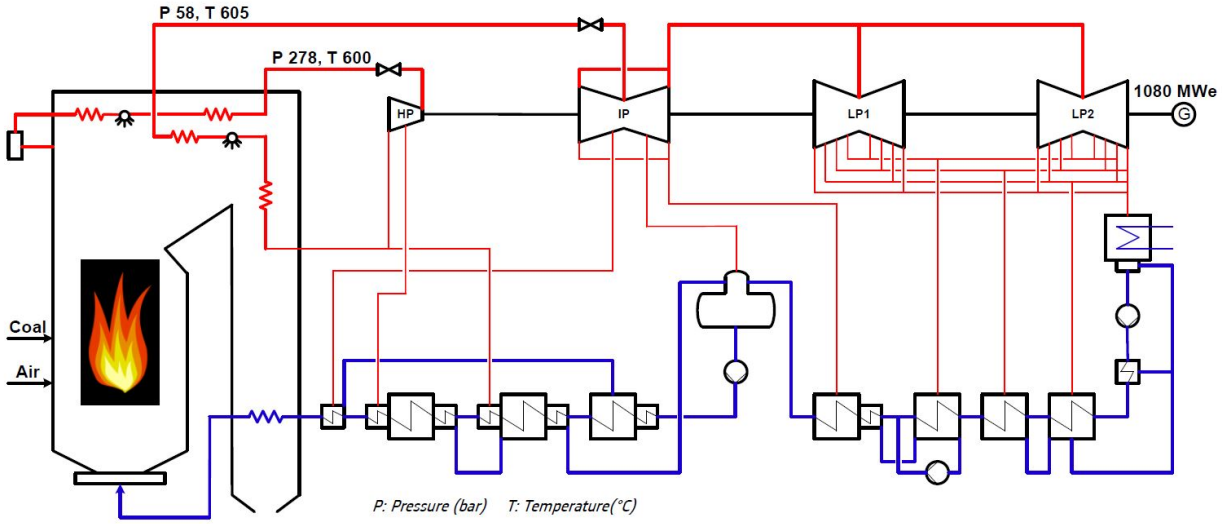


Figure 3.16: Schematic of the Steam Power Plant

The dynamic simulator of the steam plant with its inputs and outputs are shown in Figure 3.17.

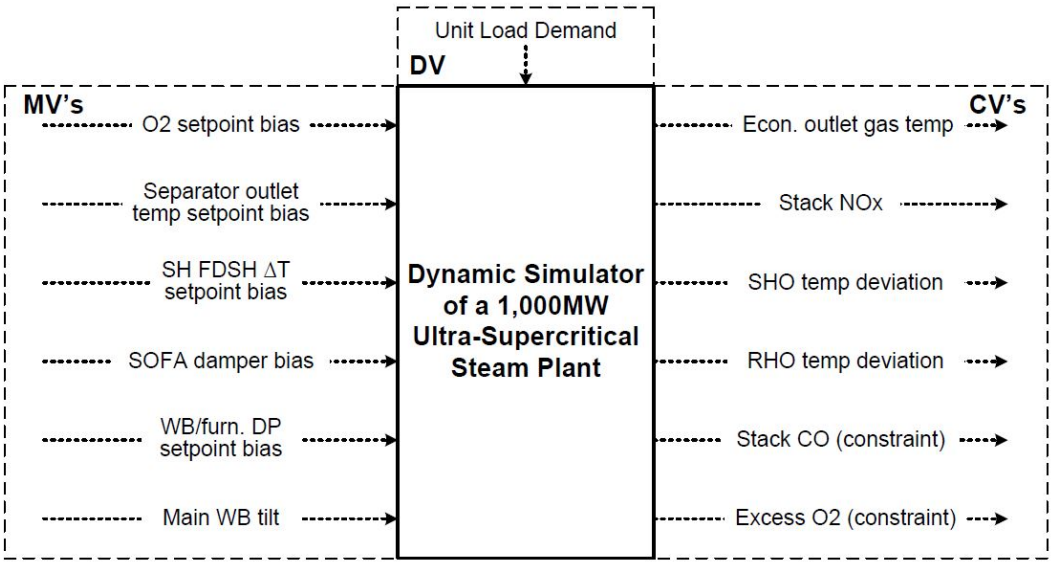


Figure 3.17: Dynamic Simulator of Steam Plant with Inputs/Outputs

A step test sequence was generated to identify the model between each input/output pair. In each case, a common initial condition is used and the simulator permitted to run until it reaches the corresponding steady state operating point. The given step test is then performed and the outputs recorded. For the dynamic simulator depicted in Figure , a total of seven (7) step tests were performed and, for each, six (6) outputs were recorded. Figure shows a representative result, namely, the normalized step response for O2Bias.

Table 3.3: Steam Plant Model Input

Inputs	
O2Bias	O_2 Setpoint Bias
DPBias	Furnace Differential Pressure Setpoint Bias
FDT	Final desuperheater Temperature
SOFA	Separated overfire air damper bias
Tilt	Main Wind Box Tilt
ULD	Unit Load Demand
WWO	Waterwall Outlet

Table 3.4: Steam Plant Model Output

Outputs	
SHODT	Superheater Outlet Temperature Deviation

RHODT	Reheater Outlet Temperature Deviation
FEGT	Final Exhaust Gas Temperature
NOX	Stack NOx
CO	Stack CO (constraint)
ExO2	Excess O_2 (constraint)

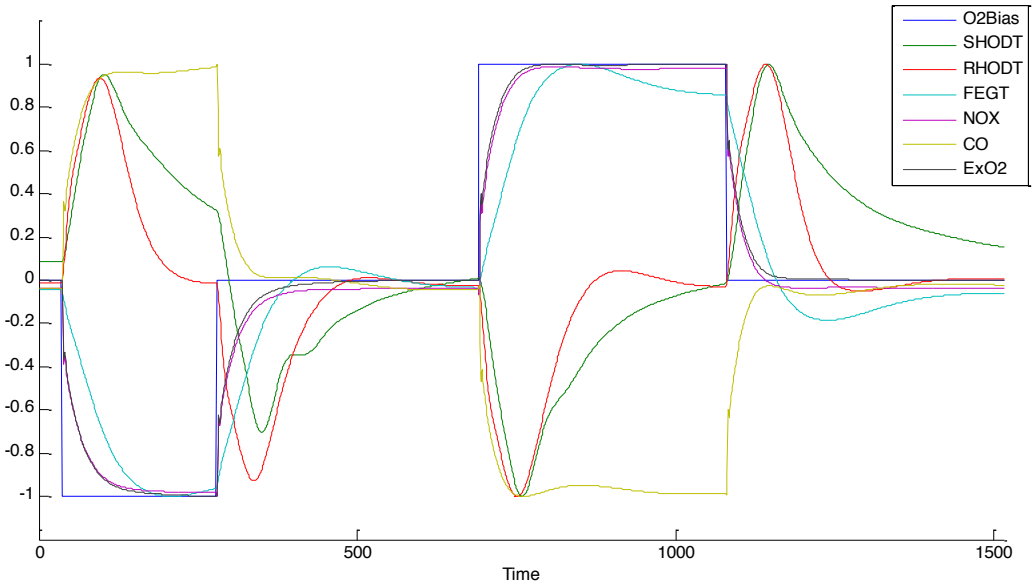


Figure 3.18: Normalized O2Bias Step Test Results

To develop a self-organizing infrastructure for health and condition monitoring of coal-fired steam plants, the monitoring infrastructure should be able to detect and identify the current operating state of the system along with any changes in that operating state. For example, by monitoring the changes in the output of a plant, the current operating state (O2Bias, FDT, NOX, etc) should be identified along with the changes in the operating state (step-up, step-down, etc).

We propose an algorithm based on ACO techniques to discover the information connectivity between the 6 outputs of the plant for each operating state. Based on those topologies, we would be able to connect it with its corresponding operating state. Also, changes in those topologies at each operating state may be connected to the changes in the operating state and by monitoring those changes we would be able to detect the changes.

The basic approach to discover the information connectivity between system elements, i.e. dynamic simulator outputs, is shown in Figure 3.19 below. Each node represents an output of the steam plant dynamic simulator.

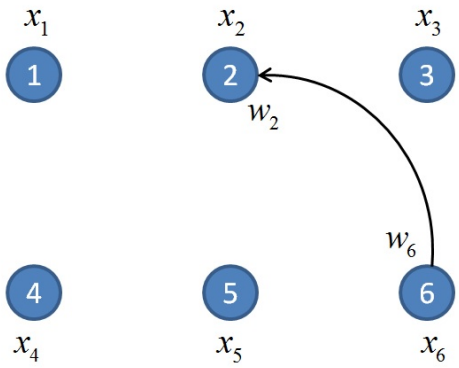


Figure 3.19: System Representation

In this figure, w_i is a time series that represents partial observations of the system at node i. Swarm agents, mimicking the behavior of ants foraging for food, traverse a network constructed between the nodes. When an agent goes from a node to another, it carries some data, described as time series w_i, from its home node to the next node. The food that the agent is foraging for is then defined as the "similarity" between these two time series. That is, the agent forages for data streams emanating from other nodes that contain the same informational content as the data it carries.

In the initial implementation of this algorithm, the correlation coefficient, i.e. the degree of linear correlation between them, is used to capture this similarity between two time series. This is shown schematically in Figure 3.20.

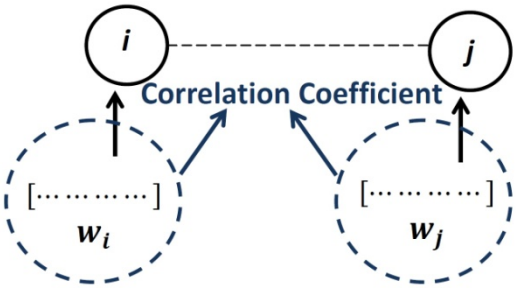


Figure 3.20: Calculating Correlation Coefficient between two nodes in the system

The correlation coefficient is a measure of linear correlation between two variables that has a value between +1 and -1, with 1 representing positive dependence, 0 meaning no correlation, and -1 as negative linear dependence. The absolute value of the correlation coefficient between the data epochs from two nodes is the strength of the connection that is attached to the path between the nodes and the stronger this connection, the larger the number of agents that will take this path.

In the ACO algorithm, ants start from the source node and take the shortest route to the food and bring it back to the source node. In an adaptation of the ACO algorithm in this project, the agents take the route with stronger connectivity, that is the route with higher absolute correlation coefficient $|\rho|$. As they move, they lay pheromone on the route that is related to the correlation coefficient between the two nodes. Therefore, more ants will take this path while discovering the network. After all the ants return to the source node, we will have a primary understanding of the information topology of the network.

The proposed algorithm is as follows:

1. Agents are placed randomly on the nodes. We set the number of agents equal to the number of nodes. Therefore, at the beginning, there is an agent at each node.
2. For agent k at node i, the next node j is chosen based on the following rule:

$$j = \begin{cases} \arg \max_{u \in J_i^k} \{[\tau_{iu}(t)]\} & if \ \mathrm{q} \le \mathrm{q}_0 \\ J & if \ \mathrm{q} > \mathrm{q}_0 \end{cases}$$

   - q: a random variable uniformly distributed over $[0,1]$
   - q_0: a tunable parameter over $[0,1]$
   - J_i^k: set of all the nodes in the system except for the current node i
   - $J \in J_i^k$: node that is randomly selected according to this probability:

$$p_{iJ}^k(t) = \frac{\tau_{iJ}(t)}{\sum_{l \in J_i^k} \tau_{iJ}(t)}$$

3. The agent goes to node j and updates the pheromone trail of the pair (i, j) according to the following rule:

$$\tau_{ij}(t) \leftarrow (1-\rho) \cdot \tau_{ij}(t) + correlation\_coefficient(w_i, w_j)$$

   - $correlation\_coefficient(w_i, w_j)$: correlation coefficient between the time series of the two nodes connecting the path
   - ρ: pheromone decay parameter

4. Go to step 2 and repeat until the end of simulation.

Note that in step 3, w_i is the data agent records while is in state i. When the agent goes to the node j, it compares its information brought from node i to the information available in node j by calculating the correlation coefficient between (w_i, w_j). Agents stay at the node for a limited time and then travel to the next node following the rule in step 2. They continue travelling through the network until the end of the simulation.

Simulation Results: In this section, we apply the proposed algorithm to the data provided by GE (Alstom). The data is categorized into 7 sections, each corresponding to a step test. We apply the algorithm to each section and illustrate the information connections between the nodes for that section. We will then compare the results and discuss the similarities and differences between them.

The first simulation run focuses on the O2Bias step test. The path traveled by each of the agents is shown in Figure where each of the starting nodes is denoted by a circle. The arrows indicate the agents' travel direction. A bidirectional arrow means that the path is used in both directions. The black arrows are for the paths used less than 10% during the simulation while red arrows are for paths used more that 90%. The black arrows occur most commonly at the beginning of the simulation, where the step test input is zero.

One can see from Figure that the path between 'CO' and 'RHODT' is common for all the agents.

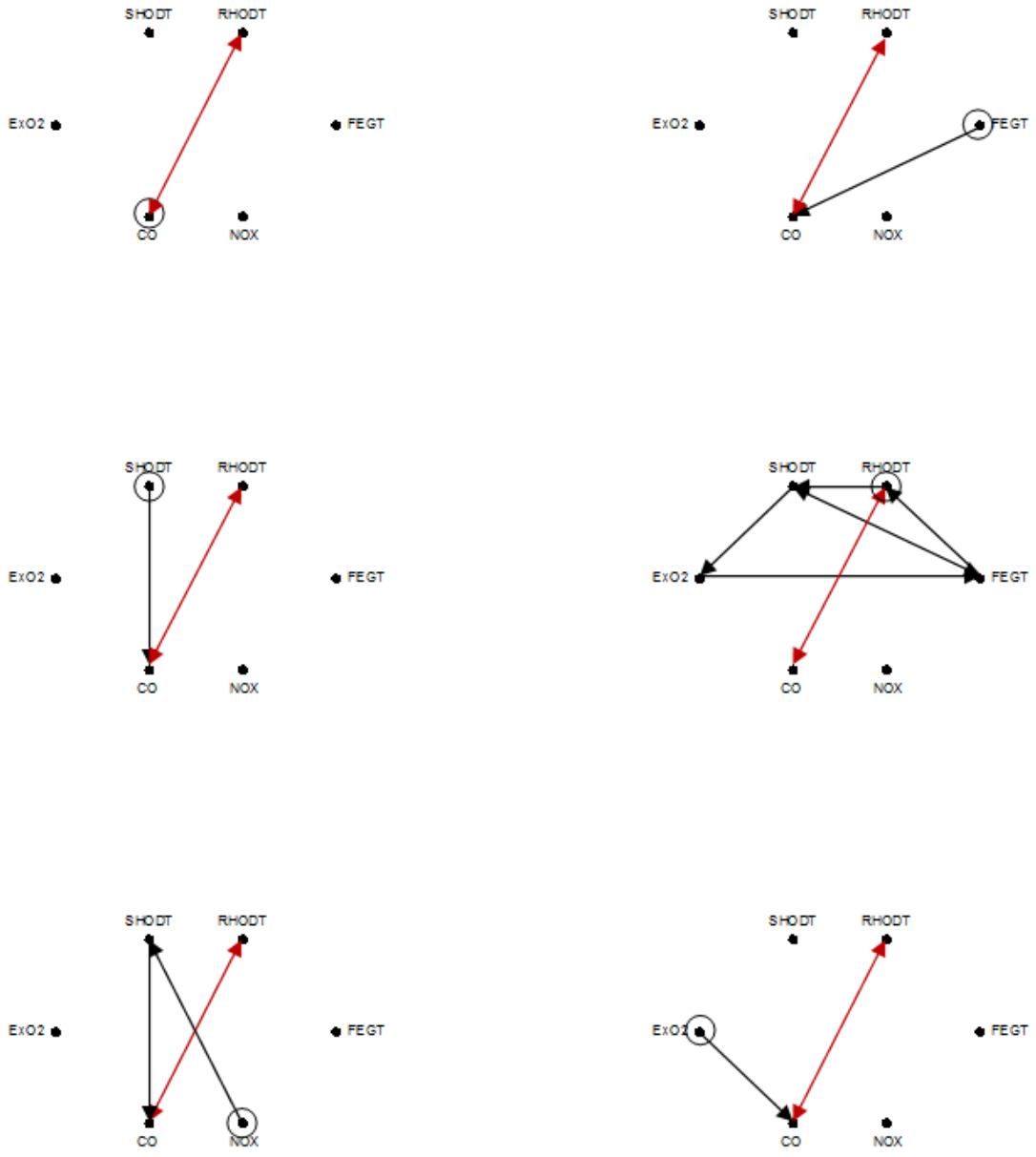


Figure 3.21: Information Topology for O2Bias Step Test

The simulation was run several times and the ensemble behavior considered. The results were found to be inconsistent across the runs as shown in Figure and Figure for two different simulation runs over the same data. As can be seen, the most traveled path in Figure is (NOX,FEGT) while in Figure, the most traveled path is (RHODT,NOX).

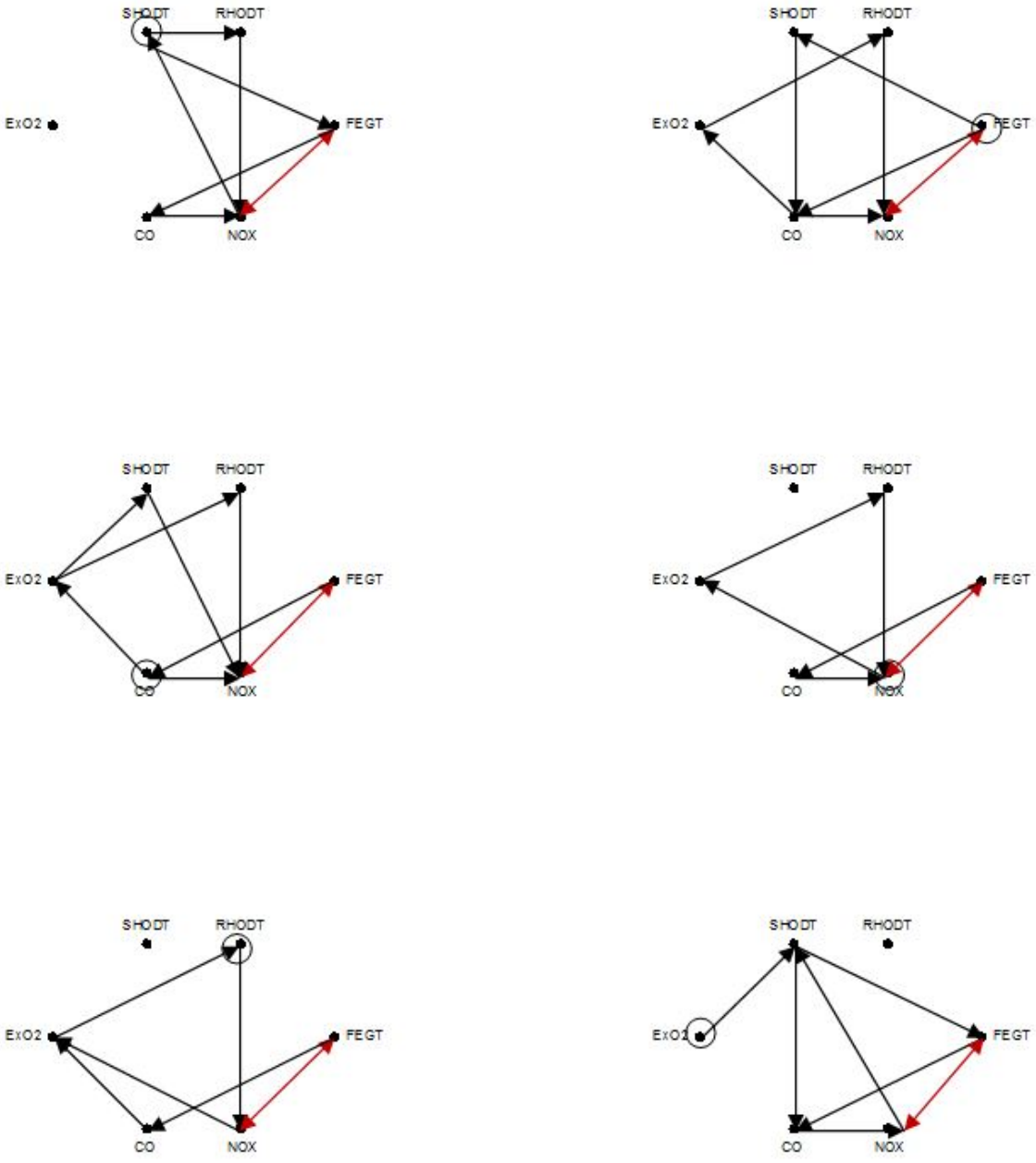


Figure 3.22: Information Topology for O2Bias Step Test - Second Run

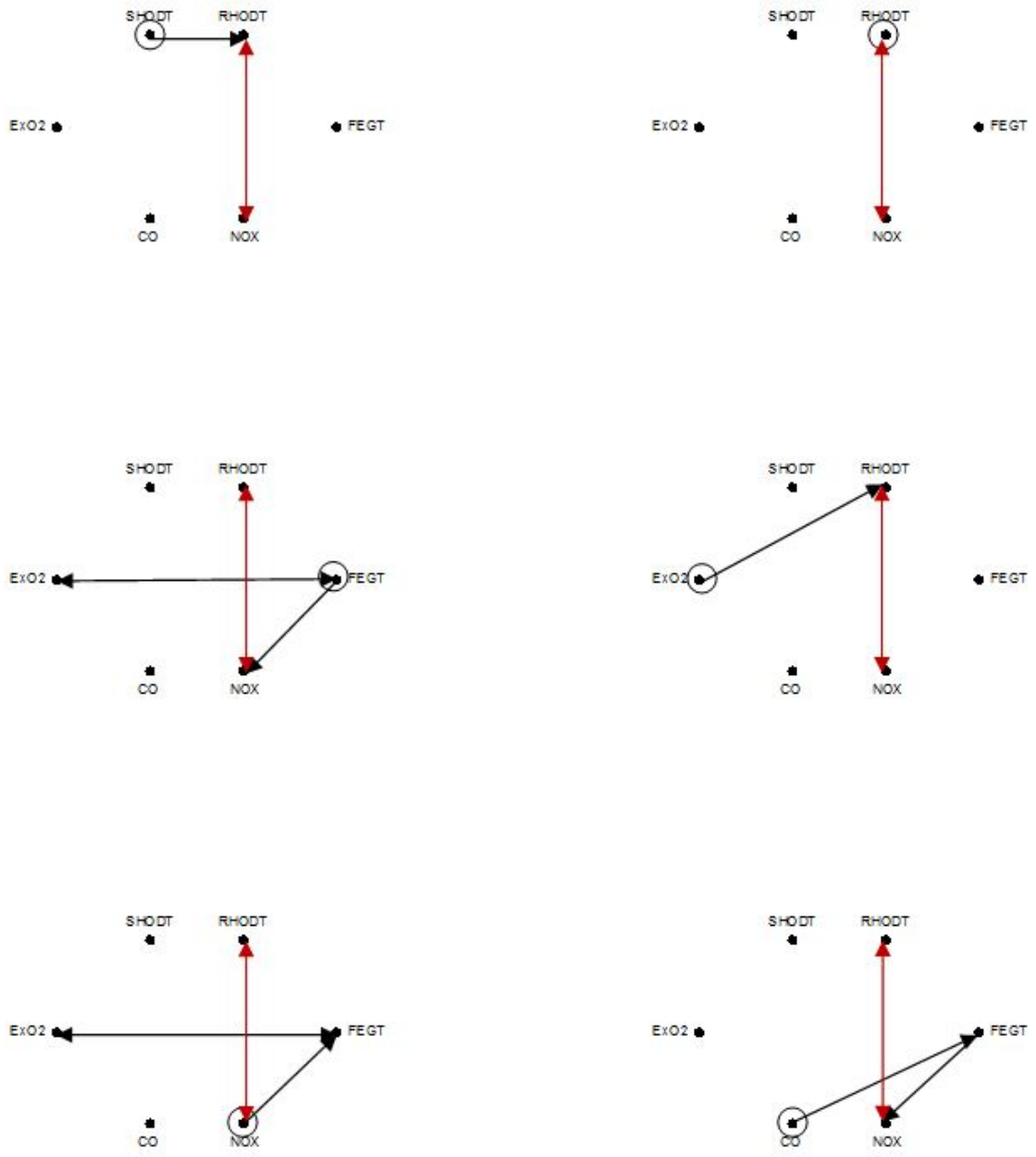


Figure 3.23: Information Topology for O2Bias Step Test - Third Run

In these runs, the agents are initially distributed randomly over the nodes and the pheromone levels on all trails are set to zero. Therefore, for the first couple of iterations, the agents move to other nodes randomly. It takes several iterations to build an initial pheromone reference table with some nonzero values. Because of the fact that this table is constructed solely based on the random travels of the agents, the routes containing nonzero pheromone values are highly dependent on the randomly selected nodes. For example, if at the first couple of iterations, nodes (1,3,4,6) are selected, there is a higher chance that at the end of simulation these nodes are among the most traveled paths. The reason is that the pheromone values on those paths will be nonzero from early iterations. This encourages the agents to utilize these paths instead of non-traveled path with zero pheromone values. Please note that these paths may have smaller correlation coefficient compared to other paths.

In order to address this issue, some changes to the algorithm such as exploration rule, initial pheromone levels and pheromone trail updates are required.

The simulation is repeated for FDT step test data for three different runs. The results are shown in Figure through Figure . The results are similar to O2Bias Step Test.

The most traveled paths for these three runs are, in order:

- {(SHODT,FEGT,CO,SHODT),(RHODT,ExO2,NOX,RHODT)}
- {(SODT,RHODT),(NOX,ExO2,CO)}
- (ExO2,CO)

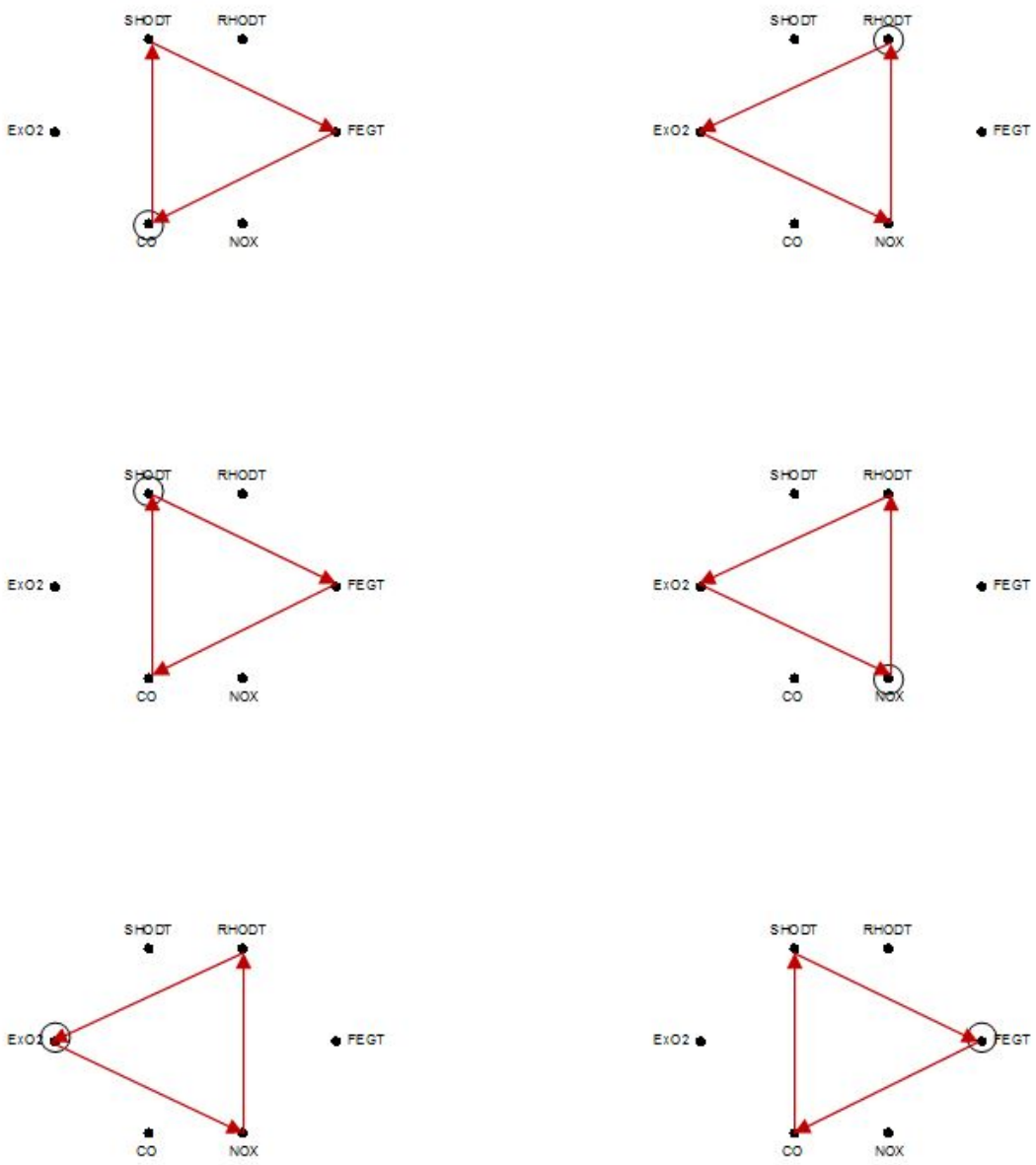


Figure 3.24: Information Topology for FDT Step Test - First Run

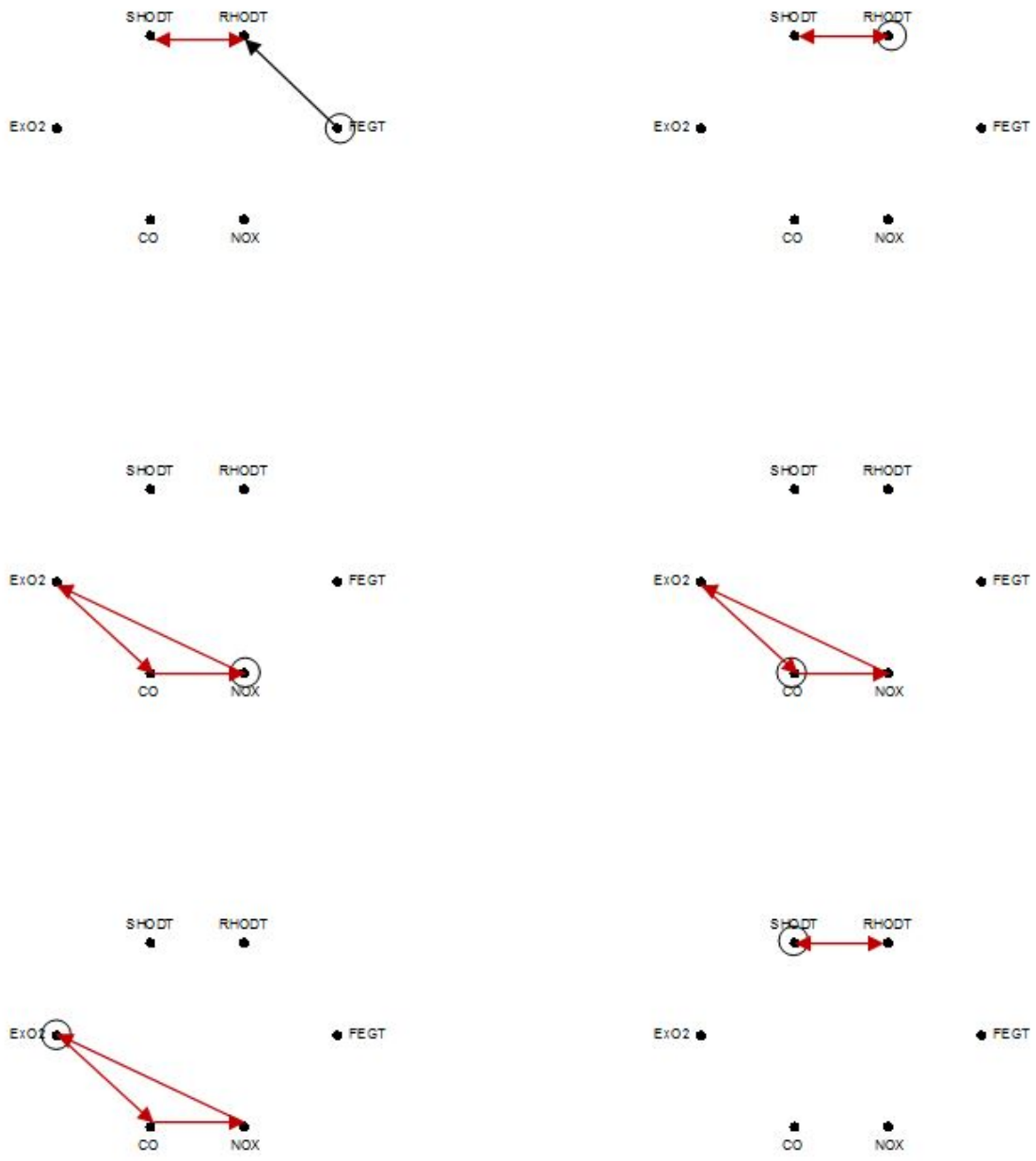


Figure 3.25: Information Topology for FDT Step Test - Second Run

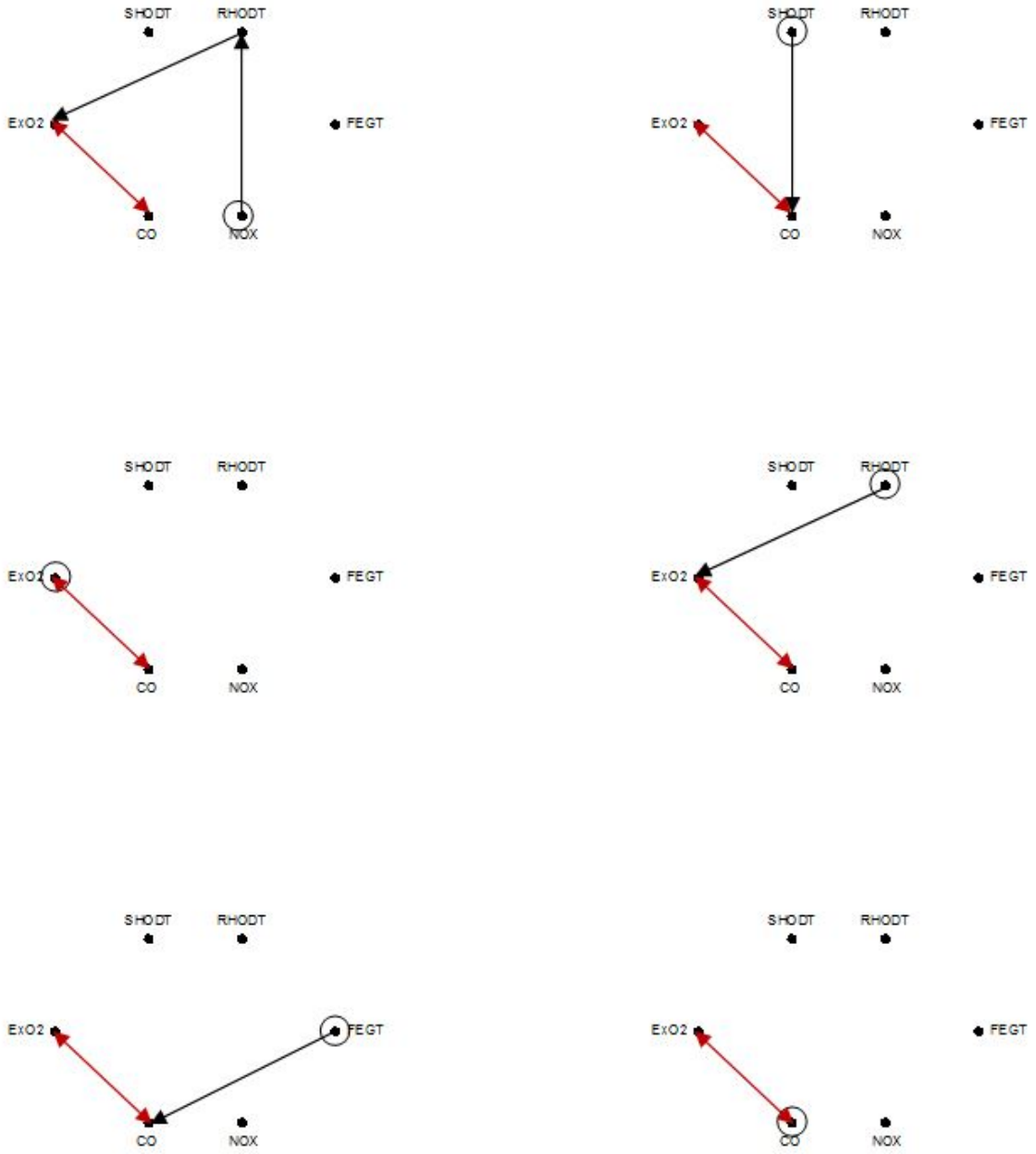


Figure 3.26: Information Topology for FDT Step Test - Third Run

Some of the most traveled paths for the rest of the step tests are given in Figure through Figure .

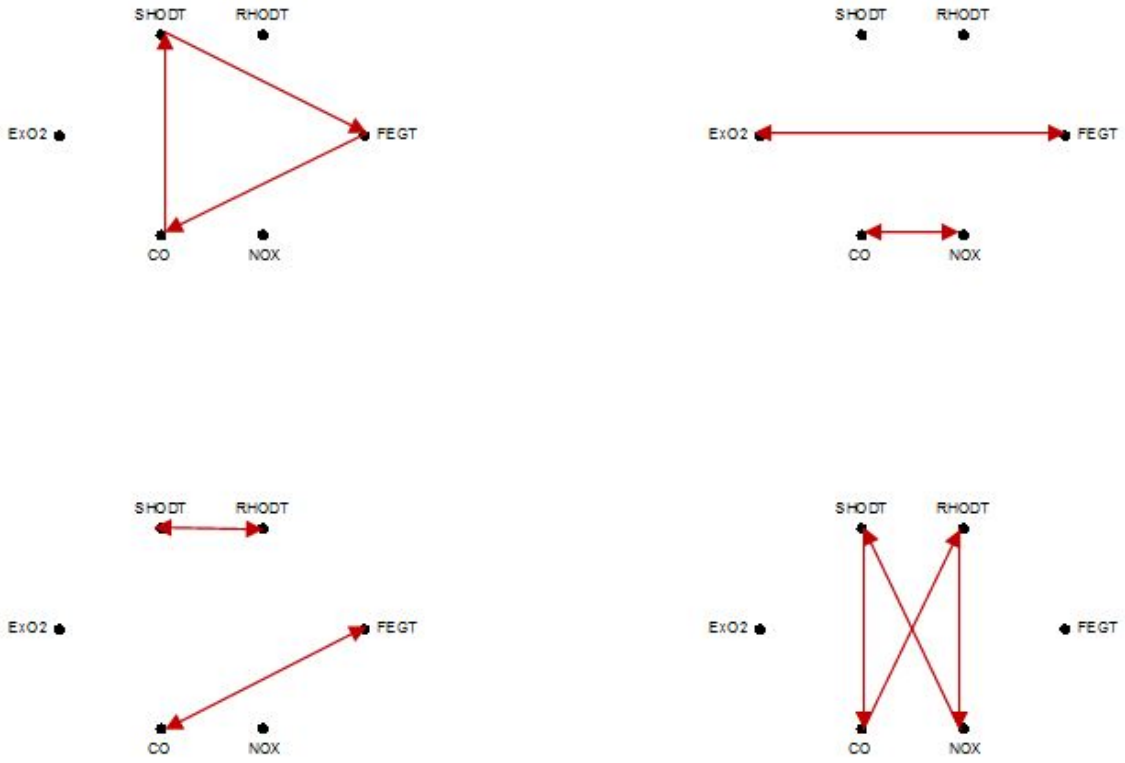


Figure 3.27: Information Topology for DPBias Step Test

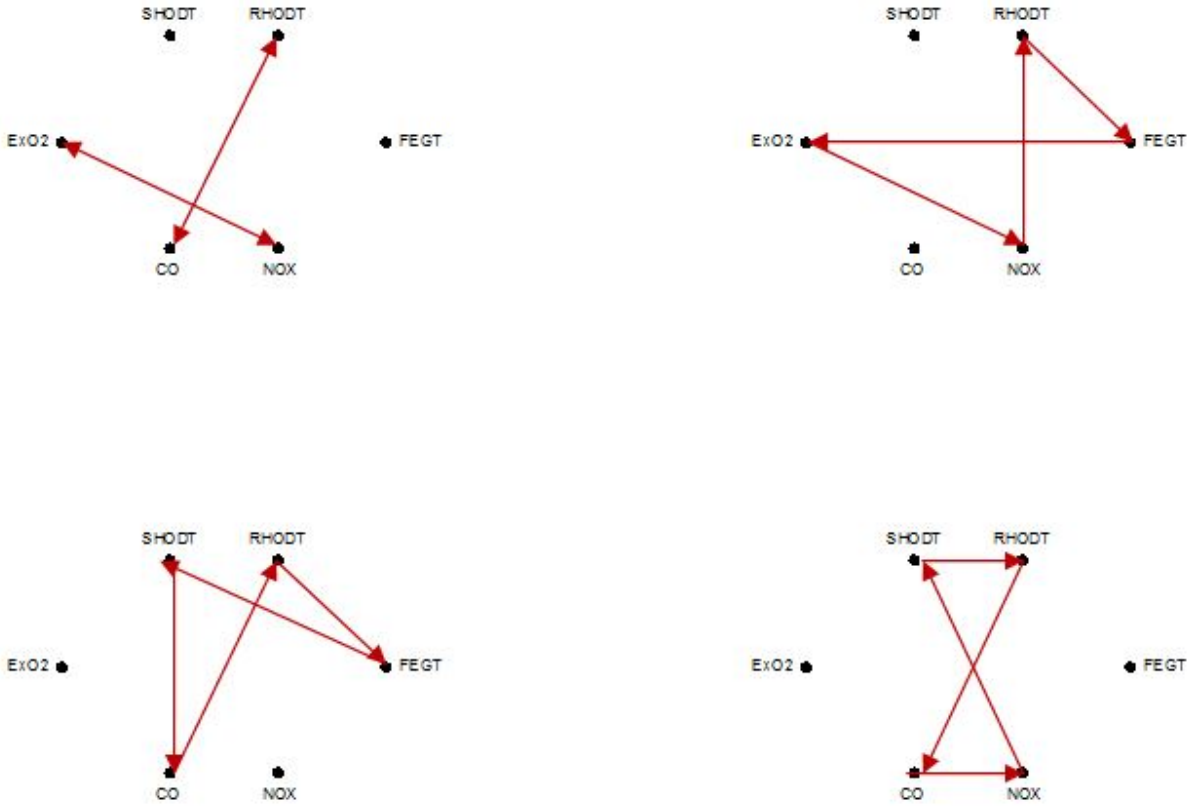


Figure 3.28: Information Topology for SOFA Step Test

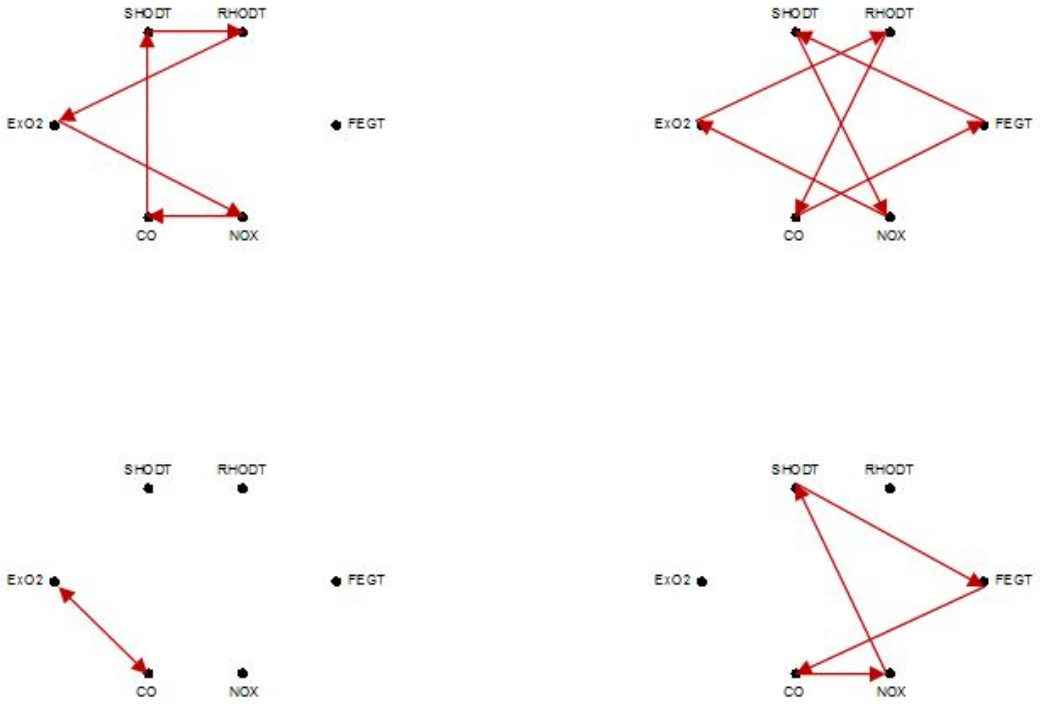


Figure 3.29: Information Topology for Tilt Step Test

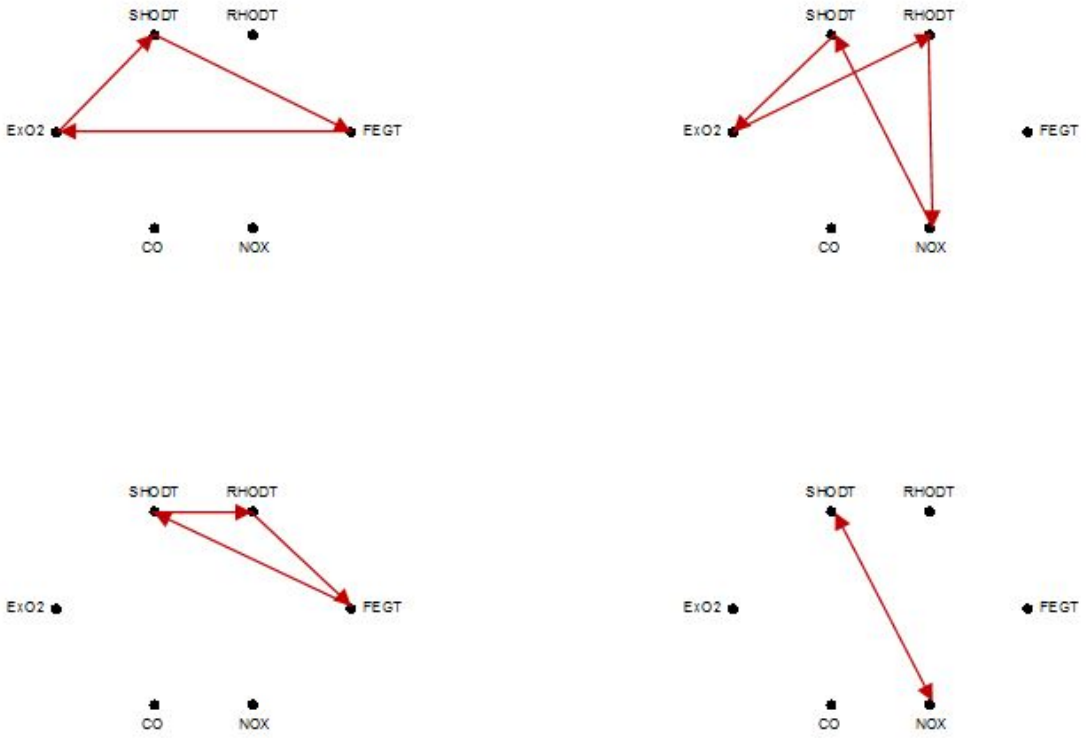


Figure 3.30: Information Topology for ULD Step Test

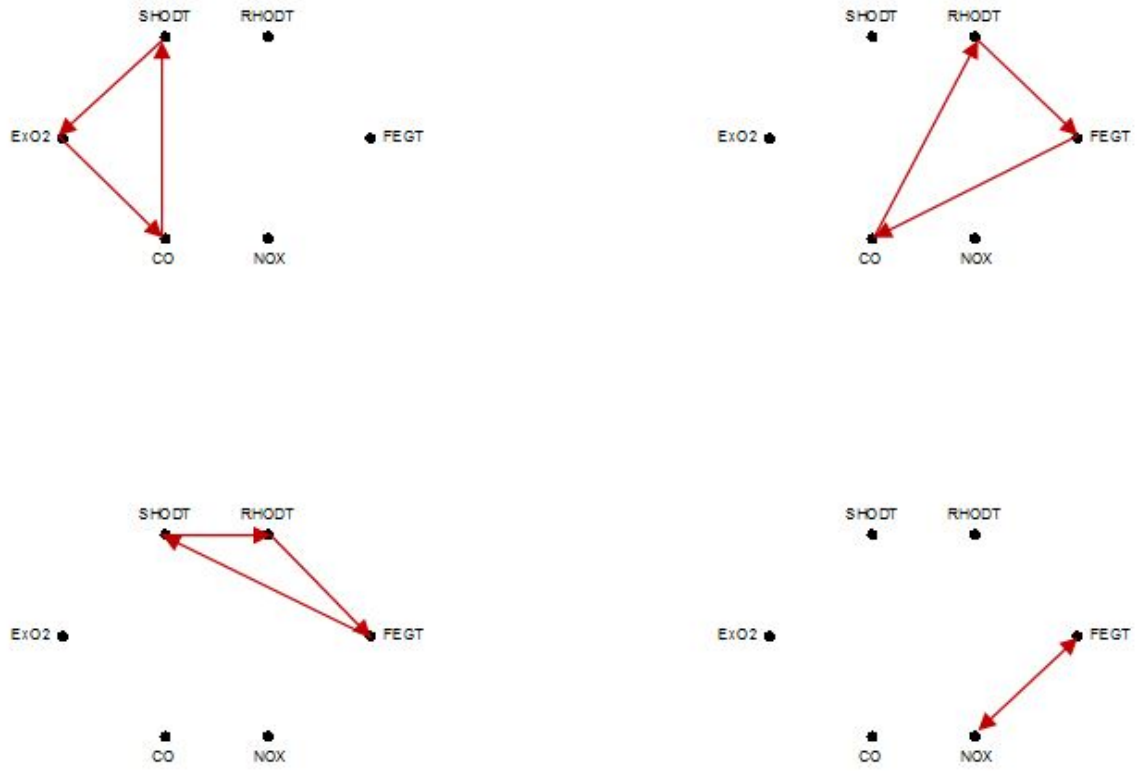


Figure 3.31: Information Topology for WWO Step Test

As a next step, we consider identifying an appropriate analytical model for this power plant simulation. To identify the model, a series of test inputs were used to probe the steam power plant simulation and identify the model between each input/output pair. In each case, a common initial condition is used and the simulator is permitted to run until it reaches the corresponding steady state operating point. The given step test is then performed and the outputs recorded. A total of seven (7) step tests were performed and, for each, six (6) outputs were recorded.

The data obtained with these test inputs consists of steps with the length of around 300 seconds each. While this is sufficient to capture the majority of the relevant aspects of the plant dynamics, it is not sufficient for the time scales over which the agent-based software under development must operate. By identifying a dynamic model from these data, a model operating over the requisite time scales can be constructed. This identified model also possesses the virtue that it will generate an appropriate response for any weighted combination of the inputs. The following sections detail the identification of this model. First, the identification of mathematical model for the dynamic system is presented. Then, its use for agent-based topology discovery is demonstrated.

Data Preparation – Modeling: The model data, consists of 7 inputs and 6 outputs. There are two principle methods available for working with multi-input multi-output data:

1. Construct a model for individual collections of inputs and output and then merge all of the resultant models into a single model as shown in Figure 3.22.
2. Merge all the data sets into a single MIMO data set and construct a single MIMO model as illustrated in Figure .

The first method works better if the noise characteristics of data sets are different. The second method, on the other hand, is more efficient if the noise conditions are roughly the same for all data sets and tends to be more robust for characterizing interactions between the disparate sets. In general, modeling multi-output systems is more challenging than modeling single-output systems. They require additional parameters to obtain a good fit and involve more complex models. They also commonly lead to worse simulation results because it is harder to reproduce the behavior of several outputs simultaneously. Both approaches were investigated and it is shown that the first approach provides better performance.

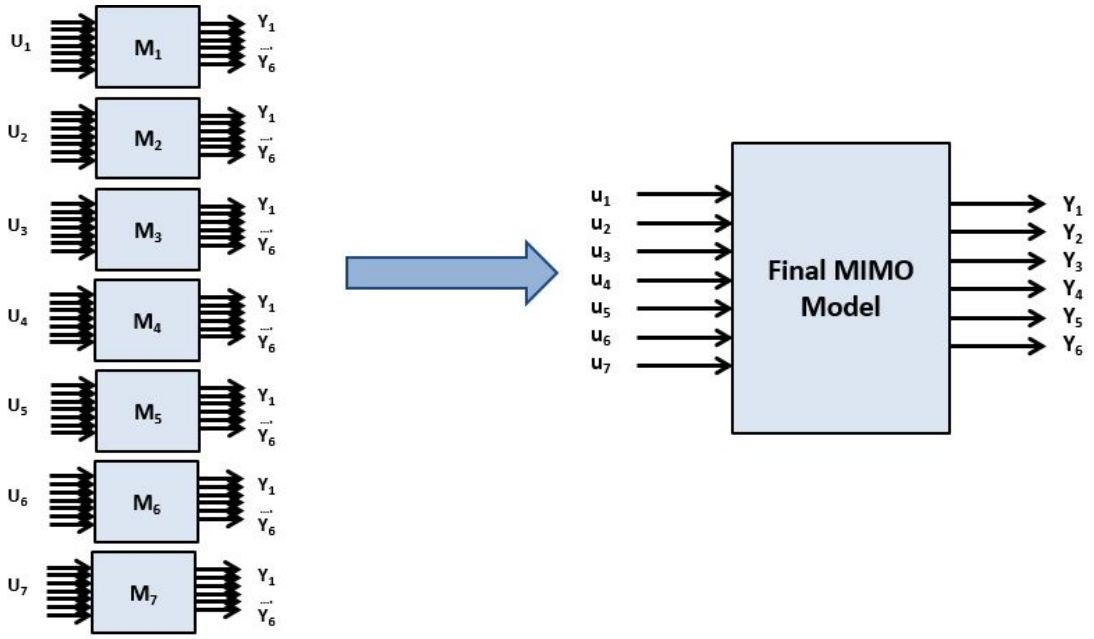


Figure 3.12: Merging Models for MIMO Modeling

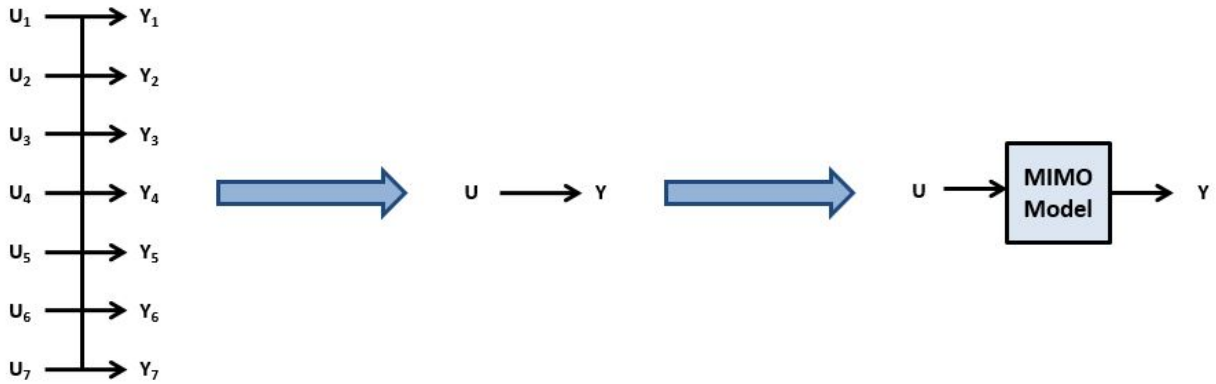


Figure 3.33: Merging Data for MIMO Modeling

Multiple-Input Multiple-Output (MIMO) Modeling: The process of identifying a general MIMO model is initiated by identifying a linear multiple-output state-space model. To this end, the System Identification Toolbox in MATLAB was used to construct a linear state-space model. Data sets from all step tests were used for model construction with 75% used for model identification and the remainder used for evaluation of the resultant models.

To improve identification, the data were normalized to be in $[0,1]$. Note that normalization does not change the correlation coefficient between data and thus normalized data can be used in lieu of raw data:

$$\left.\begin{aligned}
&\rho_{X,Y} = \frac{\mathrm{cov(X,Y)}}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{E[(X-\mu_X)^2]}\sqrt{E[(Y-\mu_Y)^2]}} \\
&X^{new} = \frac{X-\min(X)}{\max(X)-\min(X)},\ \sigma_X{}^{new} = \frac{1}{\max(X)-\min(\mathrm{X})}\sigma_X \\
&Y^{new} = \frac{Y-\min(Y)}{\max(Y)-\min(Y)},\ \sigma_Y{}^{new} = \frac{1}{\max(Y)-\min(Y)}\sigma_Y \\
&\mathrm{cov(X,Y)}^{new} = \left(\frac{1}{\max(X)-\min(\mathrm{X})}\right)\left(\frac{1}{\max(Y)-\min(Y)}\right)\mathrm{cov(X,Y)}
\end{aligned}\right\} \rightarrow \rho_{X,Y}{}^{new} = \rho_{X,Y}$$

This is important as the agent-based algorithms used to characterize the system's communication topology use the correlation coefficient between two points as a measure of connectivity strength. A model constructed on the basis of normalized data will not adversely affect the validity of the test results obtained on it.

MIMO Modeling Results: The discrete time state-space model resulting from the modeling of merged data from all step tests data is as follows:

$$\begin{cases} \mathrm{x}[t+T_s] = A\,x[t] + B\,\mathrm{u[t]} + K\,e[t] \\ \mathrm{y}[t] = C\,x[t] + D\,u[t] + e[\mathrm{t}]. \end{cases}$$

The estimated state space model is of order 6 with sampling time of 5 seconds is as follows:

$$
\begin{cases}
\begin{bmatrix} x_1[t+T_s] \\ x_2[t+T_s] \\ x_3[t+T_s] \\ x_4[t+T_s] \\ x_5[t+T_s] \\ x_6[t+T_s] \end{bmatrix} = A \begin{bmatrix} x_1[t] \\ x_2[t] \\ x_3[t] \\ x_4[t] \\ x_5[t] \\ x_6[t] \end{bmatrix} + B \begin{bmatrix} ULD[t] \\ O2Bias[t] \\ DPBias[t] \\ WWO[t] \\ SOFA[t] \\ Tilt[t] \\ FDT[t] \end{bmatrix} + K \begin{bmatrix} e_{SHODT}[t] \\ e_{RHODT}[t] \\ e_{FEGT}[t] \\ e_{NOX}[t] \\ e_{CO}[t] \\ e_{ExO2}[t] \end{bmatrix} \\
\begin{bmatrix} SHODT[t] \\ RHODT[t] \\ FEGT[t] \\ NOX[t] \\ CO[t] \\ ExO2[t] \end{bmatrix} = C \begin{bmatrix} x_1[t] \\ x_2[t] \\ x_3[t] \\ x_4[t] \\ x_5[t] \\ x_6[t] \end{bmatrix} + D \begin{bmatrix} ULD[t] \\ O2Bias[t] \\ DPBias[t] \\ WWO[t] \\ SOFA[t] \\ Tilt[t] \\ FDT[t] \end{bmatrix} + \begin{bmatrix} e_{SHODT}[t] \\ e_{RHODT}[t] \\ e_{FEGT}[t] \\ e_{NOX}[t] \\ e_{CO}[t] \\ e_{ExO2}[t] \end{bmatrix}
\end{cases}
$$

Using the System Identification Toolbox produces the following MIMO model given:

$$
A = \begin{bmatrix} 0.703 & 0.119 & 0.056 & -0.010 & 0.013 & 0.009 \\ -0.031 & 0.853 & 0.004 & 0.046 & -0.051 & 0.052 \\ 0.017 & 0.036 & 0.906 & -0.022 & 0.044 & -0.049 \\ -0.029 & -0.126 & 0.0001 & 0.997 & -0.054 & 0.047 \\ -0.090 & 0.037 & 0.026 & 0.007 & 0.985 & -0.012 \\ -0.020 & -0.019 & 0.011 & 0.014 & -0.015 & 0.993 \end{bmatrix}
$$

$$
B = \begin{bmatrix} 0.025 & 0.059 & 0.093 & -0.065 & 0.198 & 0.004 & 0.041 \\ 0.046 & 0.014 & 0.006 & 0.019 & -0.002 & -0.001 & -0.002 \\ -0.039 & -0.084 & -0.218 & 0.080 & -0.096 & -0.007 & -0.032 \\ 0.028 & 0.017 & -0.028 & 0.024 & -0.014 & -0.003 & -0.006 \\ 0.008 & 0.026 & 0.056 & -0.029 & 0.068 & -0.002 & 0.017 \\ 0.011 & 0.012 & 0.029 & -0.008 & 0.016 & -0.001 & 0.005 \end{bmatrix}
$$

$$
K = \begin{bmatrix} 0.139 & 0.867 & 0.165 & -0.108 & 0.806 & 0.670 \\ -1.572 & 0.769 & 1.103 & 0.265 & -0.329 & -0.574 \\ -15.43 & 8.114 & 4.795 & -0.106 & -3.164 & -5.614 \\ -3.278 & 2.249 & 0.994 & 0.299 & -0.755 & -1.148 \\ -0.511 & -1.066 & -0.027 & -0.347 & 0.085 & 0.260 \\ -0.336 & -0.553 & 0.045 & -0.026 & 0.068 & 0.033 \end{bmatrix}
$$

$$\mathbf{C} = \begin{bmatrix} -0.112 & 0.642 & 0.083 & -1.079 & -0.227 & -1.459 \\ 0.199 & 0.001 & 0.058 & -0.186 & -0.127 & -2.204 \\ 0.898 & 1.317 & -0.039 & 0.187 & 0.308 & -0.234 \\ -0.430 & 0.812 & -0.288 & 0.862 & -4.294 & 6.427 \\ 0.985 & -0.132 & 0.348 & -1.912 & -0.548 & 1.443 \\ 0.193 & 0.491 & -0.322 & 1.293 & -0.375 & -0.483 \end{bmatrix}$$

$$\mathbf{D} = \mathbf{0}$$

The validity of this model can be examined by considering its response to the test inputs. To this end, consider the ULD step test shown in Figure . Using the ULD step test signal as input to the above state-space model and recording the corresponding model output, we can compare the identified model's performance relative to the simulation output, shown in Figure . As can be seen, this model does not reproduce the output signals accurately.

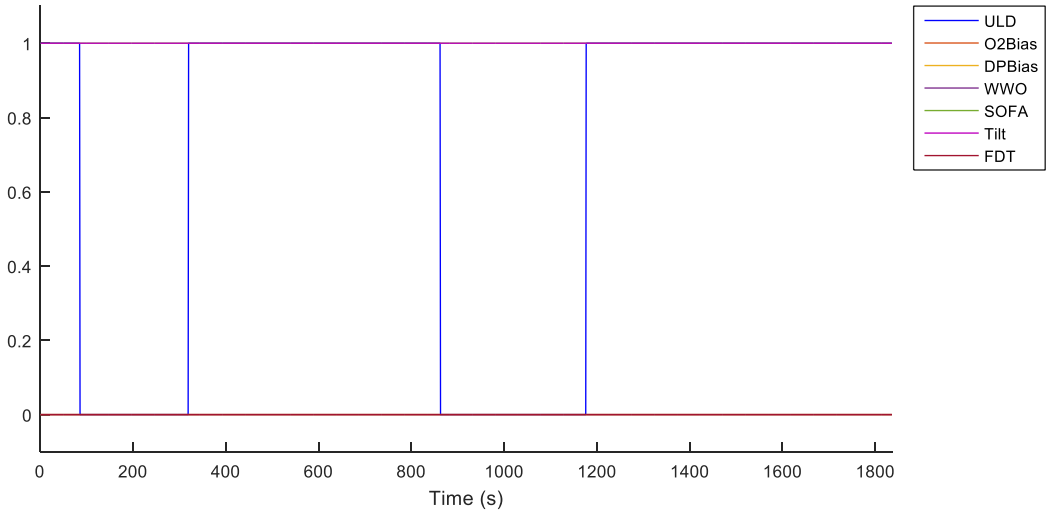


Figure 3.34: Normalized ULD Step Test Input

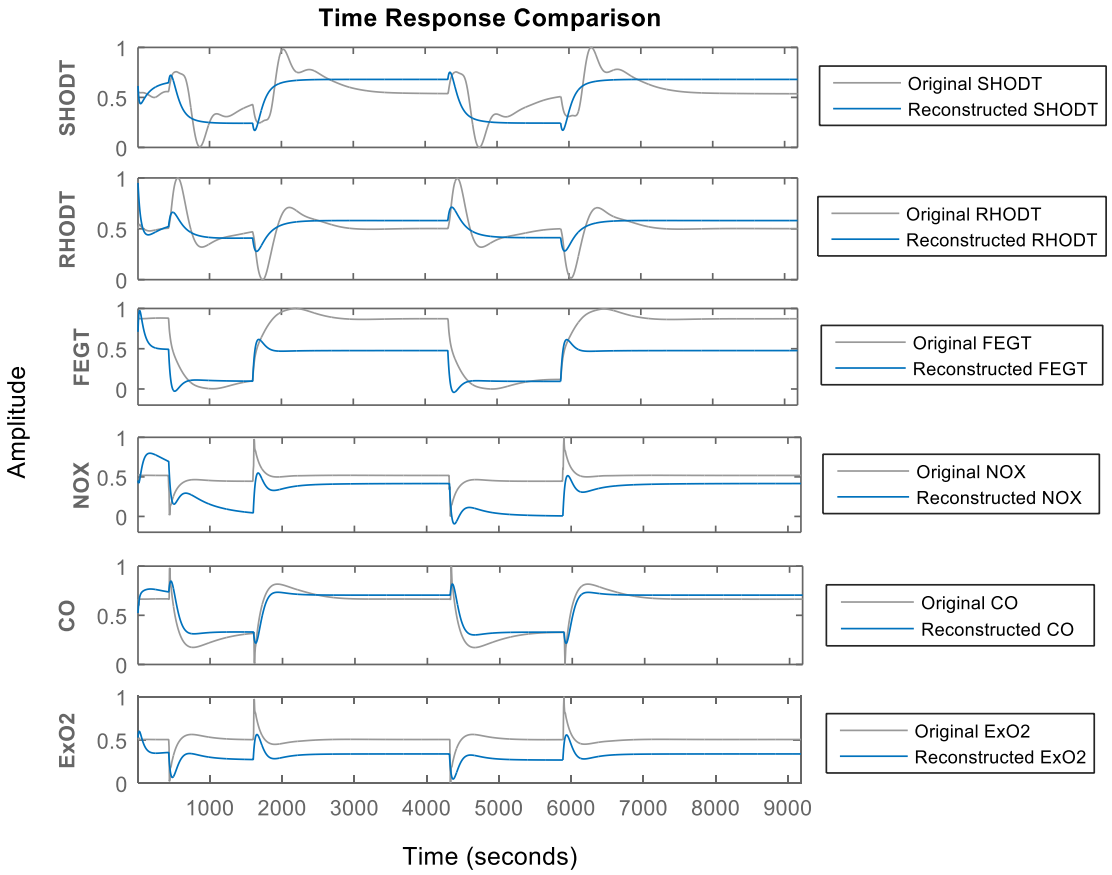


Figure 3.35: Original and Reconstructed Signals Comparison for ULD Input

The accuracy in terms of normalized root mean square (NRMSE) for all the input-output pairs are listed in Table 3.**6**:

$$\text{NRMSE} = 100\left(1 - \frac{\|y - \hat{y}\|}{\|y - \text{mean}(y)\|}\right)$$

where $\|.\|$ denotes the 2-norm, y is the original output and $\hat{y}$ is the estimated (i.e. identified model) output. This value varies between $-Inf$ (bad fit) to 100 (perfect fit).

Table 3.6: Estimation Accuracy for State-Space MIMO Model

		Output					
		SHODT	**RHODT**	**FEGT**	**NOX**	**CO**	**ExO2**
Input	**ULD**	18.36%	20.13%	5.342%	- 232.3%	66.33%	- 233.9%
	O2Bias	23.3%	14.58%	20.58%	40.5%	52.84%	49.54%
	DPBias	- 41.14%	- 61.08%	23.59%	70.09%	-271.4%	-169.9%
	WWO	- 9.169%	-7.896%	20.76%	28.93%	15.61%	- 39.12%
	SOFA	-154.7%	-53.57%	- 4.873%	12.68%	32.63%	- 637.9%

	Tilt	5.497%	- 21.27%	1.751%	15.92%	- 25.61%	- 95.72%
	FDT	- 14.83%	0.5984%	-11.72%	- 33.62%	- 29.44%	-16.59%

As can be seen from
Table 3.6, the identified MIMO model does not approximate the output signals very well. This can result from several different factors, most notably, assumed model order (i.e. under or over fit), the presence of significant nonlinearities that render the linear model used here invalid, or computational/data sufficiency issues that make the identification process intractable. To investigate the effect of model order, the model identification is repeated using state-space models of different orders. However, changes in model order do not significantly improve the goodness of fit.

As noted above, identifying MIMO models can be fraught, with the difficulty growing with the number of input and output variables. The presence of nonlinearities compounds this issue and thus, seeing that a MIMO linear model does not provide acceptable performance, a collection of single-input single-output models is investigated for modeling the disparate interactions between the data. Specifically, a set consisting of 42 (6x7) SISO models is developed corresponding to all the input/output pairs. If these models are combined via linear superposition, the resultant will be a linear MIMO model. This approach, however, provides a mechanism for incorporating nonlinear effects while maintaining a linear structure and, thus, this approach is better suited to the data at hand.

Single-Input Single-Output (SISO) Modeling: The construction of a 6x7 array of SISO models, as illustrated in Figures 3.37 and 3.38, is presented. Each block in these figures is a distinct SISO model. In the linear case, the final model output is simply the sum of all 42 outputs.

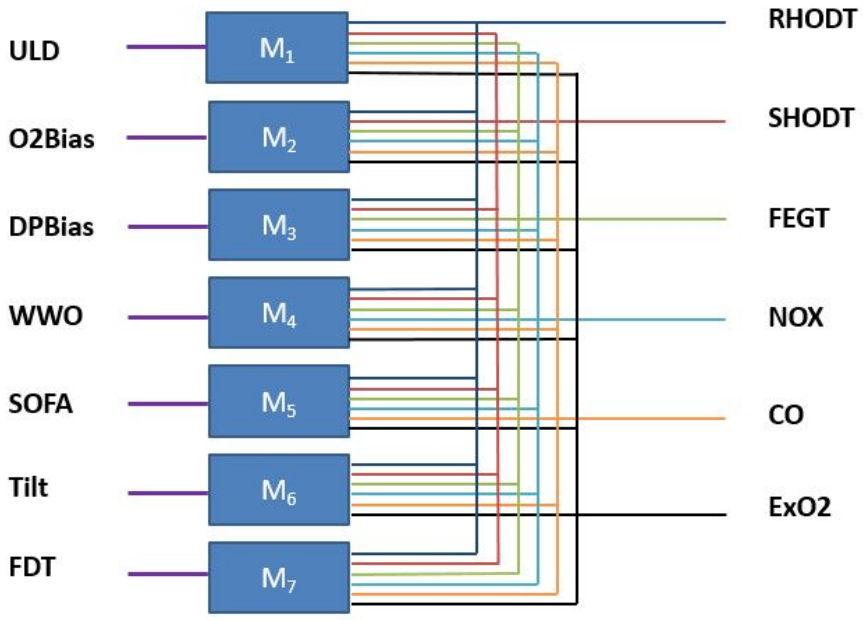


Figure 3.36: SISO Models for each input/output pair

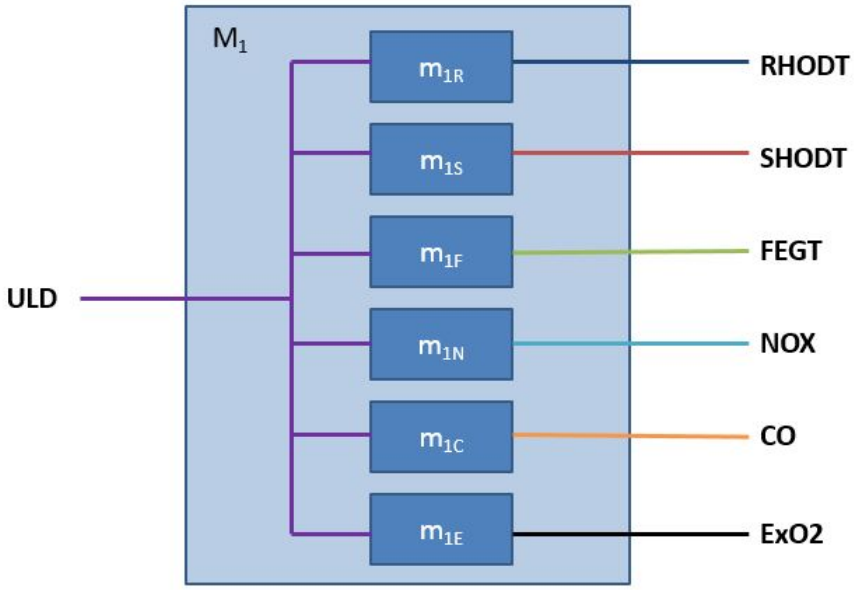


Figure 3.37: 6 SISO Models for ULD Input

SISO Modeling Results: As in the previous case, the use of linear models does not produce acceptable measures of goodness of fit, irrespective of model order. If it is posited that the inputs and outputs are modified by static memoryless nonlinearities, a nonlinear model can be constructed that adjusts the interactions between the inputs and output while preserving the linear structure of the dynamic evolution of the system via the technique of **Hammerstein-Wiener Modeling.** Hammerstein-Wiener systems have a linear I/O model between two nonlinear memoryless blocks that adjust the manner in which the model inputs and outputs, respectively, are combined. Hammerstein-Wiener models are applied in several areas, such as modeling electro-mechanical system, audio and speech processing and predictive control of chemical processes. Due to their convenient block representation and transparent relationship to linear systems, these models are very popular. They are also easier to implement than heavy-duty nonlinear models such as Neural Networks. The block diagram of a Hammerstein-Wiener model is shown in Figure .

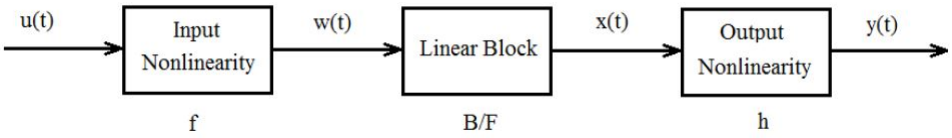


Figure 3.38: Hammerstein-Wiener Model

In the figure, we define:

- $w(t) = f(u(t))$ is a nonlinear function transforming input data $u(t)$. It is called the Input Nonlinearity because it acts on the input port of the linear block, and
- $x(t) = B/F(w(t))$ is a linear transfer function, and
- $y(t) = h(x(t))$ is a nonlinear function that maps the output of the linear block to the system output. It is called the Output Nonlinearity because it acts on the output port of the linear block.

The input and output nonlinearities are static memoryless functions, where the value of the output at a given time t depends only on the input value at time t. They can be Sigmoid Network, Wavelet Network, Saturation, Dead Zone, Piecewise Linear Function, one-dimensional Polynomial or any custom function.

Both input and output nonlinearities are not necessarily included in the model structure. When a model contains only the input nonlinearity f, it is called a **Hammerstein Model**. Similarly, when the model contains only the output nonlinearity h, it is called a **Wiener Model**. The properties of the general Hammerstein-Wiener model are as follows:

- **Input Nonlinearity**: A one-dimensional polynomial of degree m.

$$f\left(u(t)\right) = c_1 u(t)^m + c_2 u(t)^{m-1} + c_3 u(t)^{m-2} + \ldots + c_m u(\mathrm{t}) + \mathrm{c}_{m+1}$$

- **Linear Model**: A linear transfer function with nz zeros and np poles.

$$B / \mathrm{F(z)} = \frac{b_1 z^{nz} + b_2 z^{nz-1} + \ldots + b_{nz} z + b_{nz+1}}{f_1 z^{np} + f_2 z^{np-1} + \ldots + f_{np} z + f_{np+1}}$$

- **Output Nonlinearity**: A one-dimensional polynomial of degree n.

$$h\left(x(t)\right) = d_1 x(t)^n + d_2 x(t)^{n-1} + \ldots + d_n x(t) + d_{n+1}$$

The accuracy of the SISO model for the (DPBias,RHODT) pair is 95.4%. The comparison between the reconstructed RHODT signal and original RHODT signal is shown in Figure 3.39. As can be seen, this model predicts the output very well compared to the MIMO model presented previously.

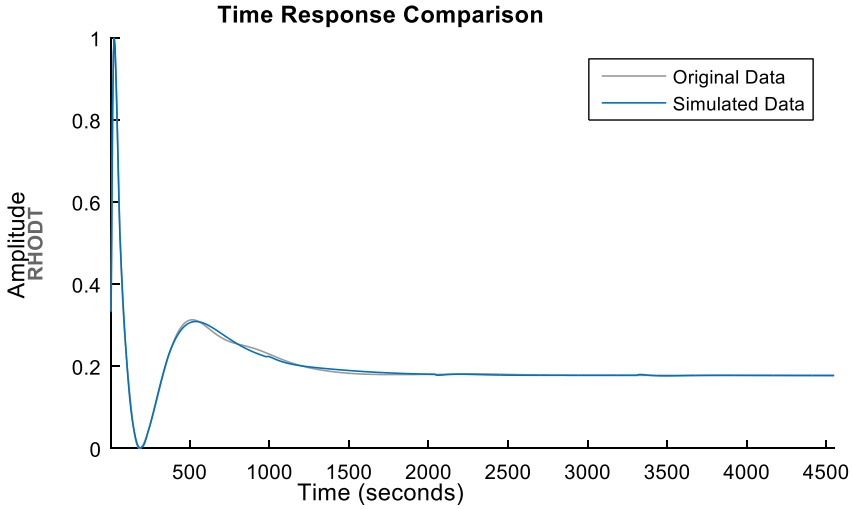


Figure 2.39: Reconstructed and Original RHODT Comparison for DPBias-RHODT Model with 95.4% Accuracy

The nonlinear and linear blocks of Hammerstein-Wiener model for this pair are shown in Figure 3.40 to Figure 3.42 .

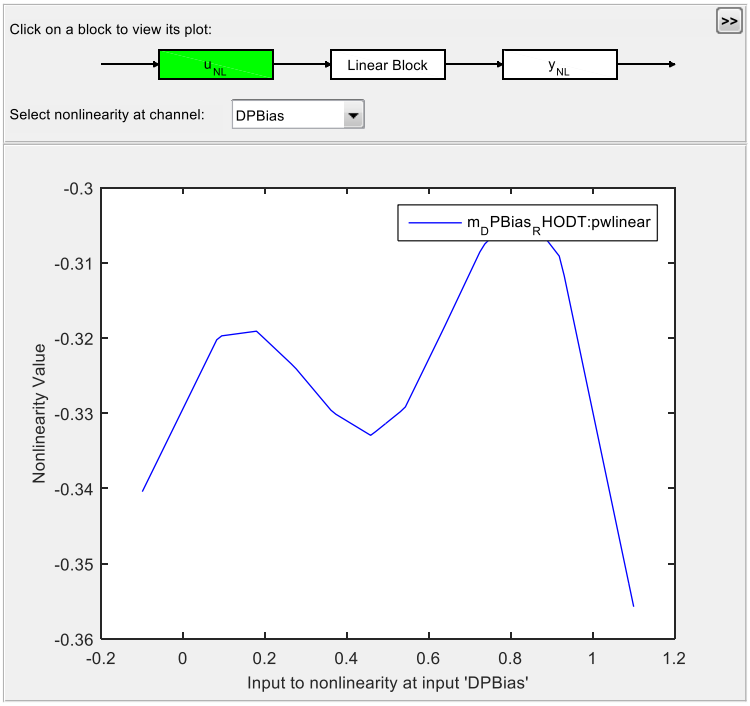


Figure 3.40: Input Nonlinearity for DPBias-RHODT Model

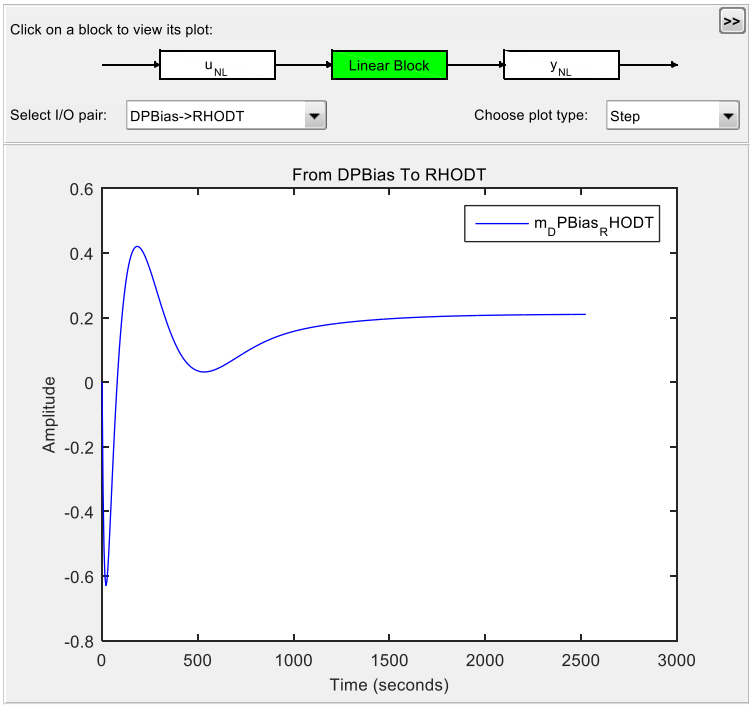


Figure 3.41: Step Response of Linear Block for DPBias-RHODT Model

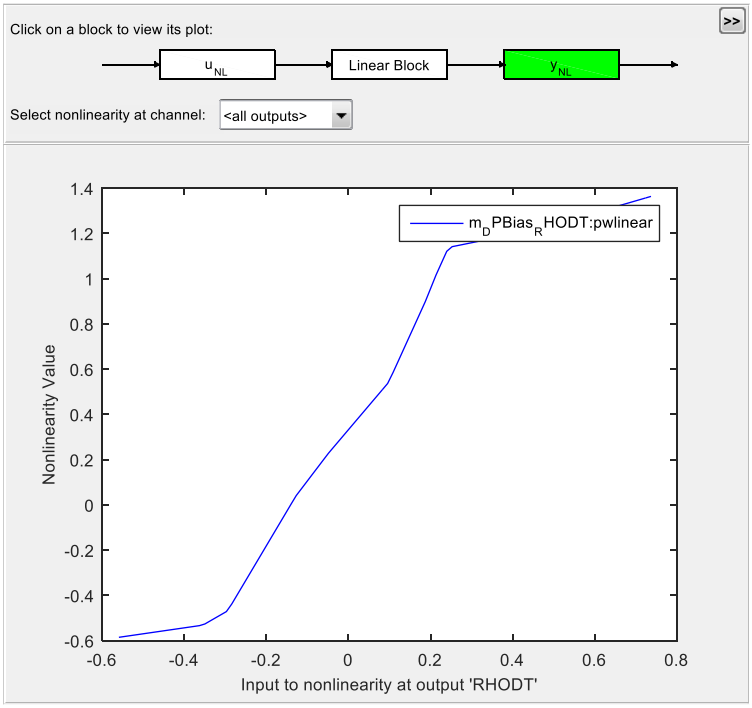


Figure 3.42: Output Nonlinearity for DPBias-RHODT Model

The estimation accuracy for all 42 models is listed in Table 3.**7**. The majority of these SISO models estimate the data precisely. These models can be used to construct customized scenarios for steam plant simulation and enables the introduction of faults into the system and hence the performance testing and evaluation of the agent-based framework for fault detection. An example of a customized input and its corresponding output is shown in Figure 3.43 and Figure 3.44, respectively.

Table 3.7: Estimation Accuracy for 7x6 SISO Models

		Output					
		SHODT	**RHODT**	**FEGT**	**NOX**	**CO**	**ExO2**
Input	**ULD**	71%	85.93%	98.8%	89.37%	82.1%	91.94%
	O2Bias	64.33%	95.3%	97.18%	94.4%	76.9%	97.86%
	DPBias	84.38%	95.4%	82.5%	99.69%	95.58%	83.51%
	WWO	63.85%	58.37%	94.82%	70.57%	58.31%	58.2%
	SOFA	72.09%	61.86%	71.97%	96.62%	95.9%	71.96%
	Tilt	59.93%	75.88%	92.65%	98.96%	72.21%	59.45
	FDT	88.65%	87.94%	89.38%	84.96%	70.13%	87.58%

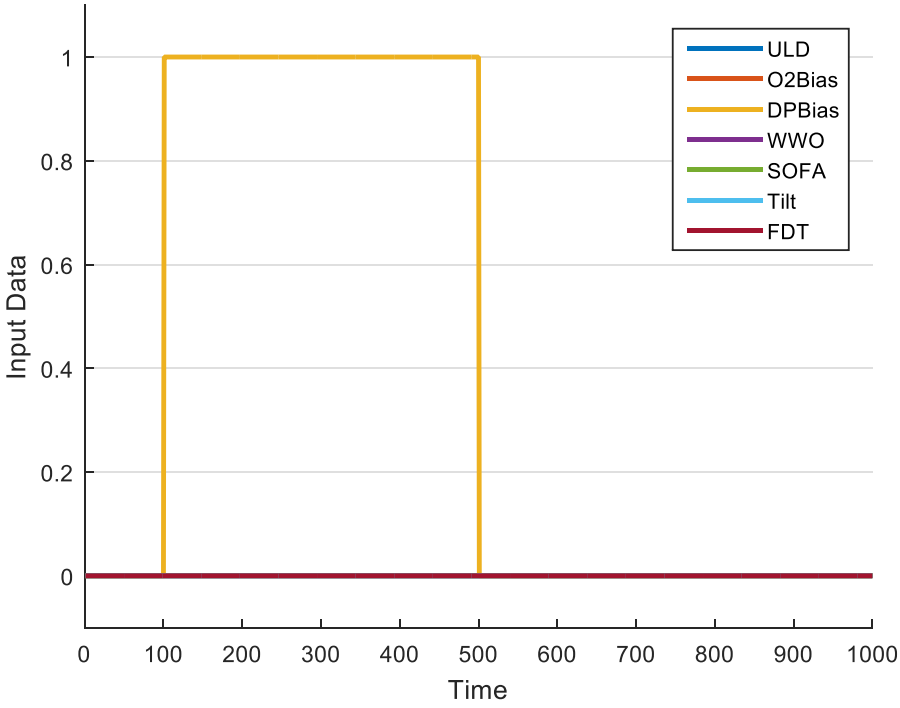


Figure 3.43: A Customized Input

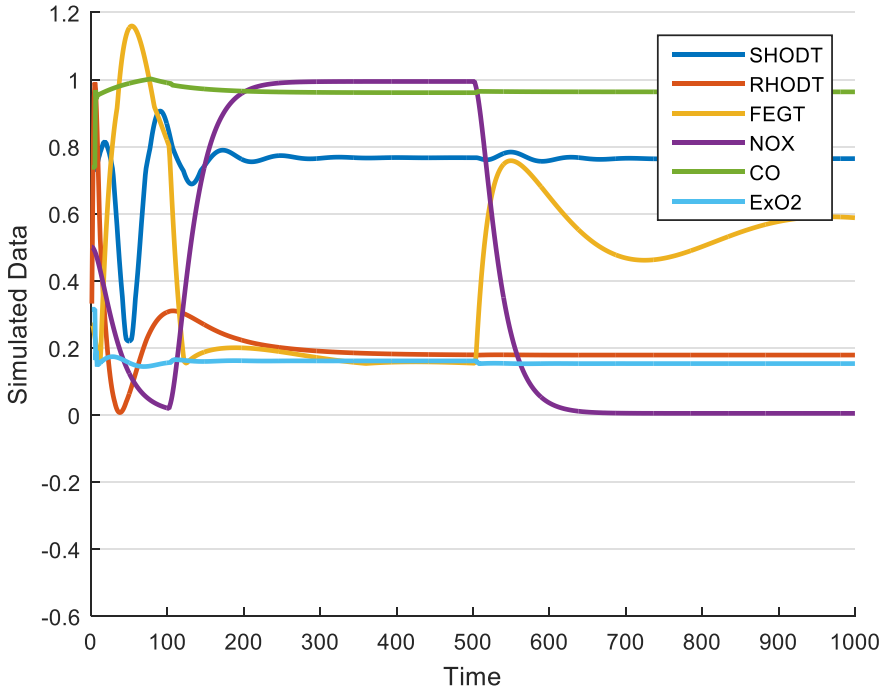


Figure 3.44: The corresponding Output

Proposed Algorithm: Algorithms based on the foraging behavior of ants is being used to discover the intrinsic communication topology between the outputs of a dynamic system, i.e. the steam plant. Based on the extracted topologies, a corresponding operating state can be identified. Note that this is a dynamic condition and changes in topologies may reflect changes in the operating state. In order to identify faults, the evolution of the intrinsic communication topology must be understood so that observed changes due to faults may be separated from those resulting from normal variation in the operating condition. A dynamic, agent-based, method for identifying the topology as it changes with time provides the ability to detect and diagnose changes. Moreover, the identification of the actual intrinsic communication topology is the first step in the design and implementation of more specific algorithms for fault detection and diagnosis as it provides the connection between observables and system elements.

In the current work, a node the represents a system input is added to the nominal system representation shown in Figure 3.19. This additional node, shown in red, in Figure 3.45 provides a mechanism to monitor the manner in which the input communicates with the system as seen via its instrumentation or output nodes, shown in blue. Only one active input is under consideration and the goal of this initial effort is to identify this input based on the discovered communication topology.

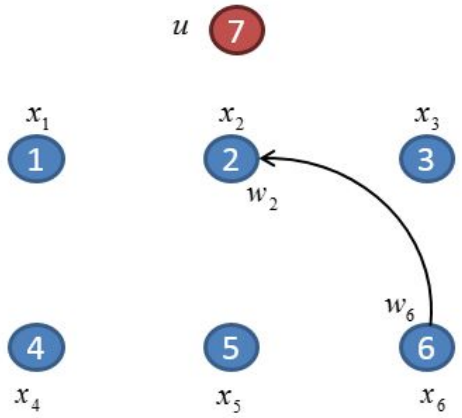


Figure 3.45: System Representation

As detailed previously, w_i is a time series that is the partial observation of the system at node i. When an agent goes from a node to another, it carries some data, described as time series w_i , from its home node to the next node. The food is defined as the “similarity” between these two time series. When the agent carries information, it looks for data containing the "same information." Moreover, it wants to preserve the dynamics of the system. To capture this similarity, correlation coefficient between two time series is computed.

As seen previously, simulation results obtained for different runs using the same data are not the same. The reason is the pheromone update and next node selection rules are probabilistic in nature and thus display randomness. The pheromone table is constructed based solely on the (random) movement of the agents. Therefore, the routes containing nonzero pheromone values are highly dependent on the nodes selected for visitation by the agents at the beginning of the simulation. In order to obtain a more regular (and predictable) behavior from the algorithm, some minor changes to the algorithm (e.g., updated exploration rules and pheromone trail update rules) are required.

Two major updates to the proposed algorithm are examined next as follows:

- **Number of Agents**

In the previous version, the number of agents was the same as the number of nodes. In this version, the number of agents was increased to approximately 700. Having more agents enables changes in the connectivity to be captured much faster and more accurately than in the base implementation. This also significantly reduces the risk associated with missing important changes in behavior.

- **Next Node Selection Rule**

  In the previous version, agents selected the next node to be visited based on the pheromone values on the ground (communications channel). In this version, the agents select the next node based on the current correlation coefficient for that time frame. They favor the nodes with higher correlation coefficient. The pheromone values on the ground are based on the previous connectivity strength between the nodes.

The updated algorithm is then as follows:

3. Agents are placed randomly on the nodes such that each node has at least one agent.
5. For agent k at node i, the next node j is chosen based on the following rule:

$$\mathrm{g}(w_i, w_u) = \left| correlation\_coefficient(w_i, w_u) \right|$$

$$j = \begin{cases} \arg \max_{u \in J_i^k} \{ \mathrm{g}(w_i, w_u) \} & if \ \mathrm{q} \le \mathrm{q}_0 \\ J & if \ \mathrm{q} > \mathrm{q}_0 \end{cases}$$

- q : a random variable uniformly distributed over $[0,1]$
- q_0 : a tunable parameter over $[0,1]$
- J_i^k : set of all the nodes in the system except for the current node i
- $J \in J_i^k$: node that is randomly selected according to this probability:

$$p_{iJ}^k(t) = \frac{\mathrm{g}(w_i, w_J)(t)}{\sum_{l \in J_i^k} \mathrm{g}(w_i, w_J)(t)}$$

6. The agent goes to node j and updates the pheromone trail of the pair (i, j) according to the following rule:

$$\tau_{ij}(t) \leftarrow (1-\rho) \cdot \tau_{ij}(t) + \left| correlation\_coefficient(w_i, w_j) \right|$$

- $correlation\_coefficient(w_i, w_j)$: correlation coefficient between the time series of the two nodes connecting the path
- ρ : pheromone decay parameter

7. Go to step 2 and repeat until the end of simulation.

Note that agents stay at the node for a limited time and then travel to the next node following the rule in step 2. They continue traversing the network until the end of the simulation.

Simulations were run for 1000s with 700 agents, zero initial pheromone on the ground and the pheromone decay parameter of 0.1. A sliding window with the length of 50s is passed through the simulation and the intrinsic communication topology for the duration of this window is discovered. Some snapshots of the simulation are shown in Figure 3.46 to Figure 3.49. The thickness of the lines in the discovered topology reflects the strength of communication along that channel. As can be seen, as the window slides through the time series data, the extracted topology changes.

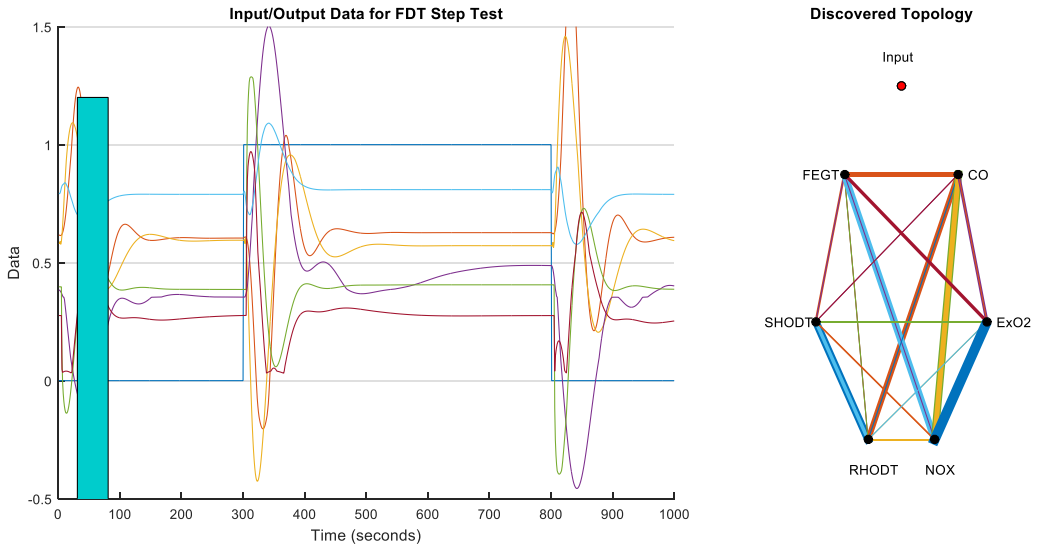


Figure 3.46: Simulation Snapshots

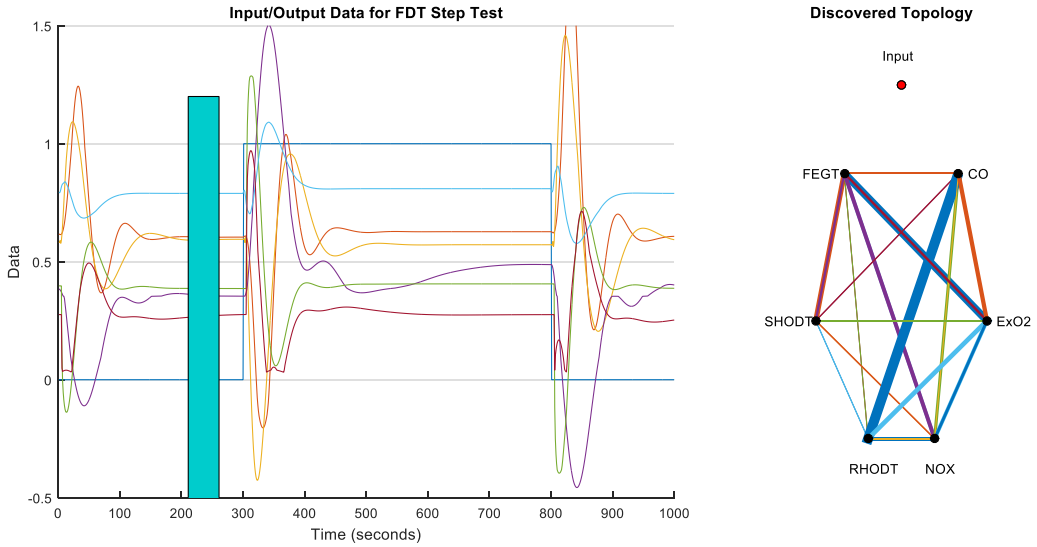


Figure 4.47: Simulation Snapshots

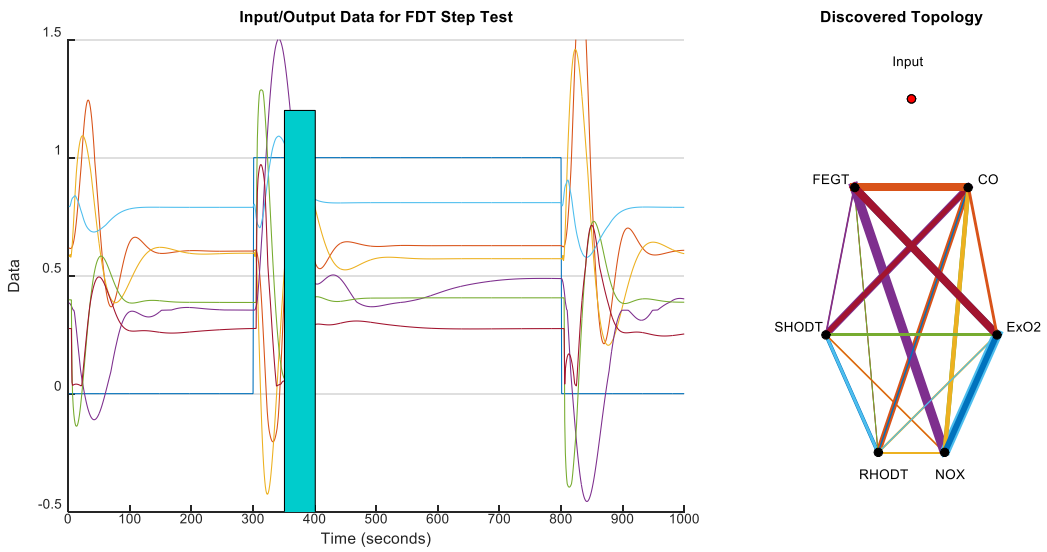


Figure 3.48: Simulation Snapshots

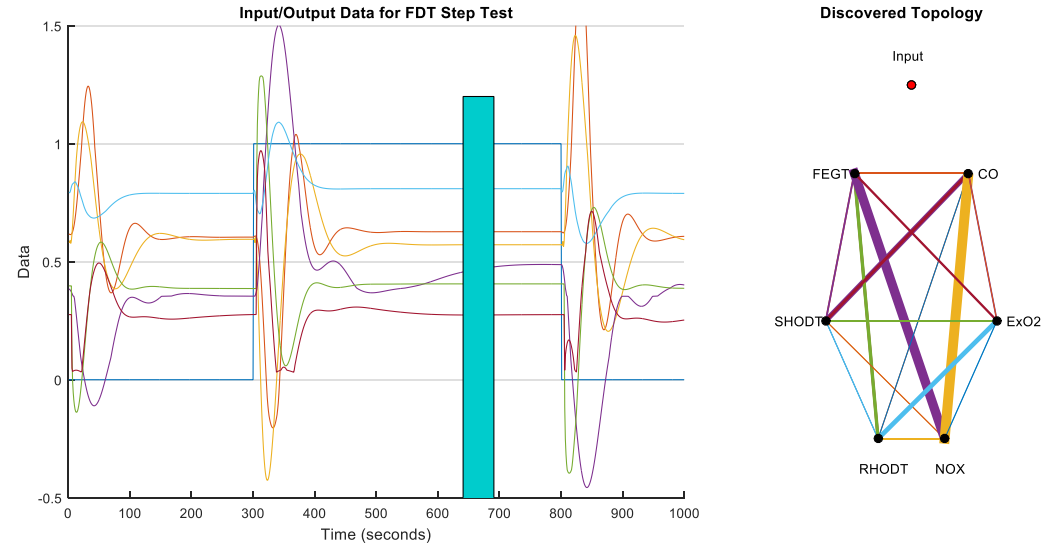


Figure 3.49: Simulation Snapshots

Exemplary Network Discovery: We consider two multivariate Gaussian distributions with different means and covariance matrices. We start sampling from one distribution and generate related data. We apply the algorithm to this data and discover its connectivity structure. We then compare the results with the actual covariance matrix of the distribution to evaluate the performance of our algorithm. After some time, we switch to the other distribution and repeat above steps to calculate its connectivity structure. This is shown in Figure 3.50.

We test the performance of our algorithm in detecting the switch from one distribution to another. This method helps us in detecting the changes in operational statuses of the power generation plant. We would like to extract distinct information for each status and infer the status of the system by monitoring those information. In this scenario, we will be able to detect errors and unexpected changes in the system configuration.

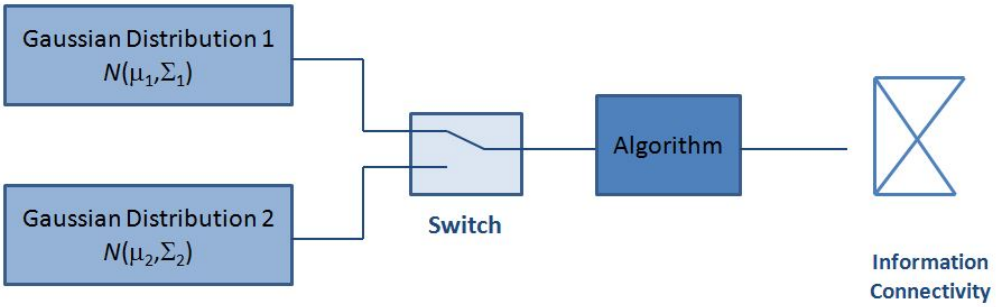


Figure 3.50: System Representation

The multivariate Gaussian distribution of a n-dimensional random vector x with mean μ and covariance matrix Σ, can be written as:

$$x: \ \mathrm{N}(\mu,\Sigma)$$

$$\mathrm{x} = \left[X_1, X_2, \mathrm{X}_3, ..., X_n\right]$$

$$\mu = \left[\mathrm{E}\left[X_1\right], \mathrm{E}\left[X_2\right], \mathrm{E}\left[X_3\right], ..., \mathrm{E}\left[X_n\right]\right]$$

$$\Sigma = \left[Cov\left[X_i, X_j\right]\right], \quad i = 1,2,3,...,\mathrm{n}; \quad j = 1,2,3,...,\mathrm{n}$$

$$p(x;\mu,\Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^n}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

where $p(x;\mu,\Sigma)$ is the density function.

As shown in the system representation in Figure 3.50, we start sampling from one distribution and at some point during the simulation we switch to sampling from the other distribution. The method for sampling a random vector x, from the N-dimensional multivariate Gaussian distribution with mean μ vector and covariance matrix Σ is as follows:

1. Apply Cholesky Decomposition to calculate the real matrix A such that $\Sigma = AA^T$.
2. Let $z = \left(Z_1, ..., Z_N\right)^T$ be a vector of N independent Gaussian variates.
3. Therefore, the sampled random vector is $x = \mu + Az$.

We consider two different 4-dimensional Gaussian distributions as our exemplary networks. There is one covariance matrix, Σ_1, associated with one distribution and another covariance matrix, Σ_2, associated with the other distribution. The means of the two distributions are the same and the covariance matrices are selected such that they are not overlapped. We consider strong correlation between two pairs and zero correlation for the rest of the pairs. The correlation structure for the other distribution is considered the opposite, i.e. the strong correlation in one distribution has zero correlation in the other and vice versa. We test the performance of our algorithm in detecting the changes in the connectivity structure with the same mean. The mean and covariance matrices of two distributions are given next.

$$x_1 : \mathrm{N}(\mu_1, \Sigma_1)$$

$$\mu_1 = \begin{bmatrix} 15 \\ 13 \\ 10 \\ 11 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 11.574 & 6.015 & 0 & 0 \\ 6.015 & 6.612 & 0 & 2.438 \\ 0 & 0 & 18.058 & 0 \\ 0 & 2.438 & 0 & 5.392 \end{bmatrix}$$

$$x_2 : \mathrm{N}(\mu_2, \Sigma_2)$$

$$\mu_2 = \begin{bmatrix} 15 \\ 13 \\ 10 \\ 11 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 11.574 & 0 & 0 & 0 \\ 0 & 6.612 & 4.128 & 0 \\ 0 & 4.128 & 18.058 & 5.872 \\ 0 & 0 & 5.872 & 5.392 \end{bmatrix}$$

We can see that the first distribution has strong correlation between the pairs $(<1,2>,<2,4>)$ whereas the second distribution has strong correlation between the pairs $(<3,2>,<3,4>)$ and the correlations for all other pairs are zero. The histogram plots for both distributions are shown in Figure 3.51 and Figure 3.52.

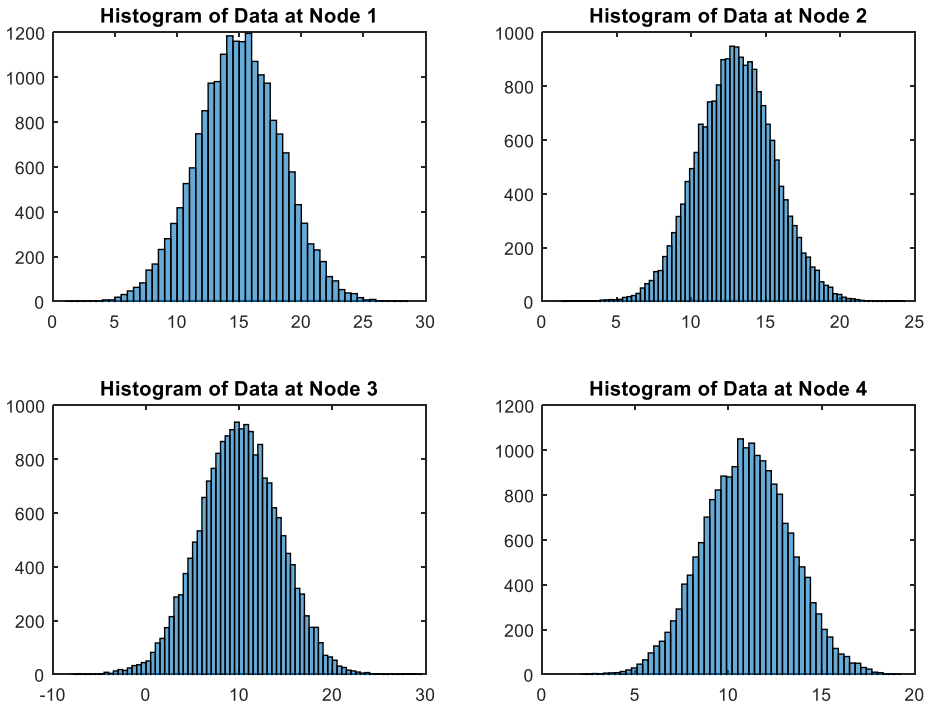


Figure 3.51: Histogram Plots for Distribution x_1

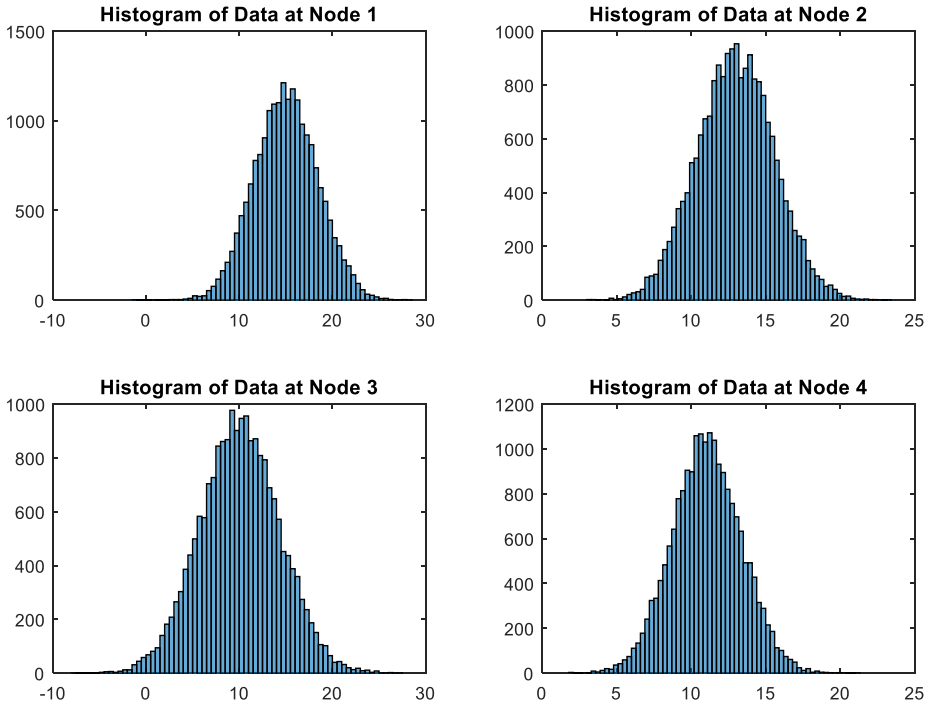


Figure 3.52: Histogram Plots for Distribution x_2

Foraging Behaviors: Our proposed algorithm discovers the connectivity structure of a system based on the foraging behavior of ants. There are some challenges in adapting foraging behavior to suit our problem and

before discussing these challenges we explain the foraging behavior of ants in more detail. Then we list the challenges and update our proposed algorithm applying some of the solutions available in the literature.

Foraging behavior is the collective behavior of ants that provides a basis for searching and optimization. This trail-laying and trail-following behavior is as follows: individual ants deposit a chemical named pheromone as they move searching for food. The pheromone level increases with traffic but dissipates over time. The pheromone marking is thus reinforced on frequently used trails and fades on infrequently used trails. The next individual ants follow the trail with higher level of pheromone thereby further reinforcing it. After some time, the shortest path is almost exclusively used.

Figure 3.53 examines the foraging patterns of three Army Ant species. Army ants are among the largest and most cohesive societies. Their foraging system coordinates hundreds of thousands of individuals and cover a thousand meters in a single day (Bonabeau, Dorigo, & Theraulaz, Swarm Intelligence, from Natural to Artificial Systems, 1999). The foraging patterns depicted in Figure 3.53 are of three army ant species, *Eciton hamatum*, *Eciton rapax* and *Eciton burchelli*.

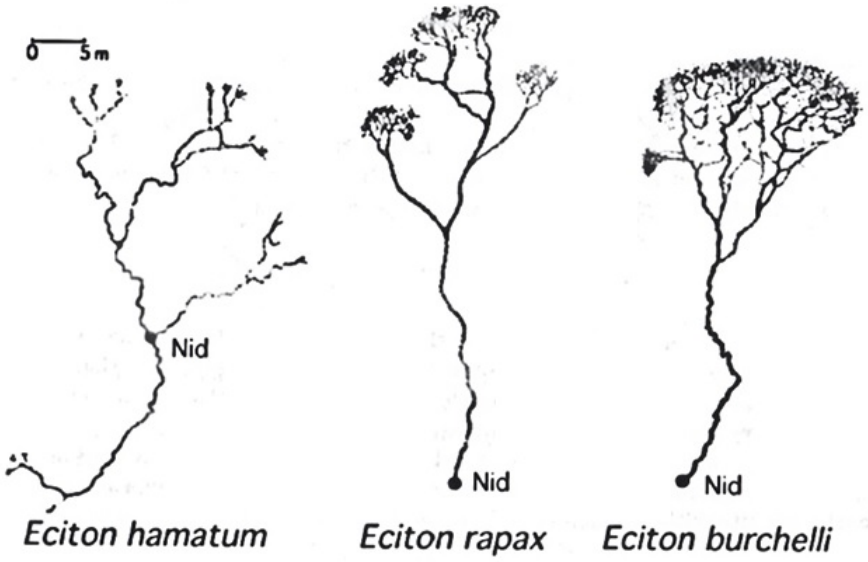


Figure 3.53: Foraging Patterns of Three Army Ant Species with Different Diets

These three species have different diets and different spatial distributions of food items. *Eciton hamatum* species feed on dispersed social insect colonies where food sources are rare but large. On the other hand, *Eciton burchelli* species feed on scattered arthropods where food can easily be found each time but in small quantities. Finally, *Eciton rapax* species have intermediary diet with intermediate food sources. These different spatial distributions explain the different foraging patterns observed in Figure 3.. Also, if all these three species have common ancestors, it is not unlikely that their behavior is similar, only their diet and environment may be different.

Deneubourg et al developed a self-organized model of army ant patterns in (Deneubourg J. , Goss, Franks, & Pasteels, 1989). In their model, ants start from the nest and move to find the food. They lay one unit of pheromone on their way to find food while deposit ten units of pheromone when they return to the nest. The probability of moving and choosing the next node is defined as a function of some parameters such as the level pheromone of the adjacent edges, the traffic at a node and the pheromone capacity of the edges. The food source can be small or large. Figure 3.54 shows two patterns with two different food distributions resulting from Monte Carlo simulation of their model.

Figure 3.54: Monte Carlo Simulation Results of the Swarm Raid Model with two Different Food Distributions

The pattern on the right side has more food units with less probability compare to the pattern on the left side. Such food distributions are similar to *E. burchelli* for the left side pattern and *E. hamatun* or *E. rapas* for the other one. The simulation results are similar to the swarm raid patterns of Figure 3.53; the swarm splits up into a couple of small columns. From Figure, one can adjust the parameters and get different raid patterns. These reconfigurable raid fronts offer tunable exploration behaviors and can be adapted to the specific of the problem at hand.

We can adjust the parameters of the probability function to adapt the behavior of the ants to our specific problem. One of the challenges that we have here is the concept of multiple food sources. When there are multiple food sources in an environment, the ants tend to exploit one food source until it is finished and then switch to the other food source. This is an important challenge that we face in our problem. Consider the Gaussian distribution for x_1. The pairs $\left(<1,2>,<2,4>\right)$ are highly correlated where the other pairs are not uncorrelated. In the eye of the ant agents, the two highly correlated pairs in this network are seen as two food sources. Therefore, they start exploiting one of the sources and when it is finished, they switch to the other one. This yields to the discovery of only one of the correlations in the extracted connectivity structure. To overcome this problem, we refer to the literature on resource exploitation strategies and load balancing techniques in communication networks.

Some of the proposed techniques in load balancing that are related to our problem includes: (Schoonderwoerd, Holland, & Bruten, Ant-like agents for load balancing in telecommunications networks, 1997), (Zhang, Long, Jianping, & Falko, 2014).

Steward and Appleby proposed using a large number of mobile agents for robustness in (Steward & Appleby, 1994). In their proposed approach, there are two species of mobile agents: load management agents and parent agents. Load management agents provide the lower level of control. They start from a node and move around the network to find the best route based on Dijkstra's shortest path algorithm. On the other hand, parent agents provide the second level of control. They travel over the network and monitor the traffic at each node. If a network management is required to relieve congestion, they travel to those locations and launch load agents.

This approach can be adapted to our network. Some of the agents could act as supervisor and monitor the traffic at nodes and n the edges. If there is congestion on an edge, it means that most of the agents are exploiting one of the food sources. Therefore, they can release appropriate load management agents to relieve the congestion.

Schoonderwoerd et al. described a novel method of load balancing in telecommunication networks in (Schoonderwoerd, Bruten, Holland, & Rothkrantz, 1996). The primary goal of their approach is to develop simple methods to encourage agents to explore short routes while avoiding heavily congested nodes. They proposed aging and delaying agents. They define the agents' age as the length of the path it has traced from the nest.

In the first method, aging the agents, the value of pheromone deposited by an agent is reduced depending on the agent's age, i.e. older agents deposit less pheromone compared to younger ones. This influences the system to respond stronger to the agents that have travelled shorter trails.

The second method, delaying agents, depends on the first method. In this approach, the agents are delayed at congested nodes. This temporarily reduces the flow rate of agents from the congested node to its neighbors, and allows the probabilities for alternate routes to increase. Also, because the agents will be older when they leave the congested node, they have less effect on pheromone levels.

In (Tonguz, 2011) a biologically inspired approach is employed to solve the traffic congestion problem by applying self-organization techniques. In this approach, a leader car at each intersection is selected in a distributed manner to act as Virtual Traffic Lights (VTL), to manage the flow of cars at that intersection by announcing the traffic condition to its neighboring cars. Based on this information, the cars approaching the intersection can seek alternate routes with lighter congestion. One of the contributions of this method is that the leader car relieves the traffic by accessing local information. This is in contrary to the approach where parent agents have access to global information.

Dussutour et al. investigated the foraging activity of an invasive ant species in (Dussutou, Nicolis, Shephard, Beekman, & Sumpter, 2009). They collected 15 colonies with 2000-300 workers and 4-6 queens in Sydney, Australia. They set up four different experiments. Their experiments showed that the ants use two different pheromone signals, one for exploration and the other for exploitation with different decay rates.
During the exploration they deposit a long-lasting pheromone, low decay rate. This rapidly establishes a new trail and acts as an external long-term memory for the colony. It ensures that when a food source is discovered, there is already an existing path to that point and the ants can quickly get there.

On the other hand, during food exploitation, they deposit a shorter-lasting pheromone that allows the colony to abandon a finished food source. They investigated the role of these pheromones under static and dynamic conditions with performing different experiments and obtained a mathematical model for their ants' behavior. They discussed that the presence of two different pheromones allows ants to track the changes in foraging conditions more quickly and more effectively than having a single pheromone in the colony.

This approach can be applied to our problem by having agents deposit stronger pheromone during exploitation of a food source and weaker pheromone during exploration.

Algorithm Updates: Based on the challenges and opportunities discussed and some trial and error, we update our proposed algorithm. Two of the changes are listed below:

- We set the value of pheromone an agent deposits as a function of its age. This is shown in the equation and in Figure 3.55.

$$\nabla\tau(age) = \frac{5 \times \text{maximum age}}{0.1 \times \text{maximum age} + age}$$

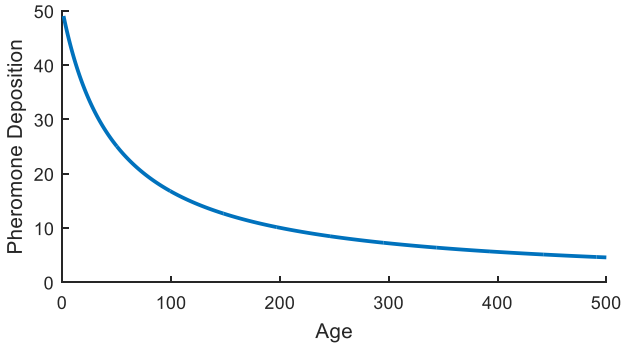


Figure 3.55: Pheromone Deposition as a Function of the Agents' Age

- A function is defined for a transition rule so that older agents are more likely to exploit while younger agents have a greater tendcy to explore.

$$j = \begin{cases} \arg\max_{u \in J_i^k} \left\{ [\tau_{iu}(t)] \cdot [\eta_{iu}]^{\beta} \right\} & \text{if } q < q_0(age) \\ J & \text{if } q \geq q_0(age) \end{cases}$$

where,

$$q_0(age) = \frac{\log_2(age)}{\log_2(\max(age))}.$$

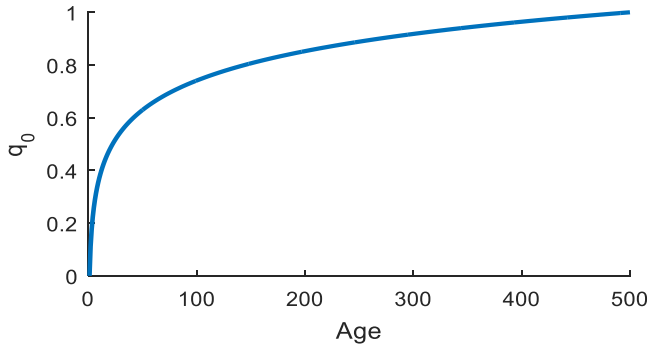


Figure 3.56: q_0 as a Function of the Agent's Age

In the updated algorithm, we consider n types of pheromone based on their home node. Each type deposits a distinct pheromone that is accumulated and followed only by the same type. We also divide the data into windows of size T. Agents are introduced to the network at the start of each time window and grab a window of data from their home node and start navigating forward through the network. At each new node, they calculate the correlation coefficient between their carried home node data and the data present at that node. If the correlation coefficient is higher than $p_{desirable}$, they have reached their destination. They now become backward agents and travel to their home node from the same path they took to this node. During their backward travel, they deposit higher amount of pheromone compare to their forward travel. This is to emphasis the path that leads to the node with high correlation to their home nodes. Once they arrive at their home node, the convert back to forward agents and start navigating the network by following the path with stronger pheromone. We assume that the agents keep their window of home node data till the end of simulation and never replace it with another one.

The algorithm has the following steps:

1. At time windows w_t agents are placed randomly on the nodes such that each node has m agents.
2. For agent k with age of age at node i, the next node j is chosen based on the following rule:

$$j = \begin{cases} \arg\max_{u \in J_i^k} \left\{ \left[\tau_{iu}(t)\right].\left[\eta_{iu}\right]^{\beta} \right\} & \text{if } q < \dfrac{\log_2(age)}{\log_2(\max(age))} \\ J & \text{if } q \geq \dfrac{\log_2(age)}{\log_2(\max(age))} \end{cases}$$

- q : a random variable uniformly distributed over $[0,1]$
- J_i^k : set of all the nodes in the system except for the current node i
- $J \in J_i^k - \arg\max_{u \in J_i^k} \left\{ \left[\tau_{iu}(t)\right].\left[\eta_{iu}\right]^{\beta} \right\}$: node that is randomly selected according to this probability:

$$p_{iJ}^k(t) = \frac{\left[\tau_{iJ}(t)\right].\left[\eta_{iJ}\right]^{\beta}}{\sum_{l \in J_i^k} \left[\tau_{il}(t)\right].\left[\eta_{il}\right]^{\beta}}$$

4. Upon arrival at the next node, the agent calculates the correlation coefficient between its carrying data and the data present at that node. The forward flag of the agent is then updated and the forward agents continue searching for the next node while the backward agent travels back to its home node following the same path it took to reach its destination node.

$$\text{Agent's Flag} = \begin{cases} \text{Forward} & if \quad \left|correlation\_coefficient(w_i, w_u)\right| \geq p_{desirable} \\ \text{Backward} & if \quad \left|correlation\_coefficient(w_i, w_u)\right| < p_{desirable} \end{cases}$$

5. The agent goes to node j and updates the pheromone trail of the pair (i, j) according to the following rule:

$$\tau_{ij}(t, age) = \begin{cases} (1-\rho)\cdot\tau_{ij}(t-1) + \dfrac{0.5 \times \text{maximum age}}{0.1 \times \text{maximum age} + age} & if \quad \text{forward agent} \\ (1-\rho)\cdot\tau_{ij}(t-1) + \dfrac{5 \times \text{maximum age}}{0.1 \times \text{maximum age} + age} & if \quad \text{backward agent} \end{cases}$$

ρ : pheromone decay parameter

6. Go to step 2 and repeat until the end of simulation.
   Note that agents stay at the node for a limited time and then travel to the next node following the rule in step 2. They continue travelling through the network until the end of the simulation.

Simulation Results: We start with the Gaussian distribution x_1 : $\mathrm{N}(\mu_1, \Sigma_1)$ and sample 500 data points from the distribution and apply our proposed algorithm. We assume zero initial pheromone on the ground, a pheromone decay parameter of 0.2 and window sizes of 50 data points. At the end of the simulation we will have around 9000 agents travelling between 4 nodes.

Travel History of the Agents: The travel history of four agents of different types between iterations 100 and 150 is shown in Figure 3.57 through Figure 3.60. As discussed, there is high correlation between the pairs $(<1,2>,<2,4>)$ while the other pairs are not correlated at. Therefore, we expect more agents to travel between the highly correlated nodes, and this can be seen from the figures.

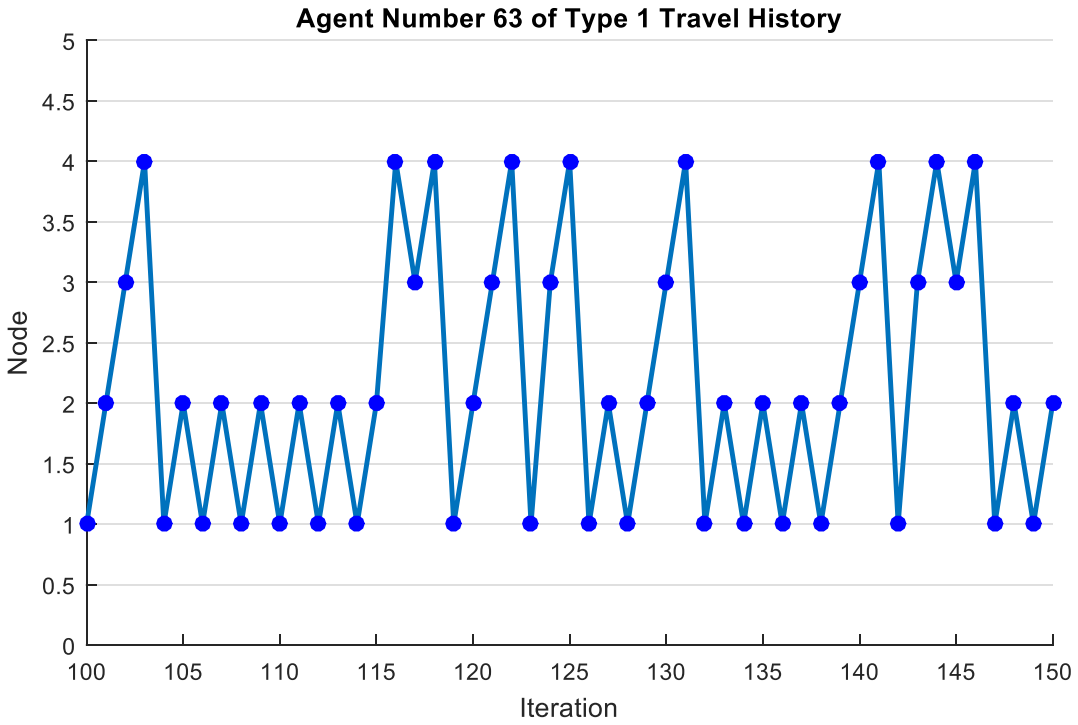


Figure 3.57: Travel History of a Type 1 Agent From Iteration 100 to 150

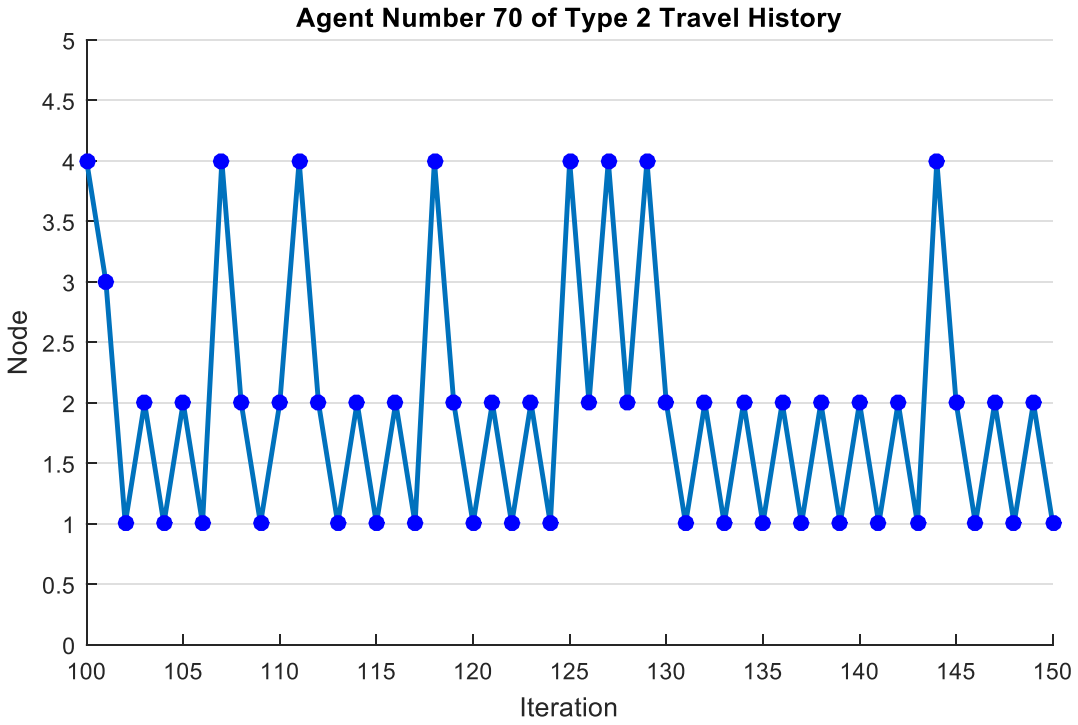


Figure 3.58: Travel History of a Type 2 Agent From Iteration 100 to 150

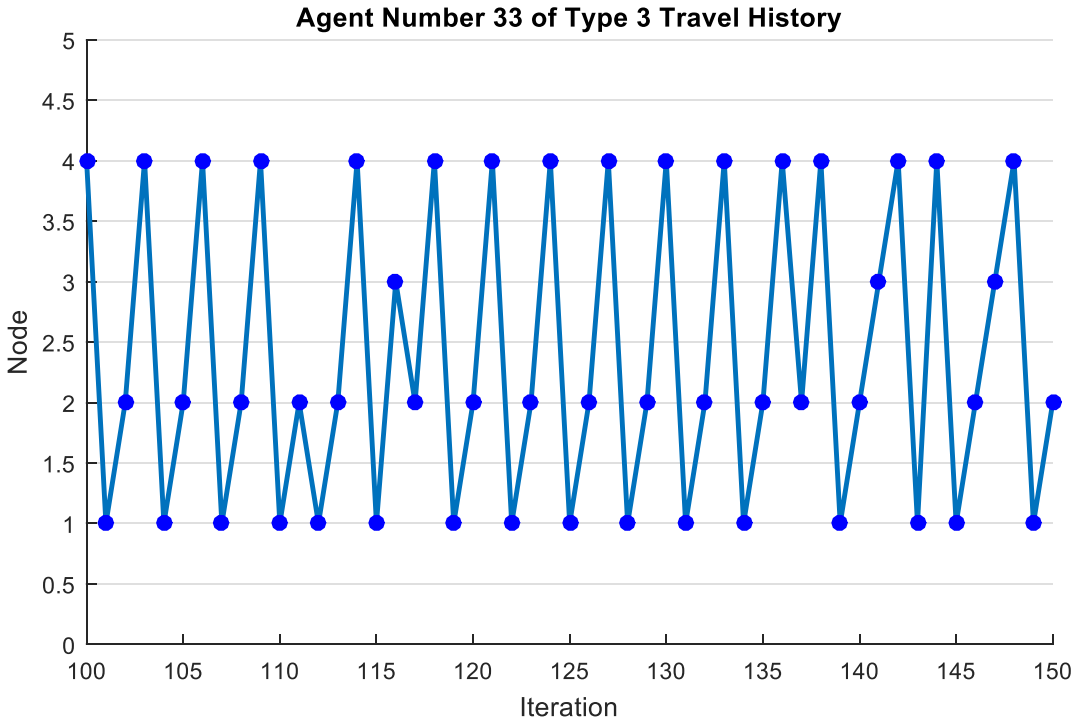


Figure 3.59: Travel History of a Type 3 Agent From Iteration 100 to 150

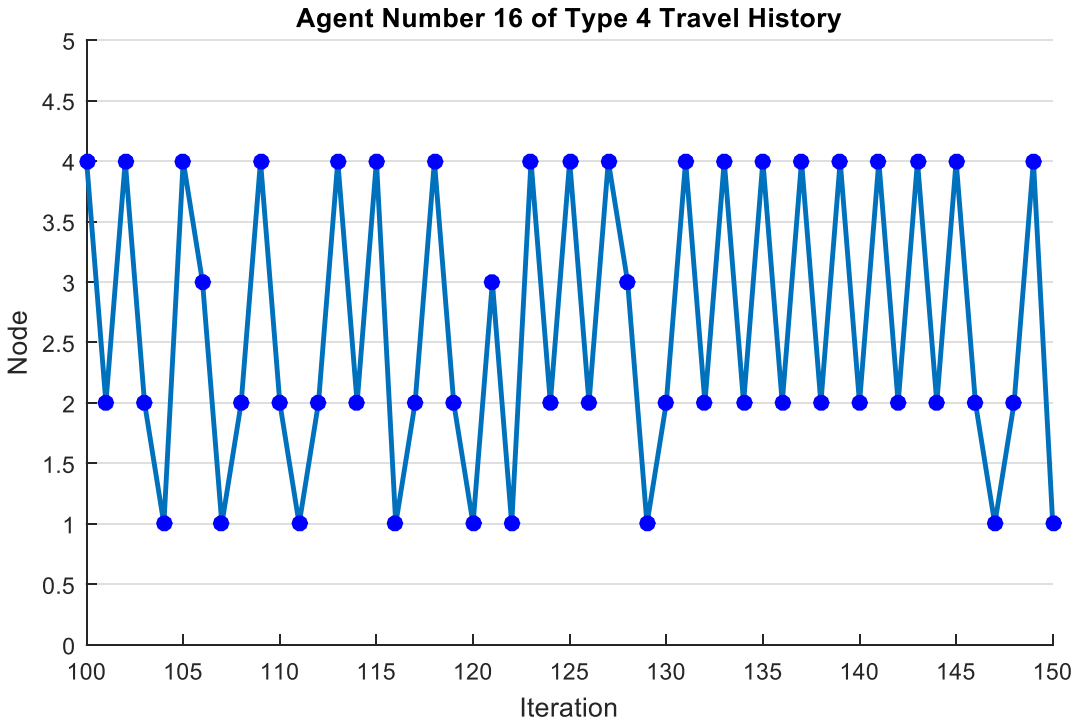


Figure 3.60: Travel History of a Type 4 Agent From Iteration 100 to 150

Pheromone Values on the Edges: Monitoring the changes in pheromone values on the edges provides valuable information about the connectivity strength on the edges. We can see the changes in the connectivity during simulation time. Figure 3.61 shows the pheromone values from node 2 to 3 and also from node 3 to 2 in the same plot. This is the summation of all four pheromone types on the edges. The first thing to observe from this figure is that there is directionality in the information flow between the nodes. The

information flow from node 2 to node 3 is different from node 3 to node 2. On the other hand, these two pheromones do not have completely different patterns. In fact, the pheromone from node 2 to 3 has around 70 iterations delay from the other pheromone.

We repeat the plot for the pheromone between nodes 1 and 2, which has higher correlation in the original Gaussian distribution. One can see from Figure 3.62 that the pheromone level is around 10 times stronger than the pheromone in Figure 3.61. We have plotted the pheromone on all the edges for comparison. It is clear from this figure that the correlation between the pairs $(<1,2>,<2,4>)$ is higher than the other pairs.

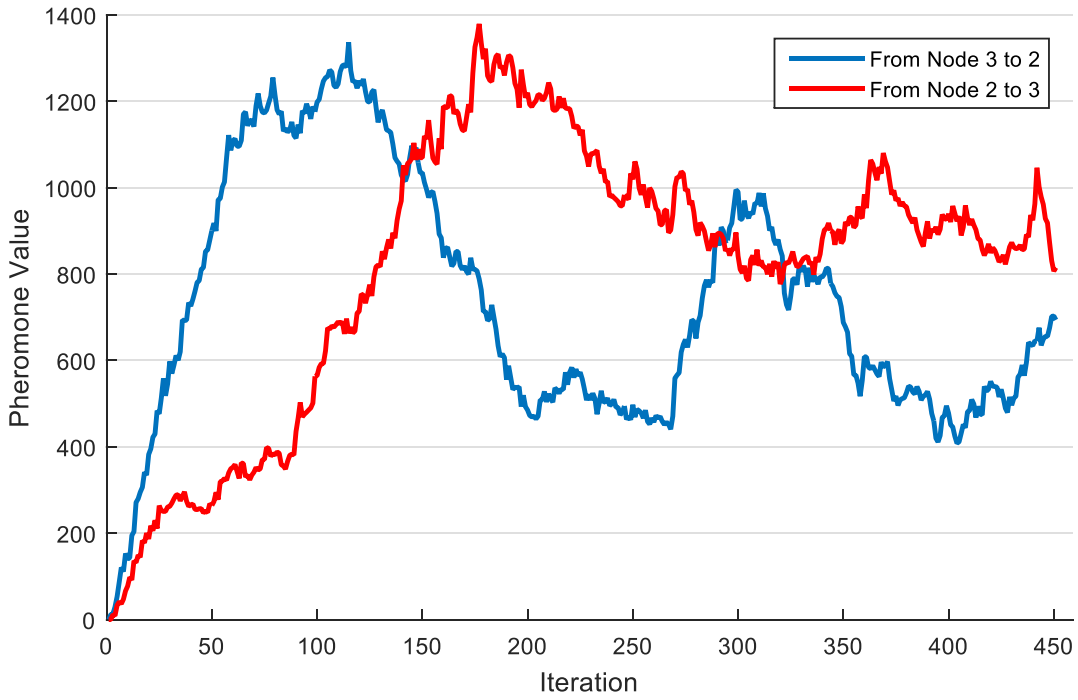


Figure 3.61: Pheromone Levels Between Nodes 2 and 3

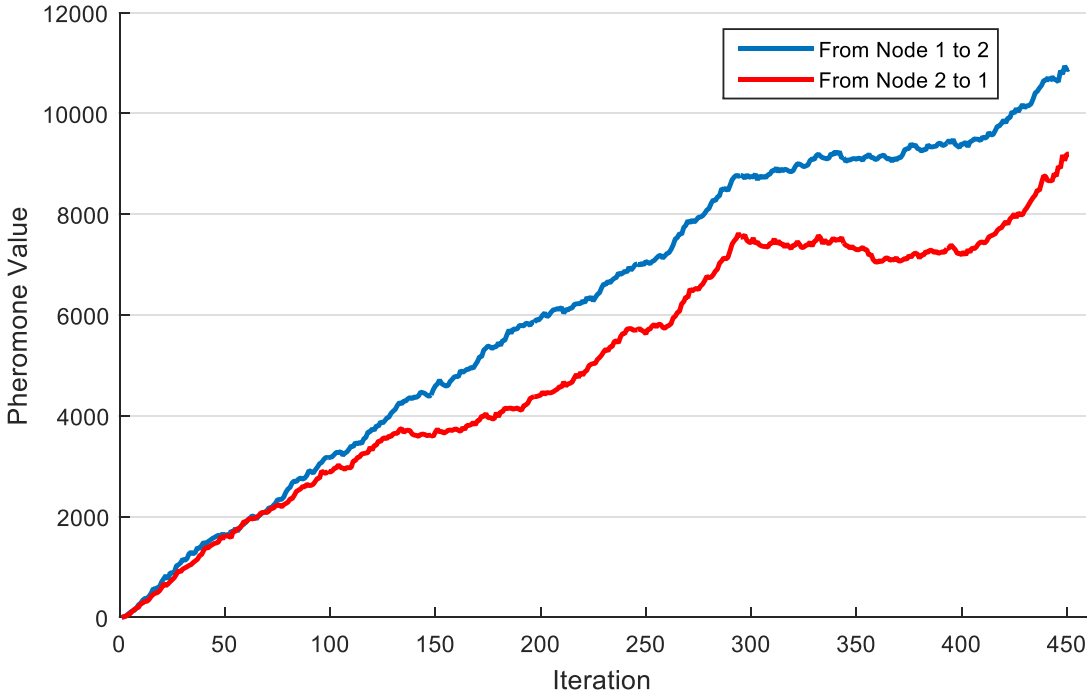


Figure 3.62: Pheromone Levels Between Nodes 1 and 2

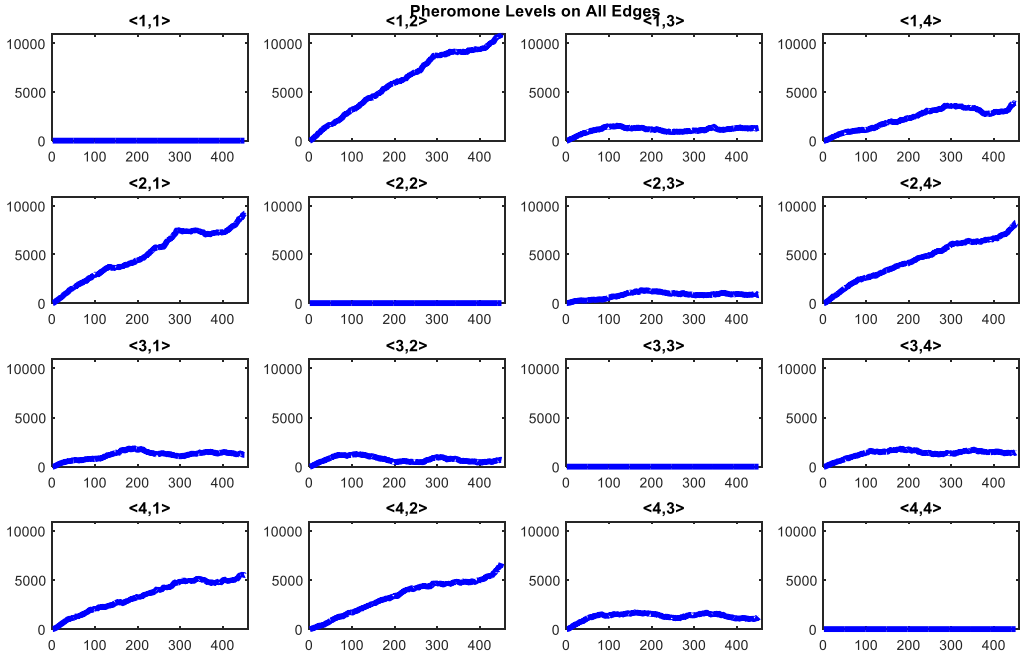


Figure 3.63: Pheromone Value on All the Edges

Connectivity Structure: Figure 3.64 shows the connectivity structure between all the nodes for all 4 types and both directions. The strength of the line between two nodes is related to the pheromone levels on the edge between them. We can see that the connectivity is stronger between $(<1,2>,<2,4>)$ pairs. We can also see that there is some connectivity between nodes $<1,4>$. Although this connection was not present

in the original Gaussian distribution, the algorithm is to quantify between strong and weak connectivity as required.

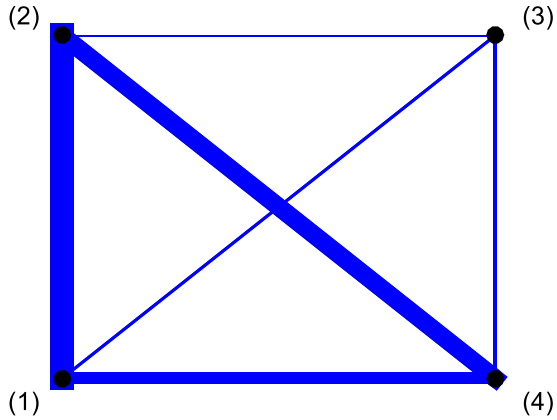


Figure 3.64: Connectivity Structure

To compare the role of different types of agents and their different pheromones, we show the connectivity structure for each agent type.

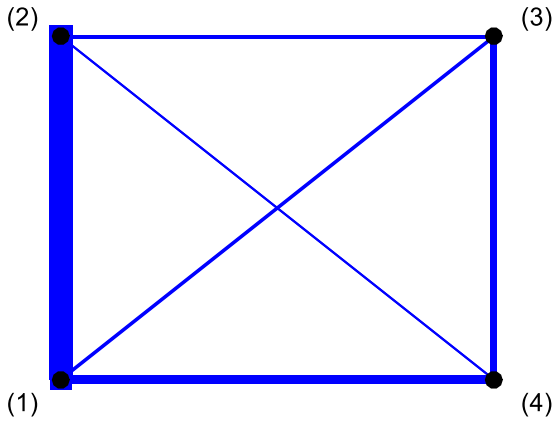


Figure 3.65: Connectivity Structure of Type 1 Agents

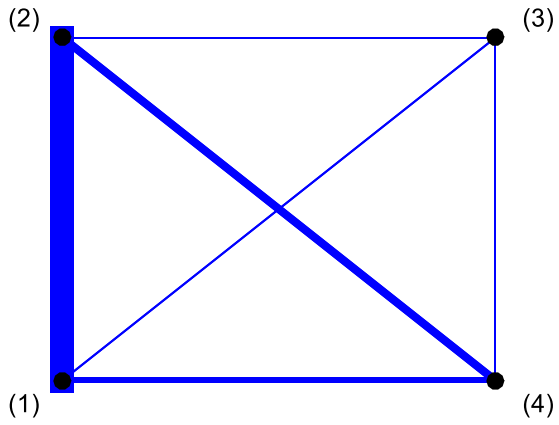


Figure 3.66: Connectivity Structure of Type 2 Agents

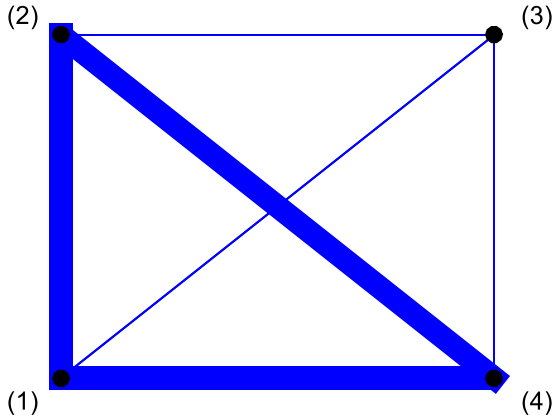


Figure 3.67: Connectivity Structure of Type 3 Agents

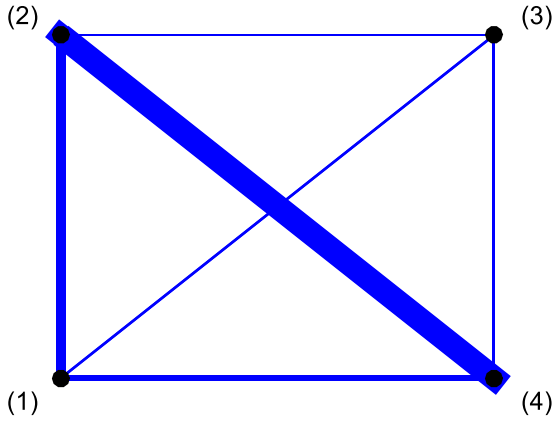


Figure 3.68: Connectivity Structure of Type 4 Agents

Note that most of type 3 agents in Figure 3.67 do not travel back to their home node, which is node 3. The reason is that the correlation coefficient between node 3 and other nodes is so small that almost all of the agents remain forward agents. They follow the path $<1-2-4-1>$ with high pheromone concentration.

We repeat the same simulation for the same Gaussian distribution but this time we sample 1500 data points from the original distribution. The results are not as good as 500 iterations. At the end of the simulation, around 29,000 agents travel through the network. Figure 3.69 and Figure 3.70 show the pheromone on all the edges for all types of agents and the extracted connectivity structure, respectively.

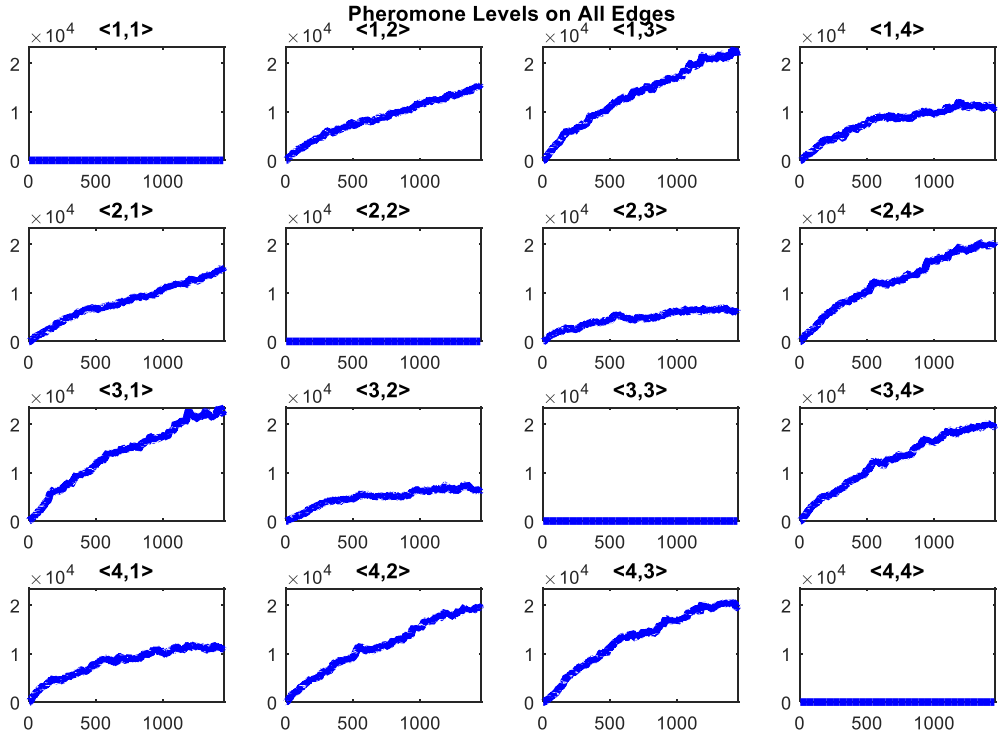


Figure 3.39 Pheromone Values on all the Edges

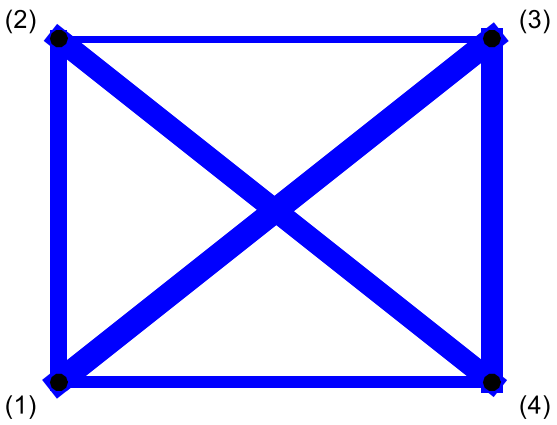


Figure 3.70: Connectivity Structure

For the next step, we repeat the same simulation for Gaussian distribution $x_2 : \mathrm{N}(\mu_2, \Sigma_2)$. Figure 3.71 and Figure 3.72 show the connectivity structure for 500 and 1500 sampled data points. We can see that the strength of connectivity between the nodes is not related to the original Gaussian distribution. The primary reason is that because the system is dynamic, the parameters that work for one condition will not necessarily work for another condition. We should develop a method to tune these parameters based on the system dynamics. Only in that case we can get accurate results for different dynamics. This is our goal for this quarter. After achieving the desired results, we start switching from one distribution to another and evaluate the performance of our algorithm in detecting the switch.

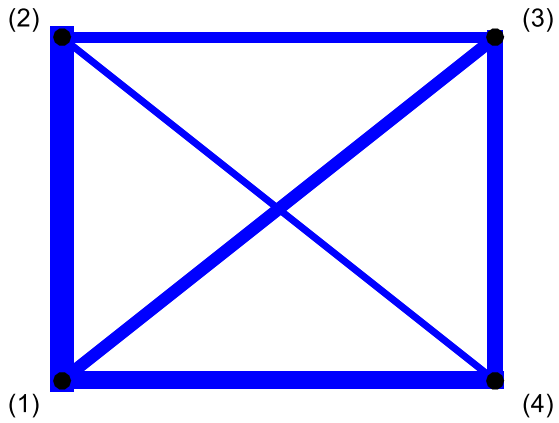


Figure 3.71: Connectivity Structure For 500 Sampled Data Points for Distribution x_2

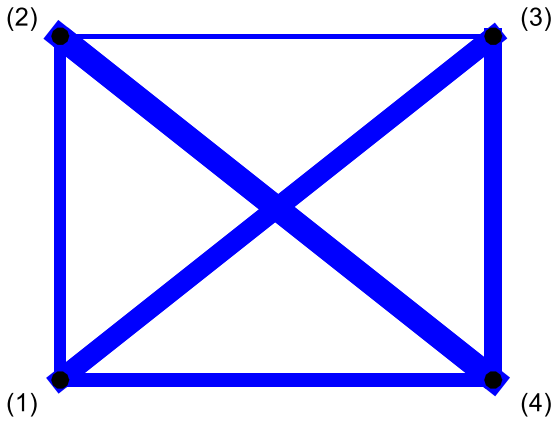


Figure 3.72: Connectivity Structure For 1500 Sampled Data Points for Distribution x_2

Power Generation Test System: As noted above, the data used for the work presented herein is from a simulation of Alstom's 1000 MWe ultra-supercritical boiler and steam plant performed using a dynamic process simulator (Yang, Lou, Neuschaefer, Boisson, & Bories, 2013). In this application, we use the information discovery algorithm to identify changes in the system operating point, i.e. changes in system inputs, from system outputs. In section 5.0, we will turn out attention to fault detection.

The steam generator produces steam flow to a turbine generator with boiler outlet conditions of the main steam flow of 600^c at 58 bar g. The plant net heat rate is 9045 kJ/Kwh. The dynamic model of the steam plant with its inputs and outputs is shown in Figure 3.16 and Figure 3.73. The inputs and outputs are listed in

Table 3.8 and

Table 3.9, respectively.

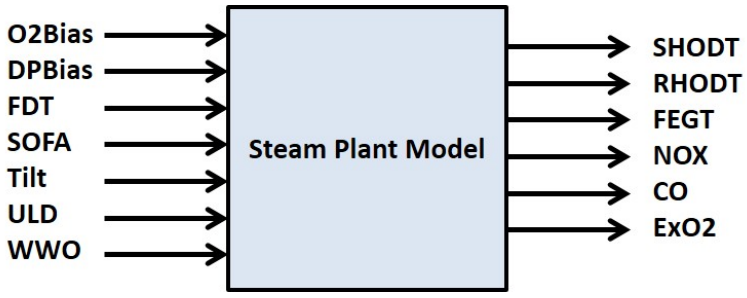


Figure 3.73: Dynamic Simulator of Steam Plant with Inputs/Outputs

Table 3.8: Steam Plant Model Inputs

Inputs	
O2Bias	O_2 Setpoint Bias
DPBias	Furnace Differential Pressure Setpoint Bias
FDT	Final desuperheater Temperature
SOFA	Separated overfire air damper bias
Tilt	Main Wind Box Tilt
ULD	Unit Load Demand
WWO	Waterwall Outlet

Table 3.9: Steam Plant Model Outputs

Outputs	
SHODT	Superheater Outlet Temperature Deviation
RHODT	Reheater Outlet Temperature Deviation
FEGT	Final Exhaust Gas Temperature
NOX	Stack NOx
CO	Stack CO (constraint)

ExO2	Excess O_2 (constraint)

A step test sequence for each input of Figure 3.73 is generated to identify the model between each input/output pair. For the initial condition, the dynamic simulator is settled down at one operating point. The step test is then performed and the outputs are recorded. Therefore, for the dynamic simulator of Figure , 7 step tests are performed and for each operating state, 6 outputs are recorded. The normalized step test for O2Bias is shown in Figure 3.75.

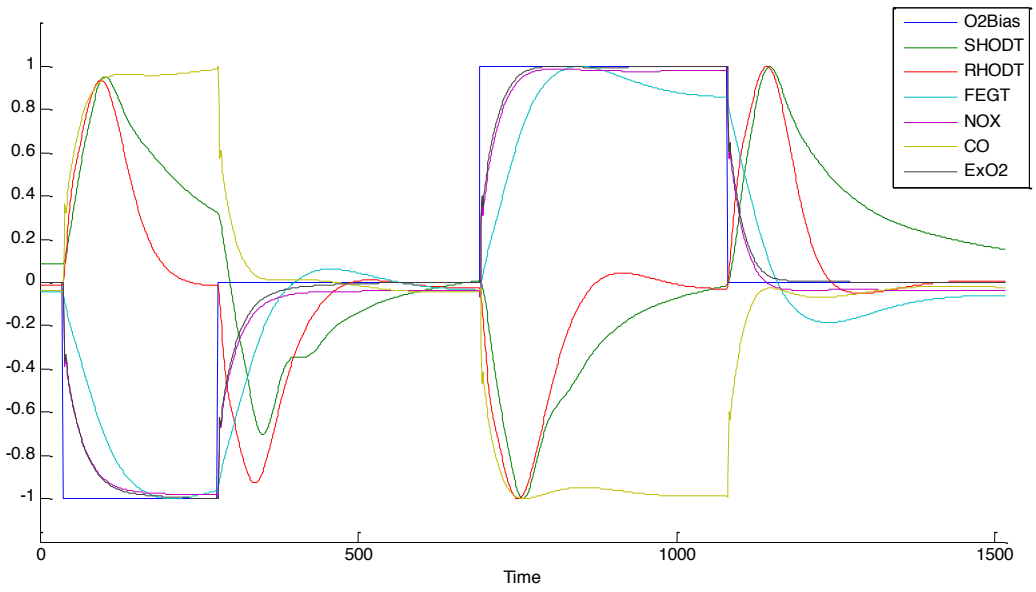


Figure 3.74: Normalized O2Bias Step Test Results

We consider a network consisting of n nodes and communication links defined in terms of pheromones associated with agents traversing the network. There are n types of pheromone corresponding to each of the n nodes that an agent can be spawned from, called the agent's home node. That is, each node spawns its own collection of agents, each of which deposits a pheromone unique to the agents' home nodes. The pheromone accumulated along a potential communication link is detectable to only agents of like type.

The amount of pheromone deposited by an agent is commensurate with the degree of communication between the home node and the other nodes in the network. As noted above, the measure of communication used herein is linear correlation. The correlation computations are performed using time series data obtained via sliding windows of size T along the data streams. Agents are introduced to the network at the start of each time window and grab the segment of data available within the window at their home node and begin to traverse the network. At each new node, they calculate the correlation coefficient between the home node data that they carry and the data present at that node. If the correlation coefficient computed at a given is higher than a threshold value $p_{desirable}$, the agents have found "food" and thus have completed foraging. The agents now become "backward" agents and return to their home node in a single iteration. During this return step, they deposit a higher amount of pheromone than they do during their forward travel. This serves to emphasize the path corresponding to the communication between the home node and the nodes containing data with high correlation to that from the home nodes. Upon their return to their respective home nodes, the agents revert to being forward agents and begin to again traverse the network, following the path with higher pheromone concentrations.

It is assumed that the agents retain their window time series data from their respective home node until the end of simulation.
Table 3.10 presents pseudo code for the proposed algorithm.

Table 3.10: High-Level Algorithm Description

Algorithm Intrinsic Communication Topology Discovery

/***Initialization***/
For every edge (i, j) **do**
$\tau_{ij}(0) = 0$
End For
/***Main Loop***/
For $t = 1$ to $t_{\max}$ **do**
For node 1 to n **do**
Place m new agents on each node
End For
For $k = 1$ to $t \times nm$ **do**
If forward agent **do**
Agent k at node i chooses the next node j, $j \in J_i^k$, as follows

$$j = \begin{cases} \arg\max_{u \in J_i^k} \{\tau_{iu}(t)\} & if\ q < 1.3 - e^{-\beta \times age}; \\ J & if\ q \geq 1.3 - e^{-\beta \times age}, \end{cases}$$

where $J \in J_i^k - \arg\max_{u \in J_i^k} \{\tau_{iu}(t)\}$ is chosen according to the probability:

$$p_{iJ}^k(t) = \frac{\tau_{iJ}(t)}{\sum_{l \in J_i^k} \tau_{iJ}(t)}$$

After each transition, agent k updates its direction flag based on the correlation coefficient between its carrying data and the data present at that node as follows

$$\text{Flag} = \begin{cases} \text{Forward} & if\ \left|\text{corr}(w_k, w_j)\right| \geq p_{desirable}, \\ \text{Backward} & if\ \left|\text{corr}(w_k, w_j)\right| < p_{desirable}. \end{cases}$$

After each transition, agent k updates the pheromone trail on the edge (i, j) with pheromone decay parameter ρ according to the rule:

$$\tau_{ij}(t) \leftarrow (1-\rho) \cdot \tau_{ij}(t-1) + 5e^{-0.0004 \times age}$$

Else
Go back to home node and update the pheromone trail on the edge (i, j) with pheromone decay parameter ρ according to the rule:

$$\tau_{ij}(t) \leftarrow (1-\rho) \cdot \tau_{ij}(t-1) + 50e^{-0.0004 \times age}$$

Update the direction flag to a Forward Flag.
End If
End For
For every edge (i, j) **do**

$$\tau_{ij}(t+1) = \tau_{ij}(t)$$

End For

End For

Print the pheromone values on all the edges

Stop

/*Values of parameters used in experiments*/

$n = 4, m = 6, \beta = 0.0004, p_{desirable} = 0.4, \rho = 0.2$

The algorithm presented in
Table 3.10 was applied to the step test data from the power generation plant simulation described in above. The system is conceptualized as a network with a total of 13 nodes, 7 input nodes and 6 output nodes. The simulation is initiated with 20 agents at each node for a total of 260 agents over the entire network. The agents traverse the network and discover the intrinsic communication topology of the power plant per the algorithm described above. At the end of the simulation, approximately 13,000 agents are travelling through the network. The initial pheromone level present along the network edges are set to zero and the pheromone decay rate is set to 0.2 (i.e. 20% of the pheromone dissipates at each time step). A sliding window of length 20s is passed over the time series data generated by the simulation to extract the data sequences carried by the agents.

Snapshots of the simulation for WWO step test data are shown in Figure 3.75 to Figure 3.83. The step test data is shown on the left hand side with the thickest line representing WWO signal. The right hand side of each figure shows the topology elicited (in terms of pheromone concentration) for a given set of windowed data. The thickness of the link between a pair of nodes corresponds to the amount of communication between the two nodes. For example, in Figure 3.75, there is high communication between SHODT and NOX output nodes while the communication between WWO and DPBias is negligible.

From these figures, it can be seen that, as the window slides, the extracted topology changes. A short time after WWO signal changes its values, new communication links are generated between WWO node and some output nodes. However, after some time, the topology settles and these communication links vanish.

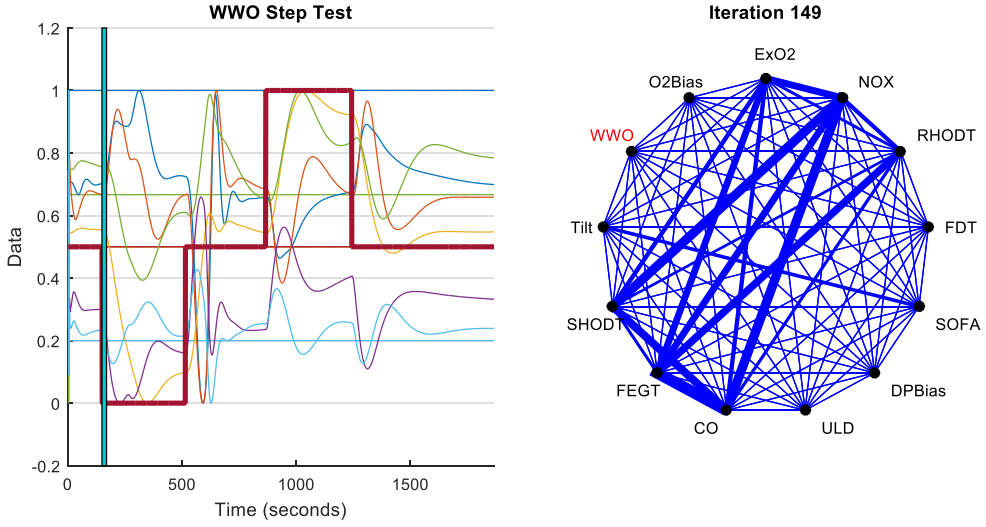


Figure 3.75: WWO Signal Changing From 0.5 to 0

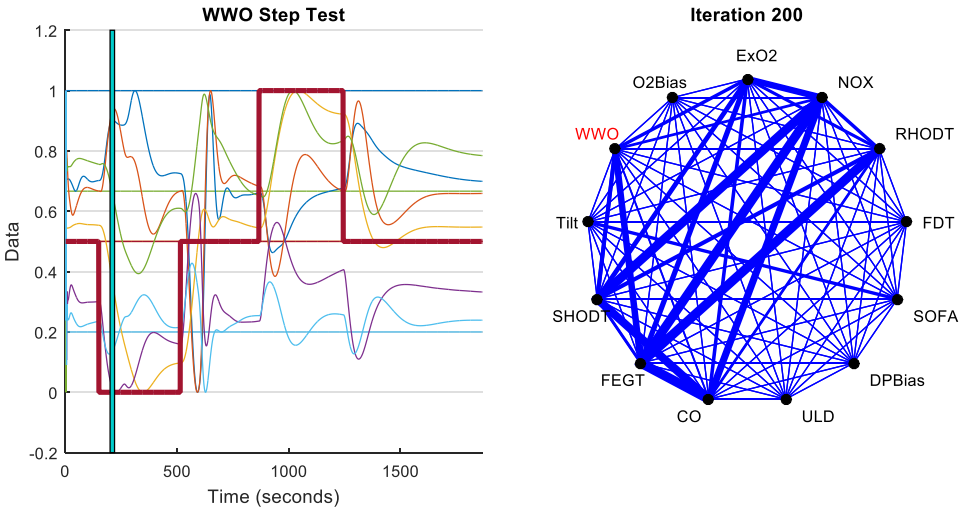


Figure 3.76: Shortly After The Change

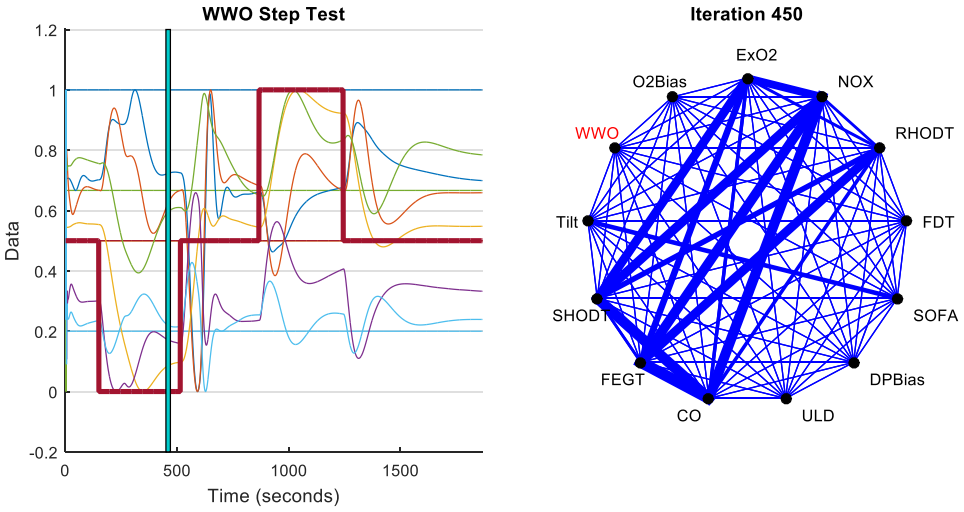


Figure 3.77: The Steady-State Topology

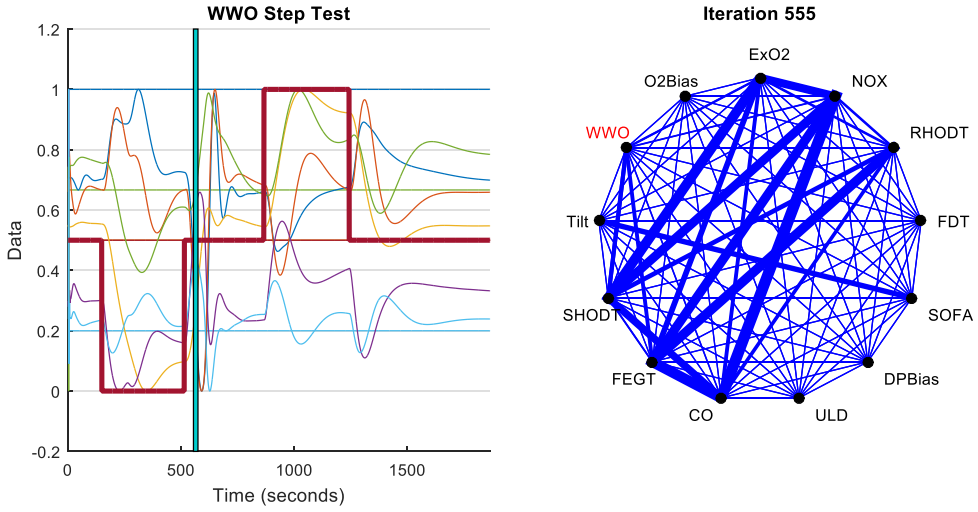


Figure 3.78: Shortly After The Change

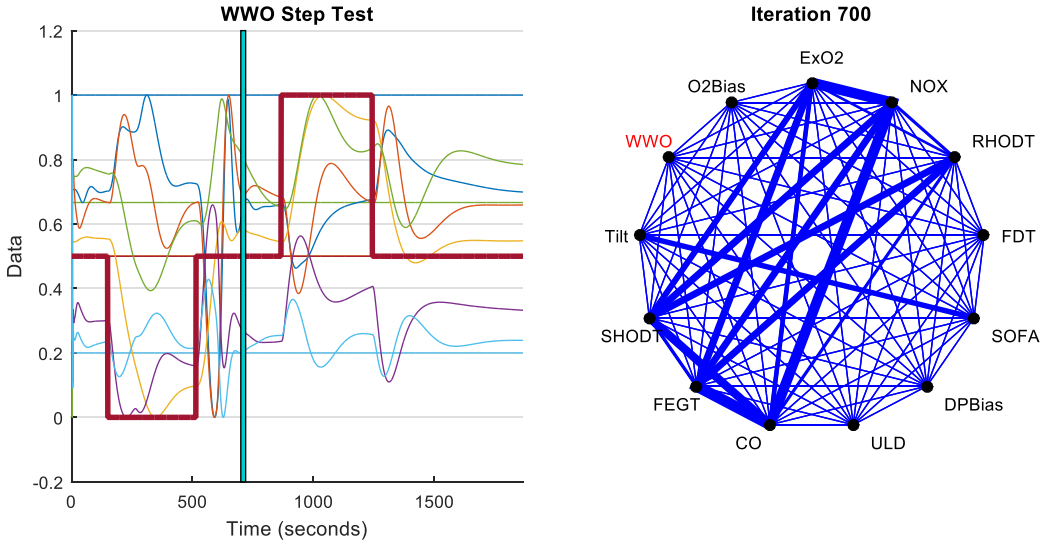


Figure 3.79: The Steady-State Topology

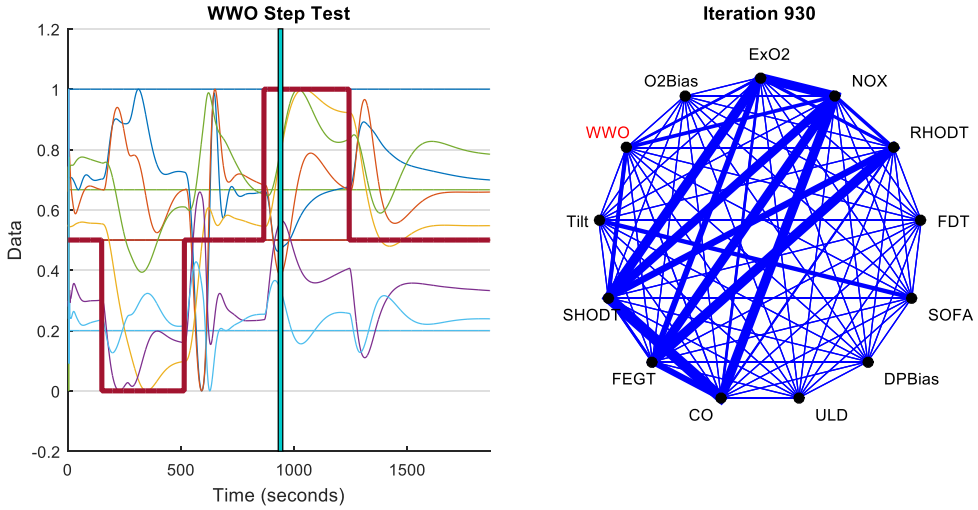


Figure 3.80: Shortly After The Change

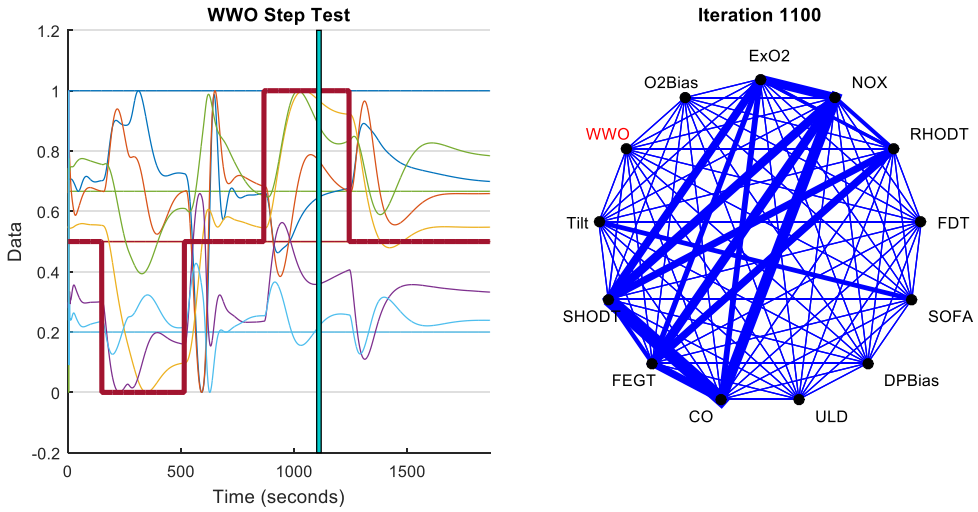


Figure 3.81: The Steady-State Topology

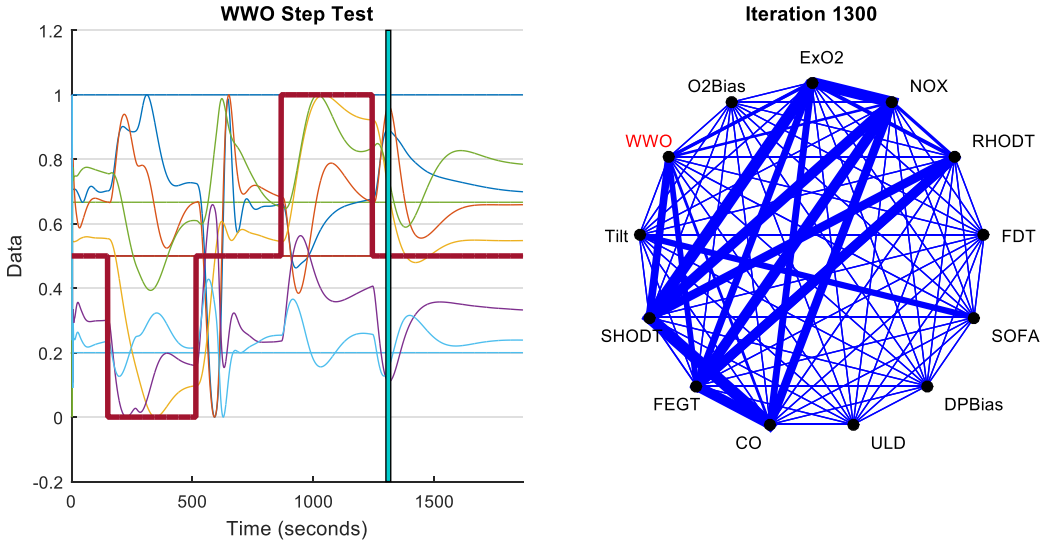


Figure 3.82: Shortly After The Change

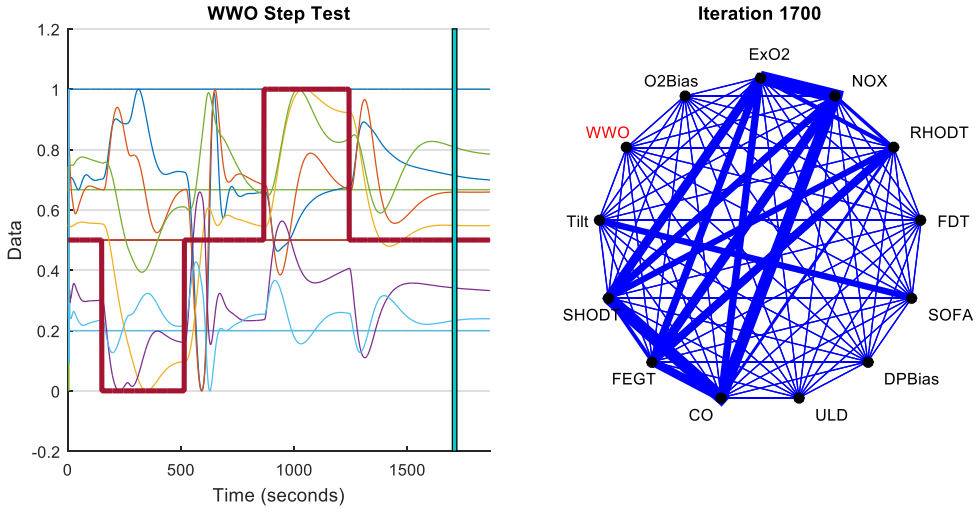


Figure 3.83: The Steady-State Topology

Figure 3.84 to Figure 3.88 illustrate the intrinsic communication topology for O2Bias step test data. As can be seen, this experiment produces results similar obtained using data from the WWO step test. If the steady-state topologies for WWO and O2Bias step test data are compared, it can be seen that they are almost the same for both step tests. This reflects the fact that merely changing the input does not change the system structure. In the more general, nonlinear case, however, it is possible that changes in the system inputs will alter the apparent communication topology.

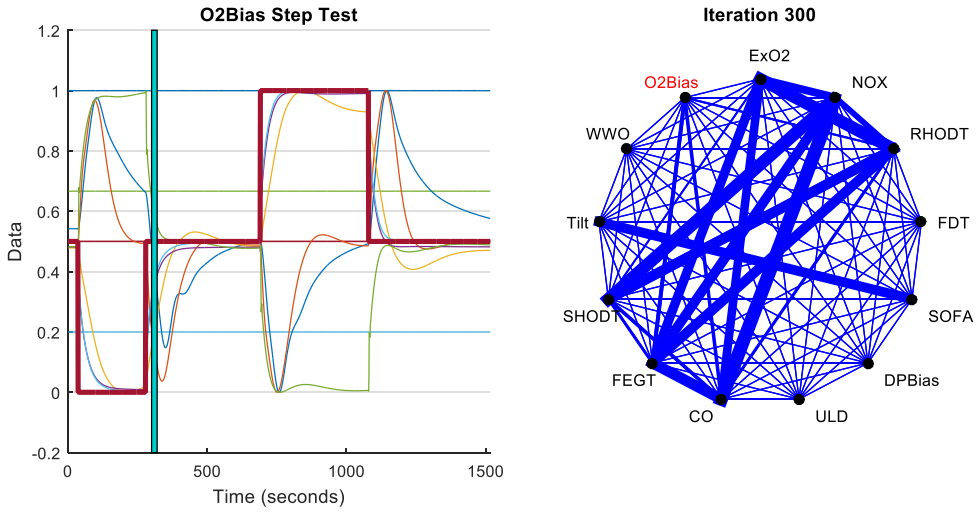


Figure 3.84: Shortly After The Change

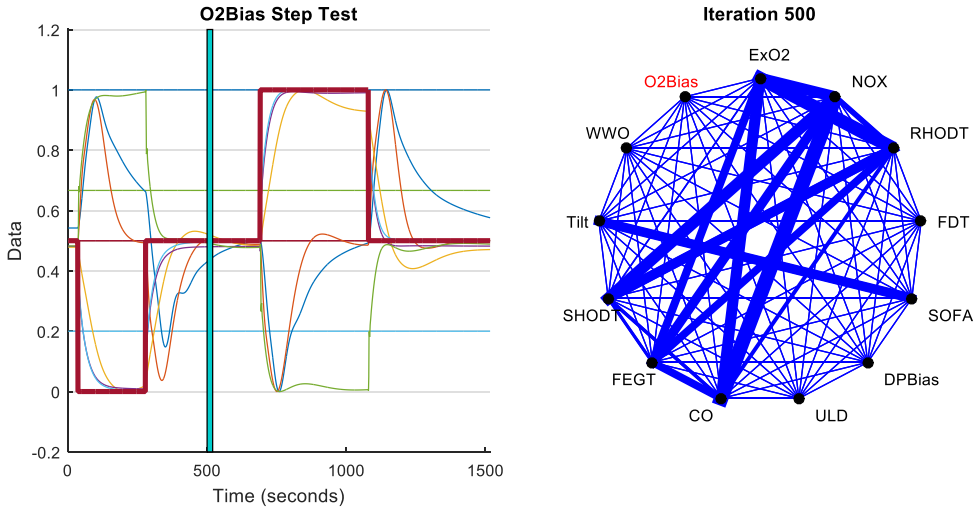


Figure 3.85: The Steady-State Topology

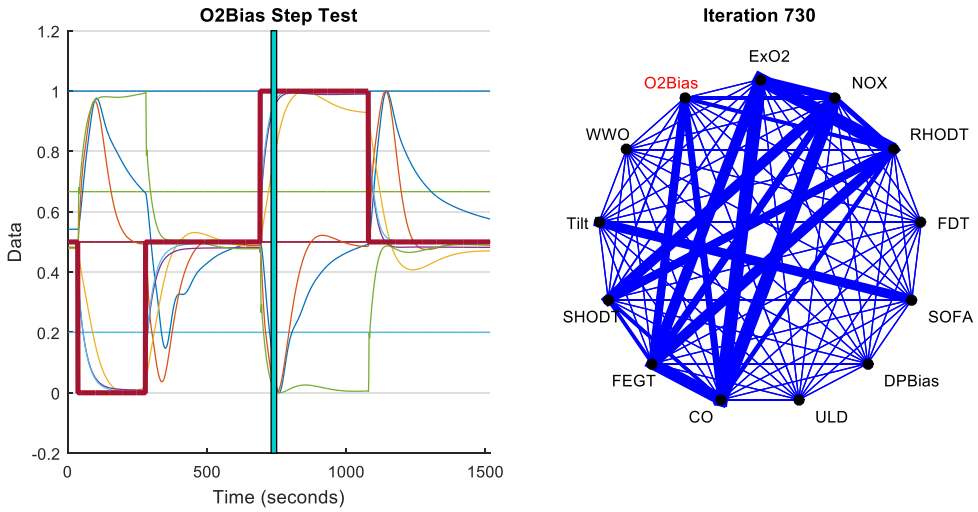


Figure 3.86: Shortly After The Change

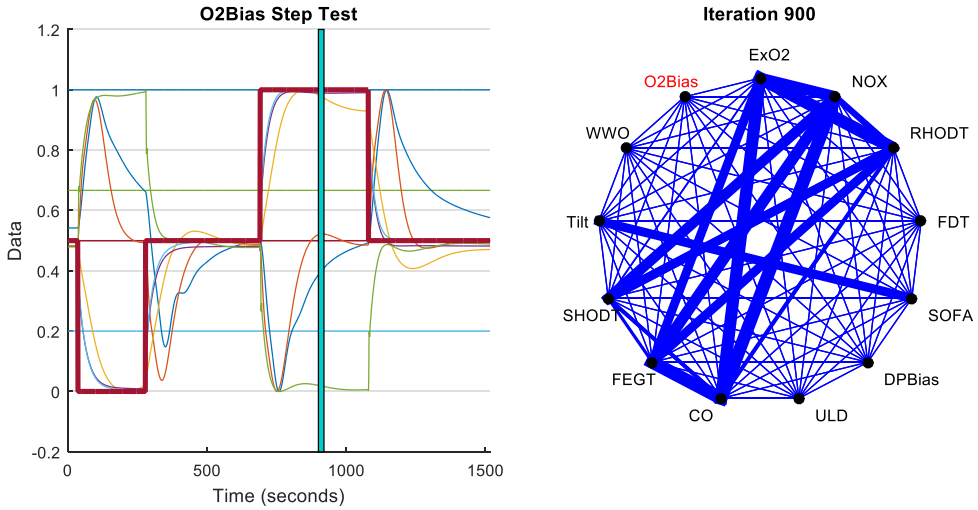


Figure 3.87: The Steady-State Topology

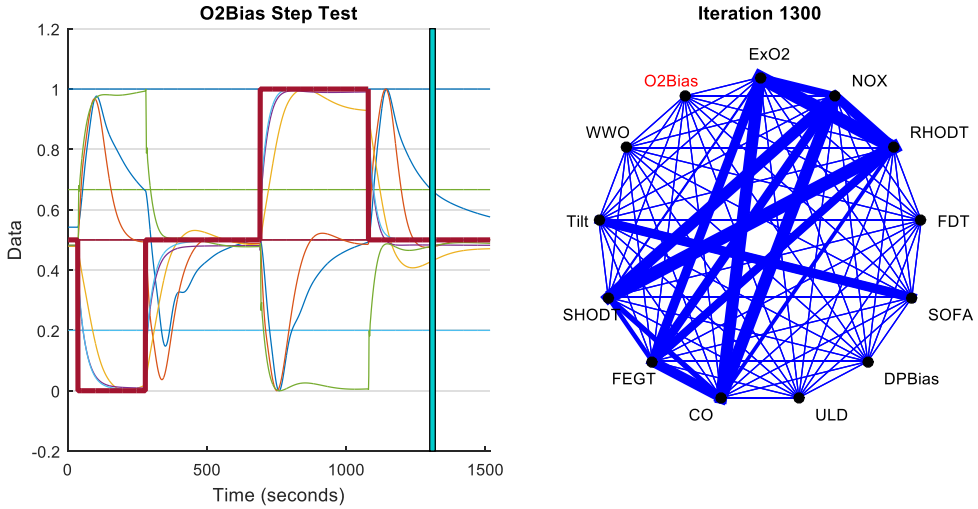


Figure 3.88: The Steady-State Topology

With a means of identifying the intrinsic communication topology of the power generation system, we now need a metric for comparing disparate topologies. That is, a metric with utility for detecting changes in topology is needed. To this end, **Graph Similarity** techniques are used to detect changes in the extracted topology. This metric provides a means, for example, of detecting when the value of O2Bias signal changes.

Commonly used measures for comparing graphs, i.e. collections of nodes connected by edges, are known as **Graph Distance Metrics**. A metric on set A is a function $d : A \times A \rightarrow \mathbb{R}$. Some commonly used graph metrics are listed in Table 3.**11** (Pincombe, 2006), (Akoglu & Faloutsos, 2013)). This table indicates that parameters of a graph are used for each metric. As topology elicited by the algorithm examined above is a weighted directed graph, only the metrics that use edges weight, i.e., Weight, MCS Weight, Modality, Entropy and Spectral metrics, are capable of using the available information from our algorithm. The focus

of this work is therefore restricted to these metrics and a side-by-side comparison of these methods shows that the Spectral metric produces the best results for the cases considered.

Table 3.11 - Graph Similarity Metrics

Metric	Vertices Used?	Edges Used?	Vertex Weights Used?	Edge Weights Used?	Range	Value for Identical Graphs
Weight	No	Yes	No	Yes	[0,1]	0
MCS Weight	No	Yes	No	Yes	[0,1]	0
MCS Edge	No	Yes	No	No	[0,1]	0
MCS Vertex	Yes	No	No	No	[0,1]	0
Graph Edit	Yes	Yes	No	No	[0,∞)	0
Median Edit	Yes	Yes	No	No	[0,∞)	0
Modality	No	Yes	No	Yes	[0,1]	0
Diameter	Yes	Yes	No	No	[0,∞)	0
Entropy	No	Yes	No	Yes	(-1,1)	0
Spectral	No	Yes	No	Yes	[0,1]	0

Spectral similarity measures were originally introduced as an intra-graph measure (Jurman, Visintainer, & Furlanello, 2011). Pincombe mentioned it as an inter-graph measure for evaluating the changes in time series of graphs in (Pincombe, 2006). One of the most common applications for graph similarity measures is for feature extraction. This application domain exploits the fact that, in graph theory, the vector of a graph's connection and Laplace matrices eigenvalues both contain important information about the graph. Moreover, the eigenvalues associated with the graph Laplacian possess an invariance property that permits local graph features to be captured. For this reason, we consider spectral methods associated with the graph Laplacian.
The Laplacian matrix for a given graph G is defined as:

$$L(G) = D(G) - A(G)$$

where D(G) is the degree matrix and A(G) is the adjacency matrix of the graph G.

The degree d_i of a vertex is the number of edges incident with i, i.e. sum of the absolute values of the weights of the edges incident with the vertex i. The degree matrix is the diagonal matrix with d_i as the i^{th} diagonal element. The adjacency matrix for the extracted communication topology in our project is the same as the connectivity matrix for that topology.

The Spectral distance metric uses the eigenvalues of the Laplacian spectra in measuring the similarity between two graphs. For a weighted directed graph, the Laplacian matrix formula is defined as (Bunke, Dickinson, Kraetzl, & Wallis, 2007):

$$L(G) = D(G) - \frac{A(G) + A(G)^T}{2}$$

For example, the Laplacian matrix for the graph in Figure 3.89 can be calculated via in a straightforward manner.

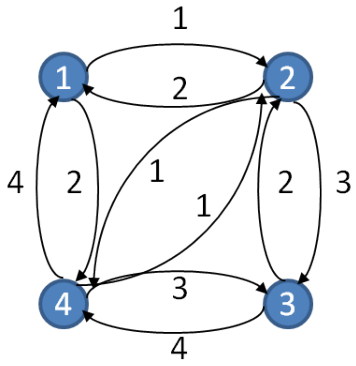


Figure 3.89: An Exemplary Weighted Directed Graph

$$A = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 2 & 0 & 3 & 1 \\ 0 & 2 & 0 & 4 \\ 4 & 1 & 3 & 0 \end{bmatrix} \quad , \quad D = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

$$L = D - \frac{A + A^T}{2} = \begin{bmatrix} 3 & -1.5 & 0 & -3 \\ -1.5 & 6 & -2.5 & -1 \\ 0 & -2.5 & 6 & -3.5 \\ -3 & -1 & -3.5 & 8 \end{bmatrix}$$

Let λ_i and μ_i be the eigenvalues of the Laplacian matrices for two graphs G and H, respectively. The formula for Spectral Distance is then as follows (Jurman, Visintainer, & Furlanello, 2011):

$$d_k(G,H) = \begin{cases} \sqrt{\dfrac{\sum_{i=N-k}^{N-1} (\lambda_i - \mu_i)^2}{\sum_{i=N-k}^{N-1} \lambda_i^2}} & \text{if } \sum_{i=N-k}^{N-1} \lambda_i^2 \le \sum_{i=N-k}^{N-1} \mu_i^2 \\ \sqrt{\dfrac{\sum_{i=N-k}^{N-1} (\lambda_i - \mu_i)^2}{\sum_{i=N-k}^{N-1} \mu_i^2}} & \text{if } \sum_{i=N-k}^{N-1} \lambda_i^2 > \sum_{i=N-k}^{N-1} \mu_i^2 \end{cases}$$

where k is chosen such that

$$\min_j \left\{ \frac{\sum_{i=1}^{k} \lambda_{ij}}{\sum_{i=1}^{n} \lambda_{ij}} > 0.9 \right\} \qquad \text{for } j = 1, 2.$$

In implementation, the top k eigenvalues that contain 90% of the energy are retained. The similarity score is between $[0, \inf)$ with values close to 0 representing two very similar graphs, while high values indicate dissimilarity (Zager & Verghese, 2008).

To augment the metric information supplied by the spectral distance and to provide a basis for comparison, additional metric characterization of graphs are being investigated. One metric that has been examined under to auspices of this work is the Graph Diameter Distance, a measure that has become popular in recent research on characterizing the properties of complex interactive networks and network analysis (Gaston, Kraetzl, & Wallis, 2006). It is defined based on the averages of the longest shortest path between all vertices. For weighted graphs, the shortest path between two nodes is the path with the smallest sum of weights for all the edges between the two nodes.

For graph $G = (V, E)$ the eccentricity of a vertex $v \in V$, denoted as $\varepsilon(v)$, is the maximum distance from v to any other vertex in the graph:

$$\varepsilon(v) = \max_{u \in V} d(v, u)$$

where $d(v, u)$ is the length of the shortest path between the vertex v and u.
There are multiple algorithms for calculating the shortest path in a weighted graph. We choose Dijkstra's algorithm in this project. This algorithm starts from a source vertex and grows a tree that will ultimately cover all the reachable vertices from the source vertex. Vertices are added to the tree in order of distance, i.e. first the vertices closest to the source vertex and so on.

Graph diameter is defined based on the eccentricities of all the vertices in the graph as:

$$D(G) = \frac{\sum_{v \in V} \varepsilon(v)}{|V|}$$

where $|V|$ is the total number of vertices in the graph G.

Graph diameter measure is used for measuring the change, or difference, between two graphs G and H as follows:

$$f(G, H) = |D(G) - D(H)|$$

In above formula, the absolute values of difference are used to measure the magnitude of the change between two graphs.

Results: The two graph similarity measures discussed above were applied to the communication networks elicited from power generation plant simulation data. The graph similarity measures discussed in the previous section were applied to the output of the topology elicitation algorithm and their efficacy in detecting changes in the operating state are examined.

Consider the WWO step test data and its extracted topologies as depicted in Figure 3.75 through Figure 3.83. The spectral distance and graph diameter distance were computed between subsequent graphs generated for the various sets of windowed data for this step test. The WWO step test data is shown in Figure 3.90 with WWO input signal denoted by a thicker line. It can be seen that the value of the input signal changes four (4) times during the course of step test. In order to evaluate the efficacy of the graph similarity measures, their ability to resolve these change points is investigated.

Figure 3.91 shows the spectral distance metric for this step test over the duration of the simulation. As mentioned before, values close to zero indicate similar graphs while higher values shows dissimilarity.

Therefore, changes in the network topology are represented as peaks in the spectral distance metric measurement. From Figure 3.91, one can see the peaks appearing in the plot of the spectral distance, but it is not clear if these peaks are related to changes in topology or simply a false positive.

Next, the graph diameter measurement is investigated using this data. The results are shown in Figure 3.92. From this figure, one can easily say that there have been four (4) instances where the dissimilarity between consequent graphs increases and then decreases. The increase in dissimilarity is happening when the input is changing and inducing a corresponding change in communication topology. This is also shown in subsequent figures where the topology changes shortly after the input change. After some time, the graph distance decreases to nearly zero. This occurs when the topology is constant for sufficient time after the input change for the elicitation algorithm to converge.

The calculations were repeated for O2Bias step test data, Figure 3.93, and produce similar results, as can be seen in the resulting plots of spectral distance and graph diameter distance, shown in Figure 3.94 and Figure 3.95, respectively. As can be seen, the graph diameter distance detects the change in the operating state of the system more accurately than spectal graph distance.

The simulation results for the rest of the step test data are shown in Figure 3.96 to Figure 3.110.

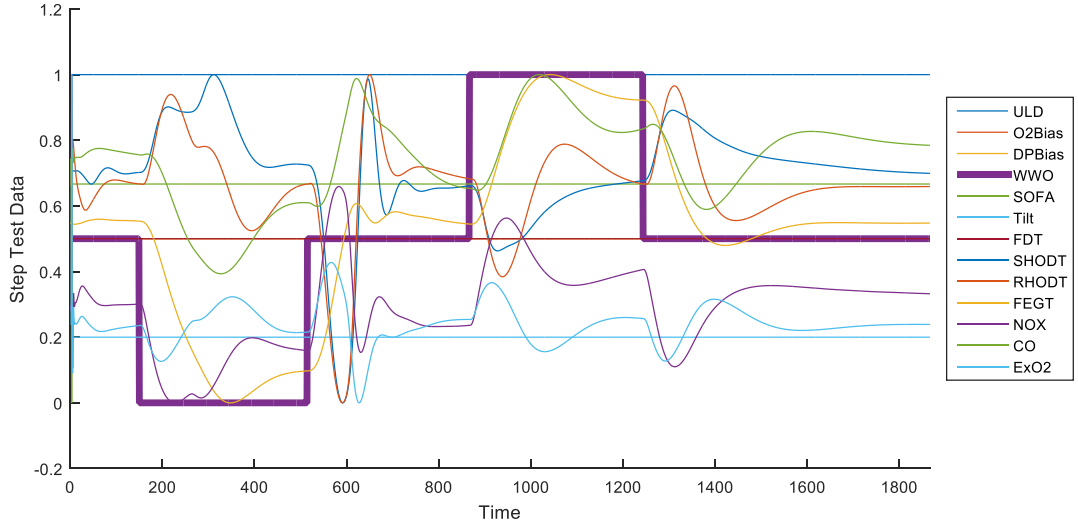


Figure 3.90: Normalized WWO Step Test Data

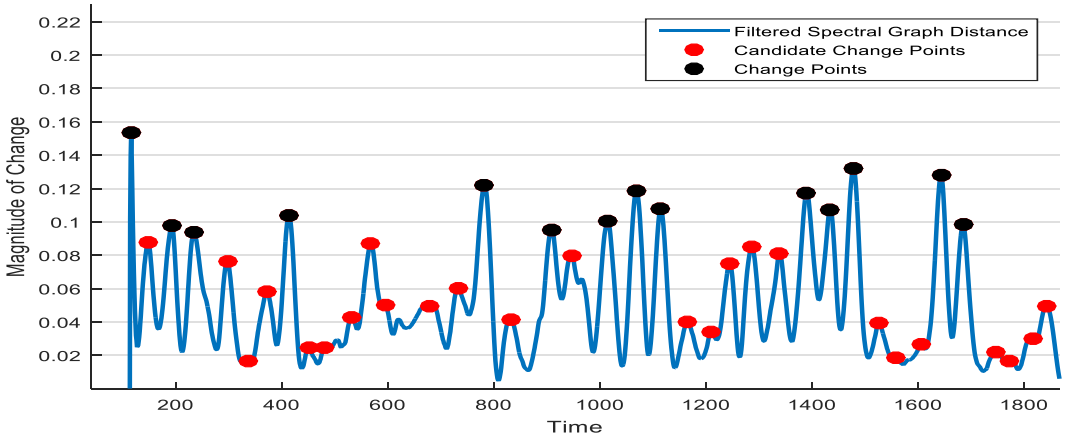


Figure 3.91: Spectral Graph Distance For WWO Step Test Data

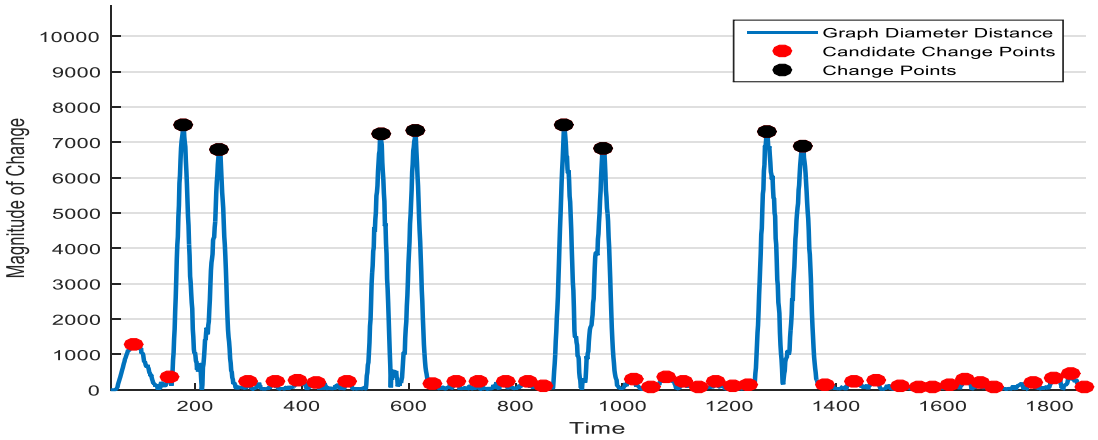


Figure 3.92: Graph Diameter Distance For WWO Step Test Data

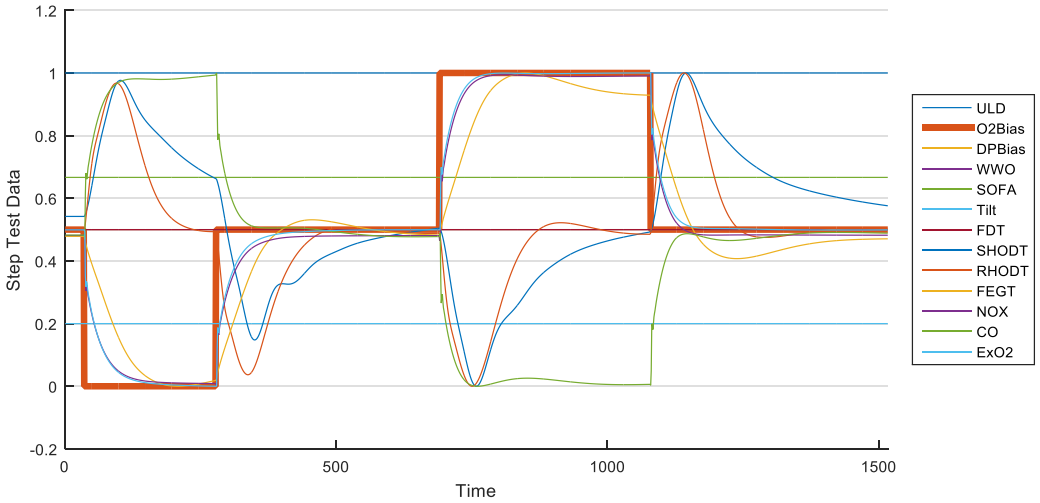


Figure 3.93: Normalized O2Bias Step Test Data

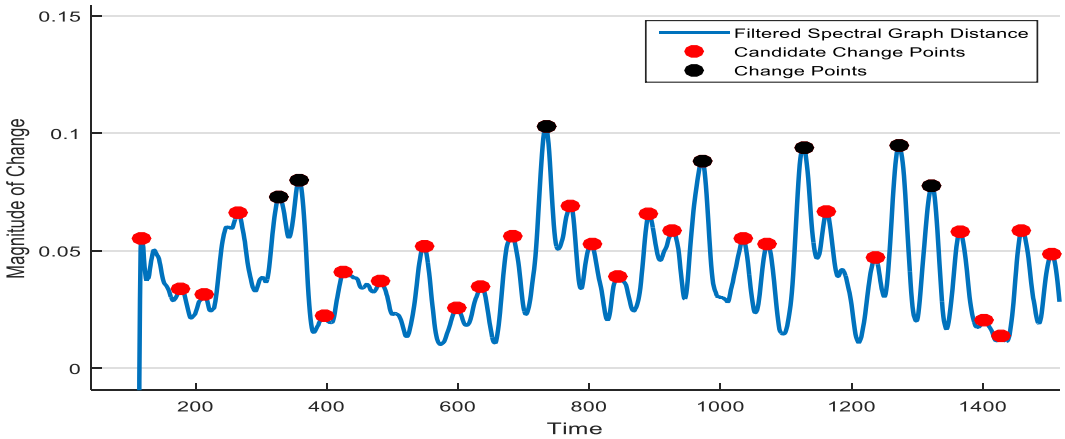


Figure 3.94: Spectral Graph Distance For O2Bias Step Test Data

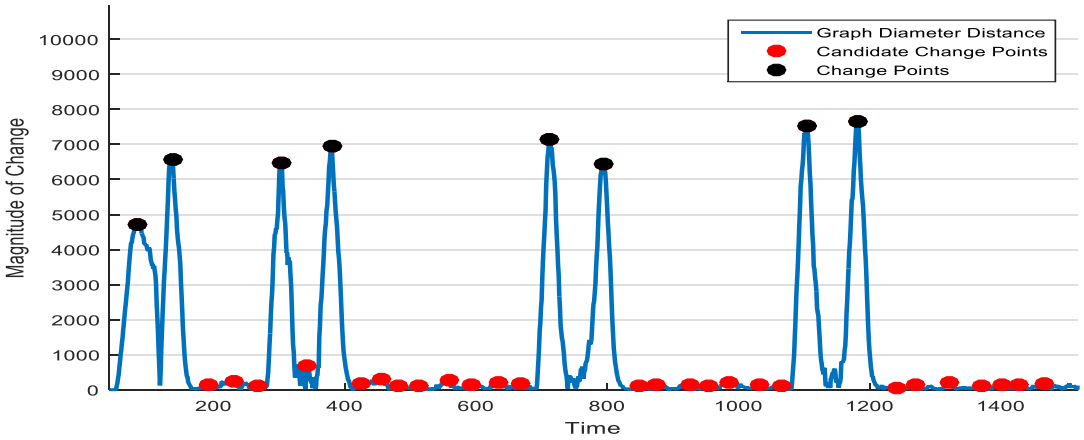


Figure 3.95: Graph Diameter Distance For O2Bias Step Test Data

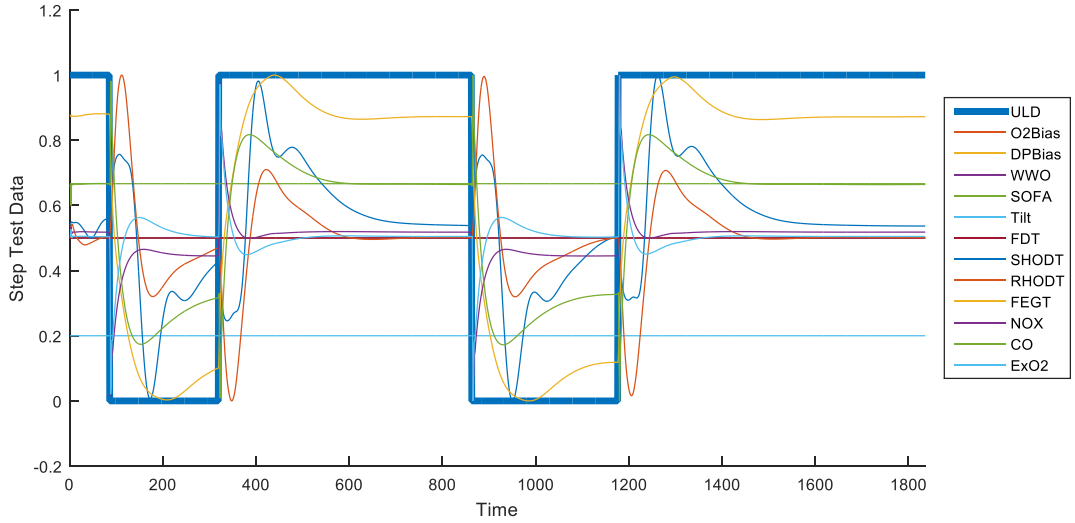


Figure 3.96: Normalized ULD Step Test Data

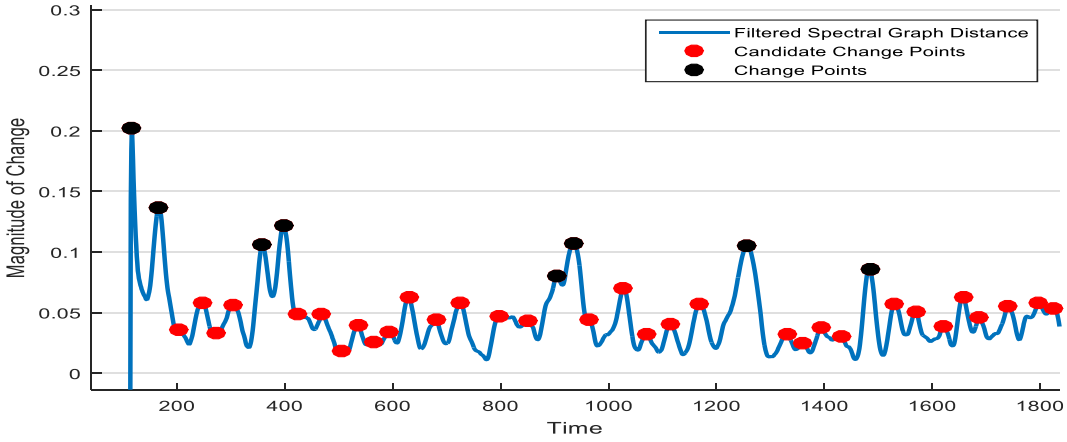


Figure 3.97: Spectral Graph Distance For ULD Step Test Data

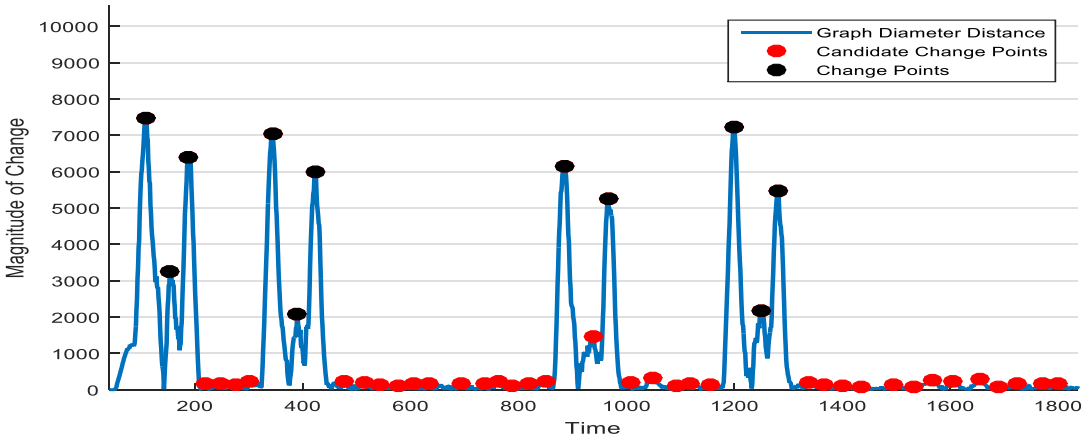


Figure 3.98: Graph Diameter Distance For ULD Step Test Data

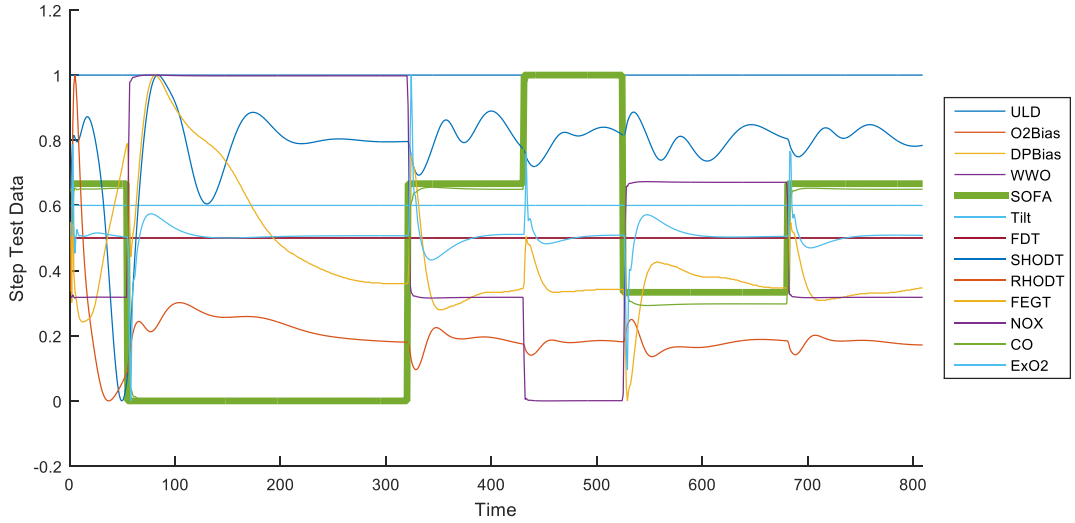


Figure 3.99: Normalized SOFA Step Test Data

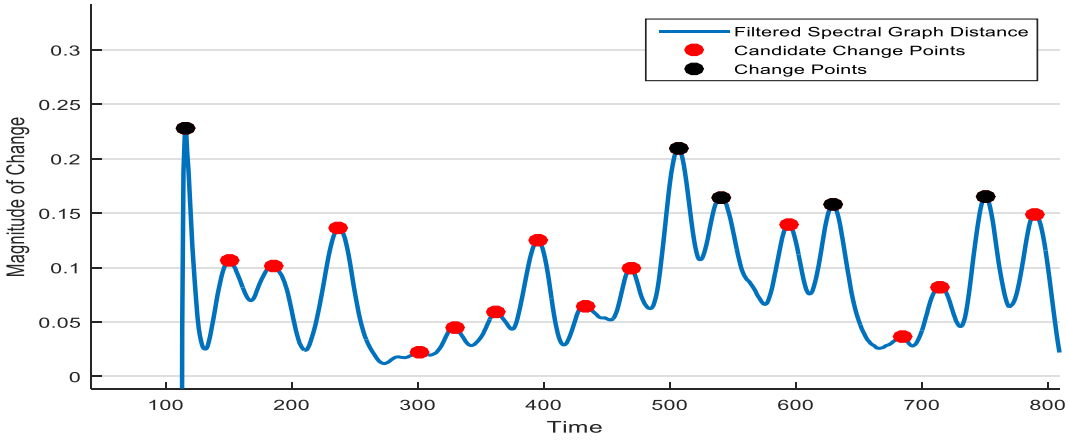


Figure 3.100: Spectral Graph Distance For SOFA Step Test Data

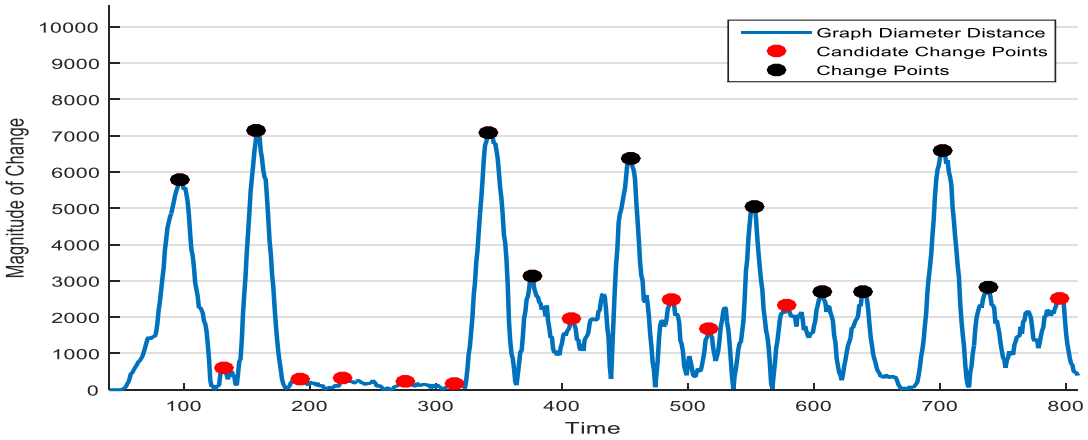


Figure 3.101: Graph Diameter Distance For SOFA Step Test Data

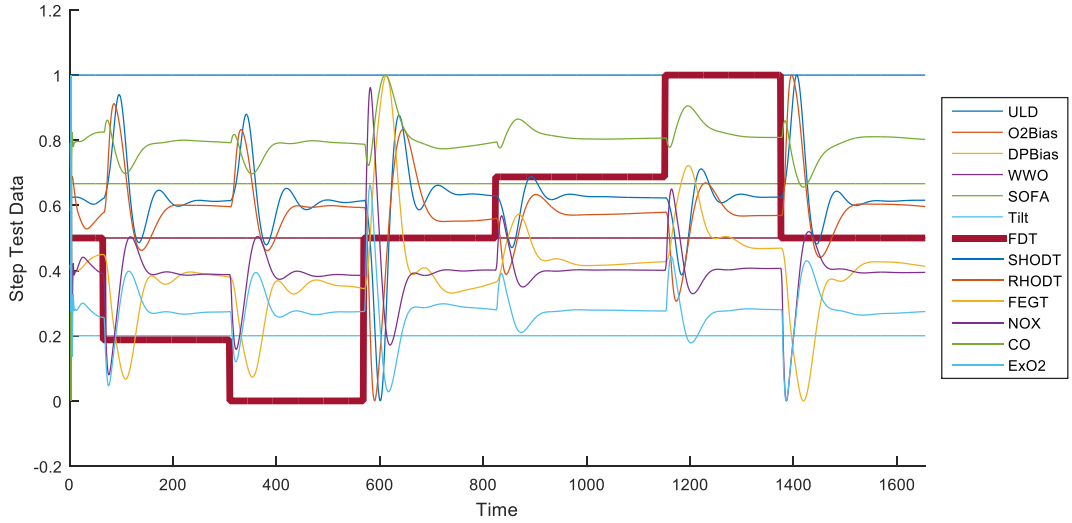


Figure 3.102: Normalized FDT Step Test Data

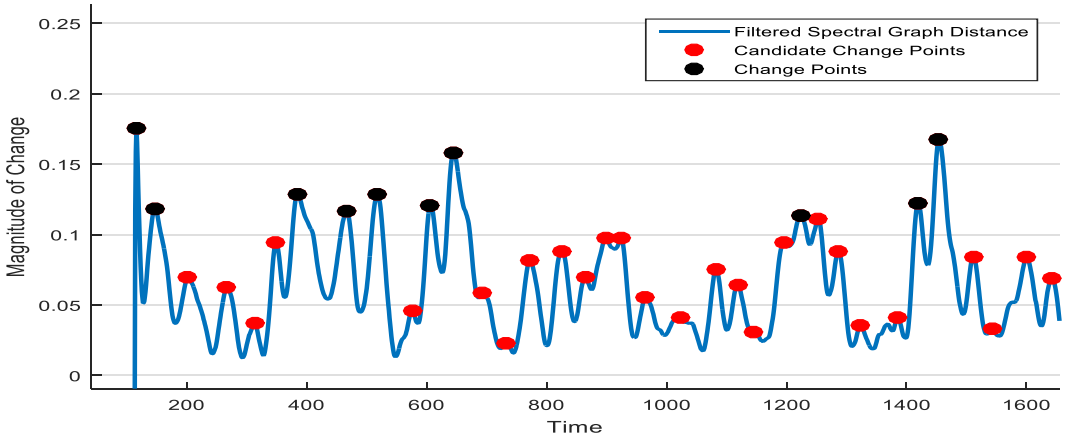


Figure 3.103: Spectral Graph Distance For FDT Step Test Data

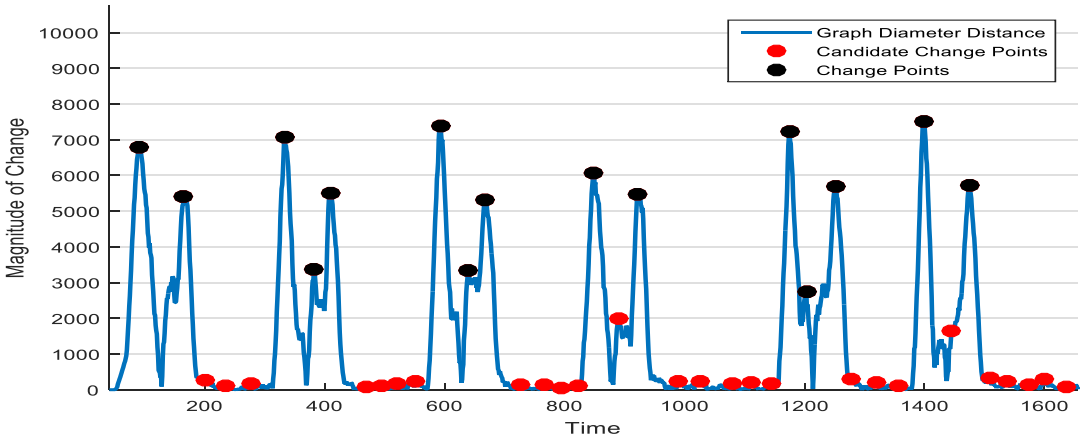


Figure 3.104: Graph Diameter Distance For FDT Step Test Data

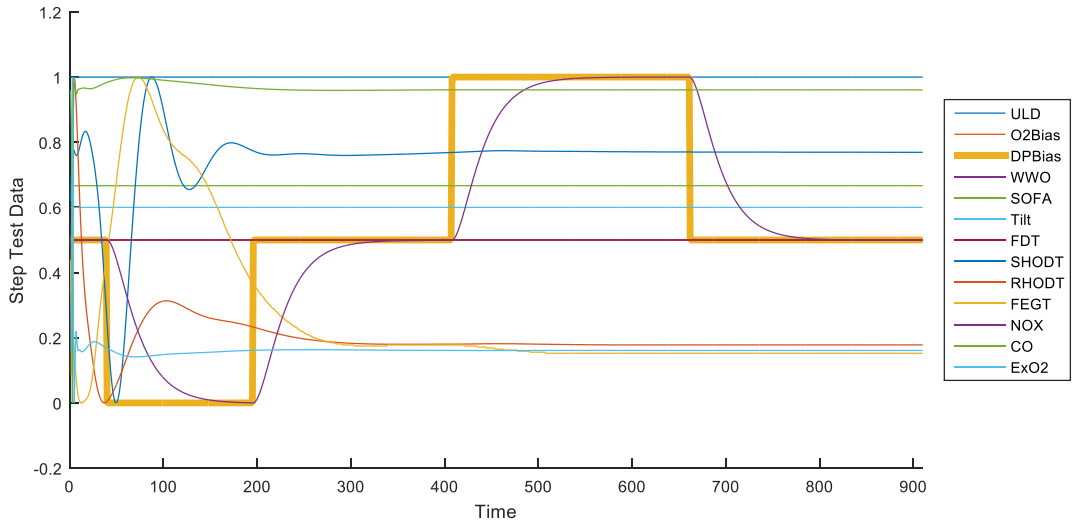


Figure 3.105: Normalized DPBias Step Test Data

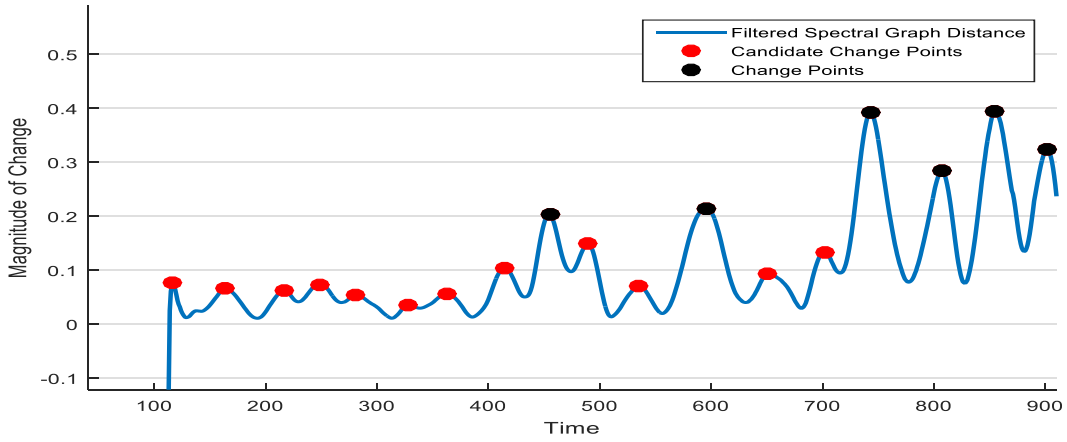


Figure 3.106: Spectral Graph Distance For DPBias Step Test Data

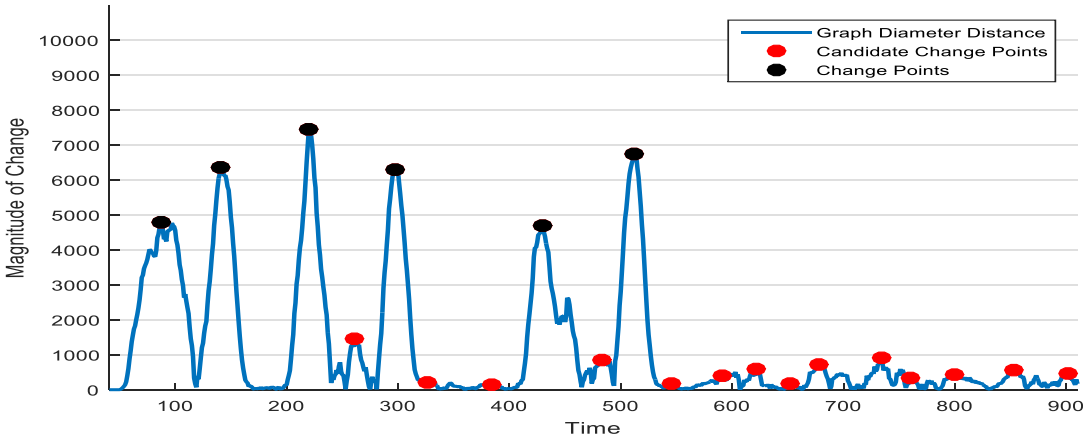


Figure 3.107: Graph Diameter Distance For DPBias Step Test Data

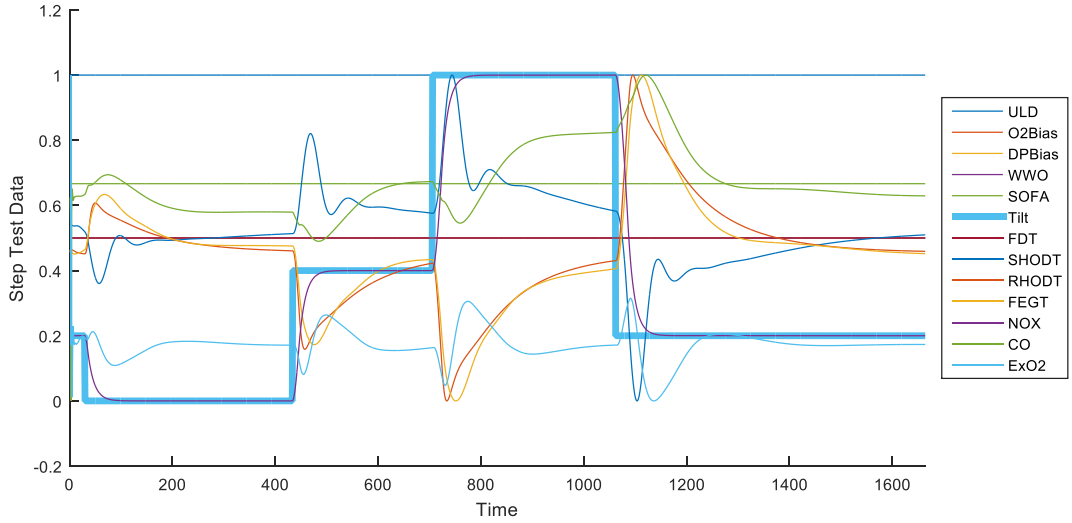


Figure 3.108: Normalized Tilt Step Test Data

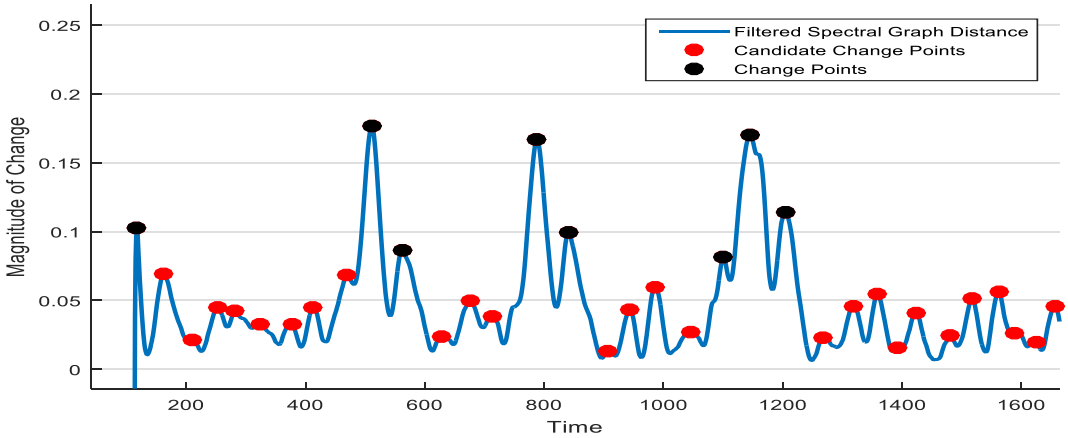


Figure 3.109: Spectral Graph Distance For Tilt Step Test Data

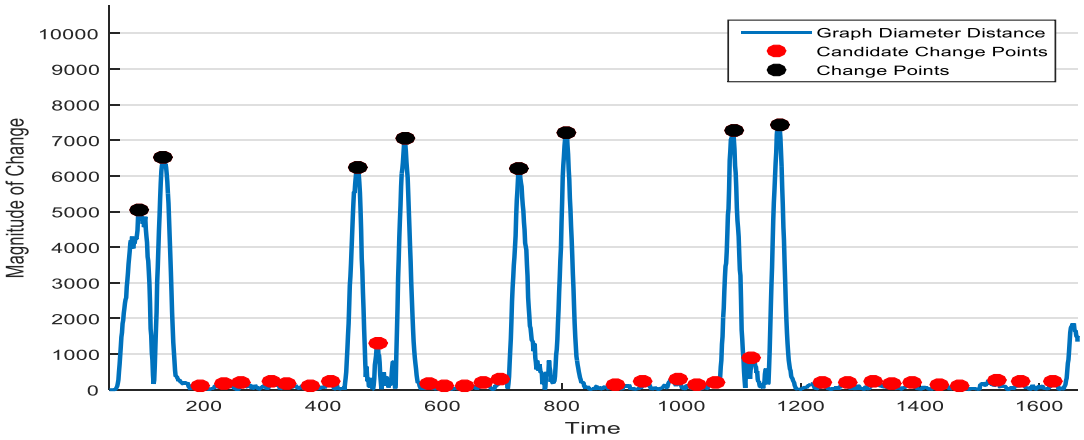


Figure 3.110: Graph Diameter Distance For Tilt Step Test Data

Next we examine step test data for the final desuperheater temperature (FDT) operating point shown in Figure 3.111.

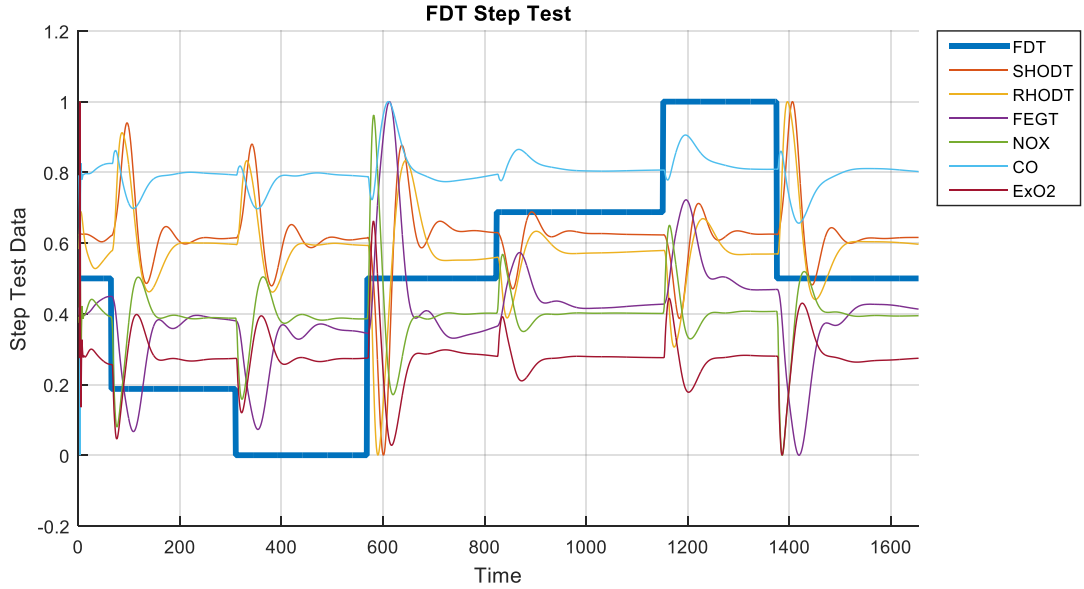


Figure 3.111: Normalized FDT Step Test Data

The objectives of this investigation are:

1. Identify the active operating point by observing measured signals.
2. Detect changes in the operating point. For example, detect when the normalized final desuperheater temperature (FDT) is changed from 0.2 to 0 as shown in Figure 3.111.

We introduce graph node similarity measures that have applications in data analysis. These measures have been successfully applied to a range of practical problems like synonym extraction (Blondel, Gajardo, Heymans, Senellart, & Van Dooren, 2004), social network analysis (Leicht, Holme, & Newman, 2006), database structure matching (Melnik, Garcia-Molina, & Rahm, 2002), etc. One of the prominent iterative approaches in identifying web-pages relevant to a query is the Kleinbergs approach, hub and authority scoring of nodes (Kleinberg, 1999). Blondel et al. generalized this approach in (Blondel, Gajardo, Heymans, Senellart, & Van Dooren, 2004). They introduced a concept for similarity measures between vertices of two

directed graphs G_A and G_B with n_A and n_B vertices, respectively. The similarity matrix S is defined such that s_{ij} represents the similarity score between vertex j in G_A to vertex i in G_B. Let A and B be the adjacency matrices of G_A and G_B, respectively. The similarity matrix S is then calculated as follows:

1. Set $Z_0 = I$.
2. Iterate an even number of times
3. $$Z_{k+1} = \frac{BZ_kA^T + B^TZ_kA}{\left\| BZ_kA^T + B^TZ_kA \right\|_F}$$
   a. and stop upon convergence.
4. The similarity matrix S is the last value of Z_k.

The matrix norm $\|.\|_F$ is the Euclidean or Frobenius norm; the square root of the sum of all squared entries. The similarity matrix between G_B and G_A is the transpose of the similarity matrix between G_A and G_B. When $G_A = G_B = G$, the similarity matrix S is a square matrix whose elements are the similarity score between vertices of graph G. This matrix is called the *self-similarity matrix* of G.

Consider the weighted directed graphs given in Figure 3.112 where graphs H_1 and H_2 are similar to graph G except for some changes on their weights. In H_1, the weights on all the edges connected to vertex 3 are reduced, while in H_2 only the edges connected to vertex 1 have nonzero weights.

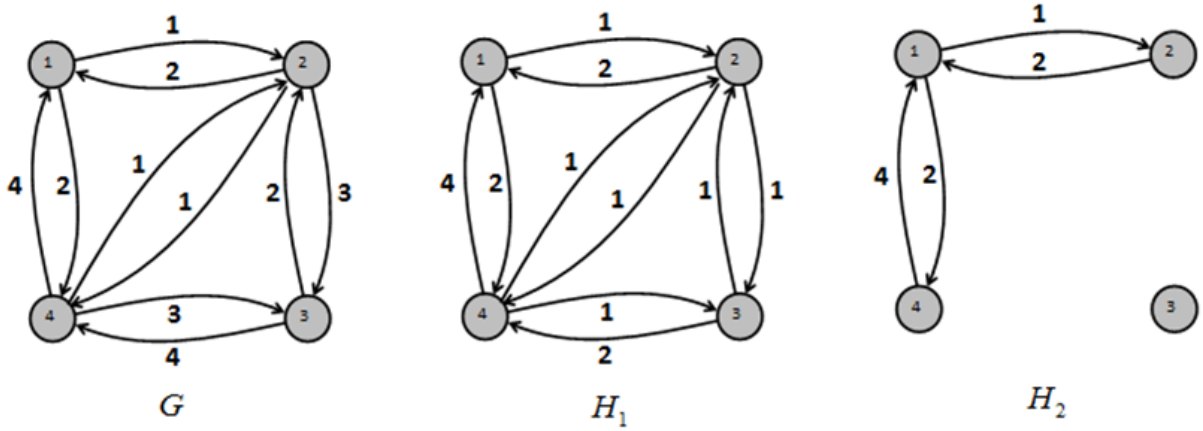


Figure 3.112: Exemplary Weighted Directed Graphs

The similarity matrices between these graphs are given below and we can see that the elements of the third row of the similarity matrix S_{G,H_2} are all zero. The reason is that vertex 3 of graph H_2 is isolated from the graph and is not similar to any vertices of graph G.

$$S_{G,H_1} = \begin{bmatrix} 0.2784 & 0.2252 & 0.3230 & 0.3281 \\ 0.1620 & 0.1892 & 0.2227 & 0.2576 \\ 0.1253 & 0.1372 & 0.1776 & 0.1817 \\ 0.2309 & 0.2809 & 0.3174 & 0.3853 \end{bmatrix}$$

$$S_{G,H_2} = \begin{bmatrix} 0.3195 & 0.2125 & 0.3702 & 0.2910 \\ 0.1040 & 0.1773 & 0.1535 & 0.2458 \\ 0 & 0 & 0 & 0 \\ 0.2080 & 0.3545 & 0.3069 & 0.4916 \end{bmatrix}$$

Consider FDT step test data and its extracted topologies in Figure 3.113 to Figure 3.115. We calculate node similarity between consequent graphs for this step test. As shown in these figures, the value of the FDT input signal changes 6 times during the course of the simulation. We would like to test the ability of graph similarity measures in detecting these 6 change points.

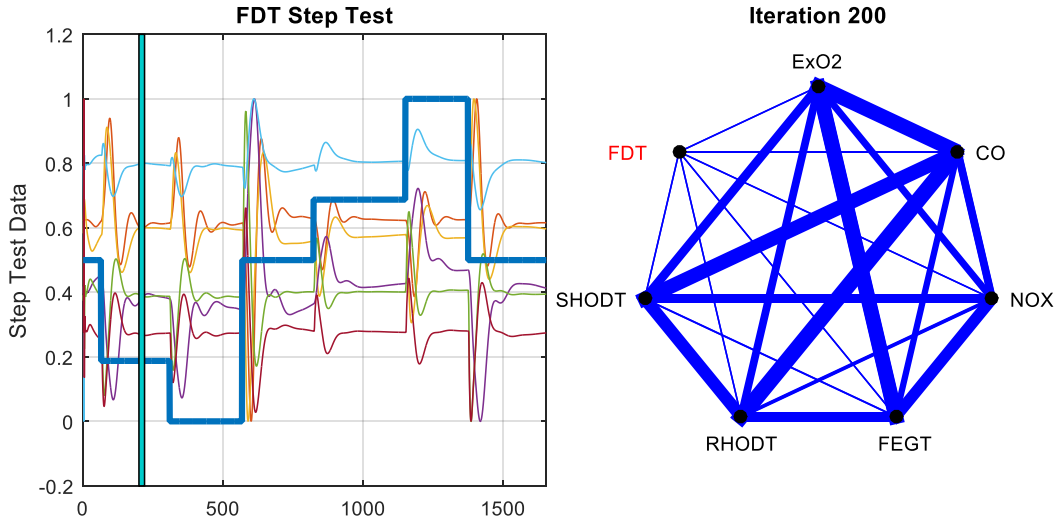


Figure 3.113: Connectivity Structure For FDT Step Test Data With FDT Signal at 0.2

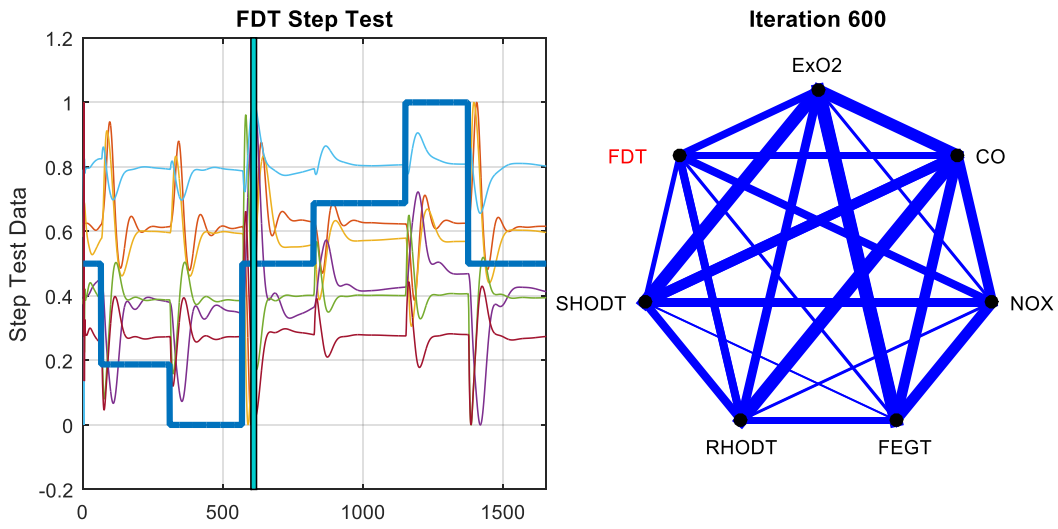


Figure 3.114: Connectivity Structure For FDT Step Test Data With FDT Signal at 0.5

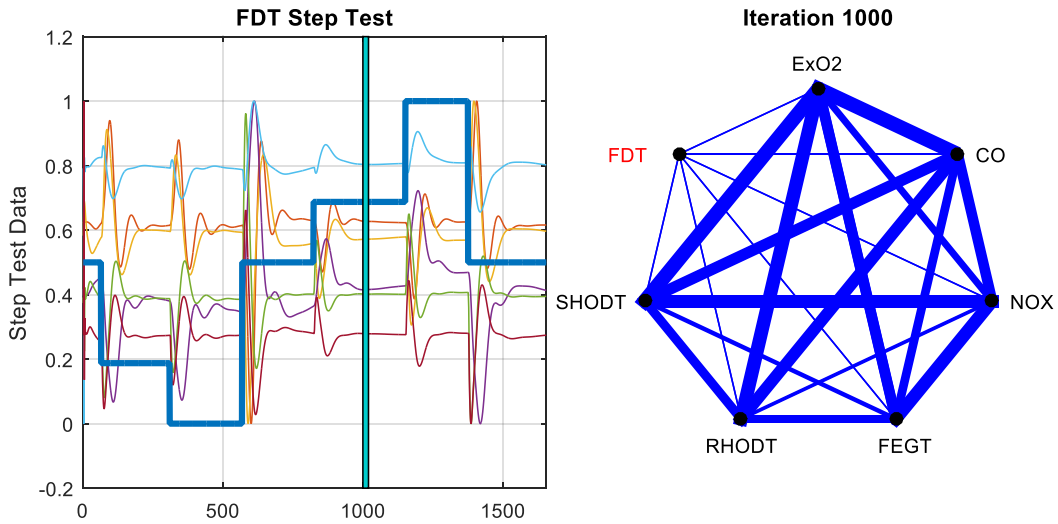


Figure 3.115: Connectivity Structure For FDT Step Test Data With FDT Signal at 0.7

Figure 3.116 shows normalized FDT step test data along with node similarity measure for change detection. Node similarity measure shows the changes in the similarity score for each node. This score is a function of the weights of the edges connected to this node and changes of its neighboring nodes. If this value does not change much, it indicates that the graph topology around this node has not changed much either. On the other hand, a sudden change in this value, increase or decrease, is indicative of a change in the graph connectivity around this node.

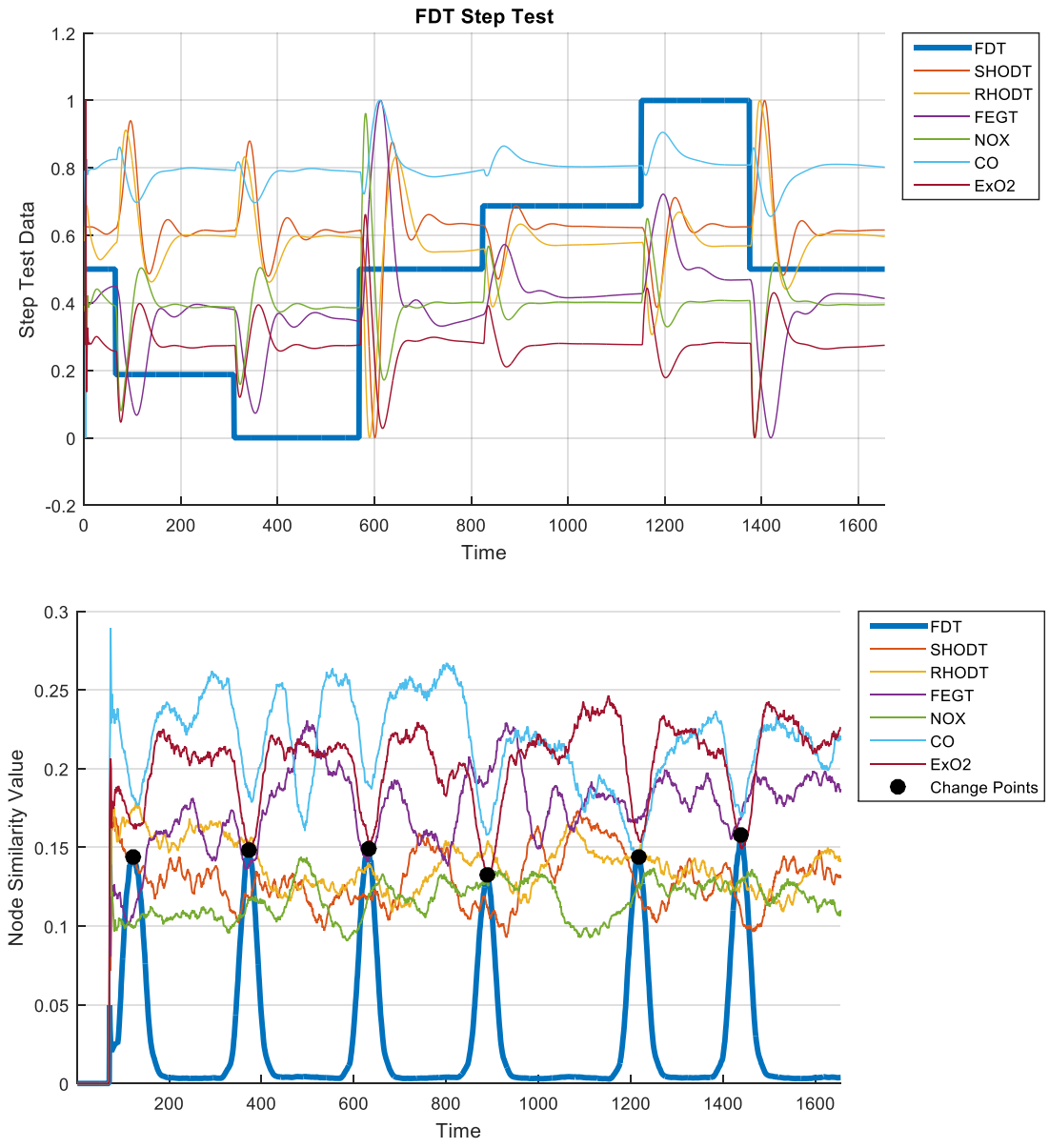


Figure 3.116: Change Detection For Normalized FDT Step Test Data

From Figure 3.116, we can see that the node similarity value for output nodes does not change much. However, this value for the input node, FDT, changes dramatically in 6 occasions. These are the points where the change is happening in the input signal. The results for the rest of the operating points are shown in Figure 3.117 to Figure 3.122. We can conclude from these figures that node similarity measure technique could correctly detect all the changes in the active operating point.

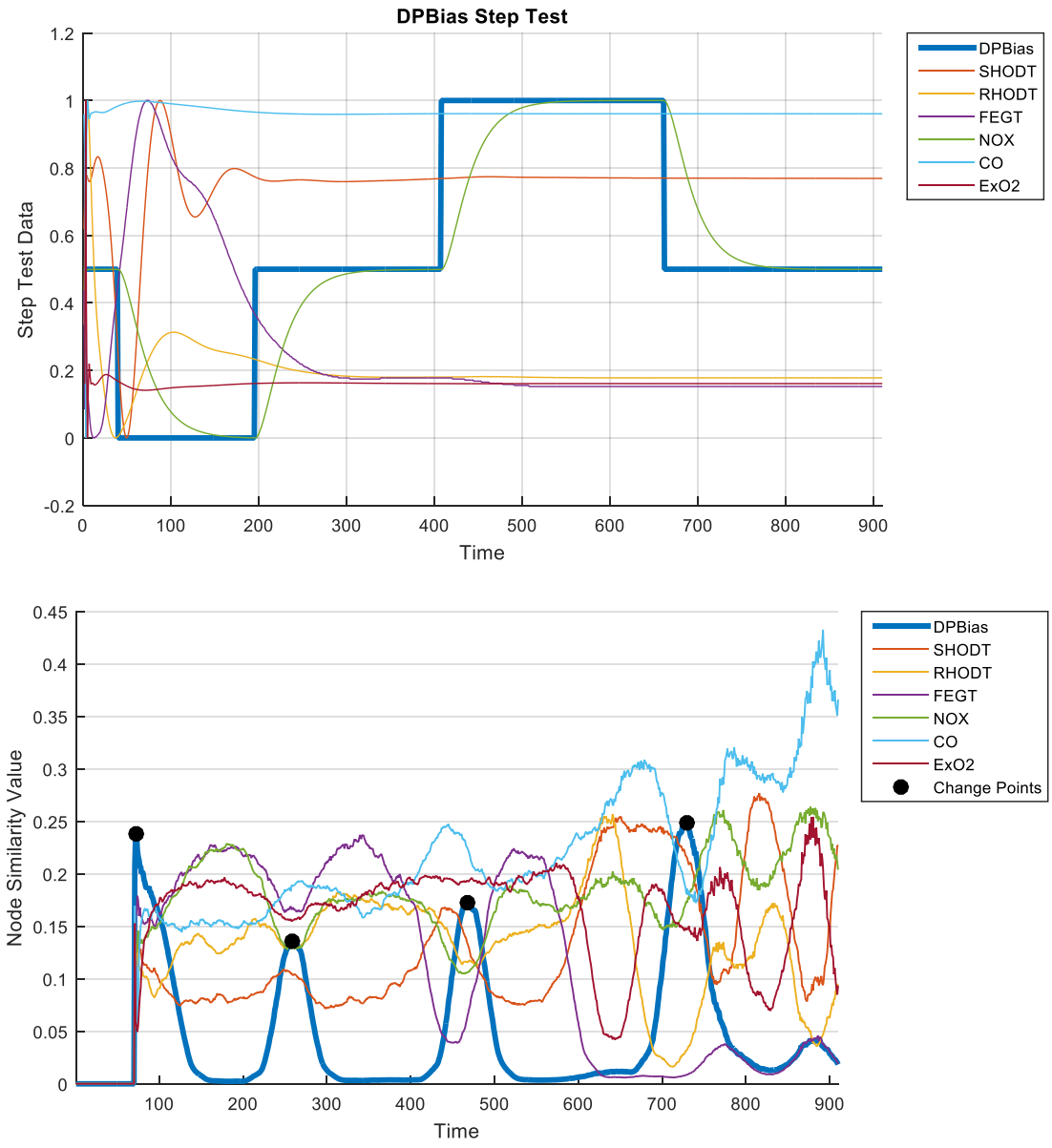


Figure 3.117: Change Detection For Normalized DPBias Step Test Data

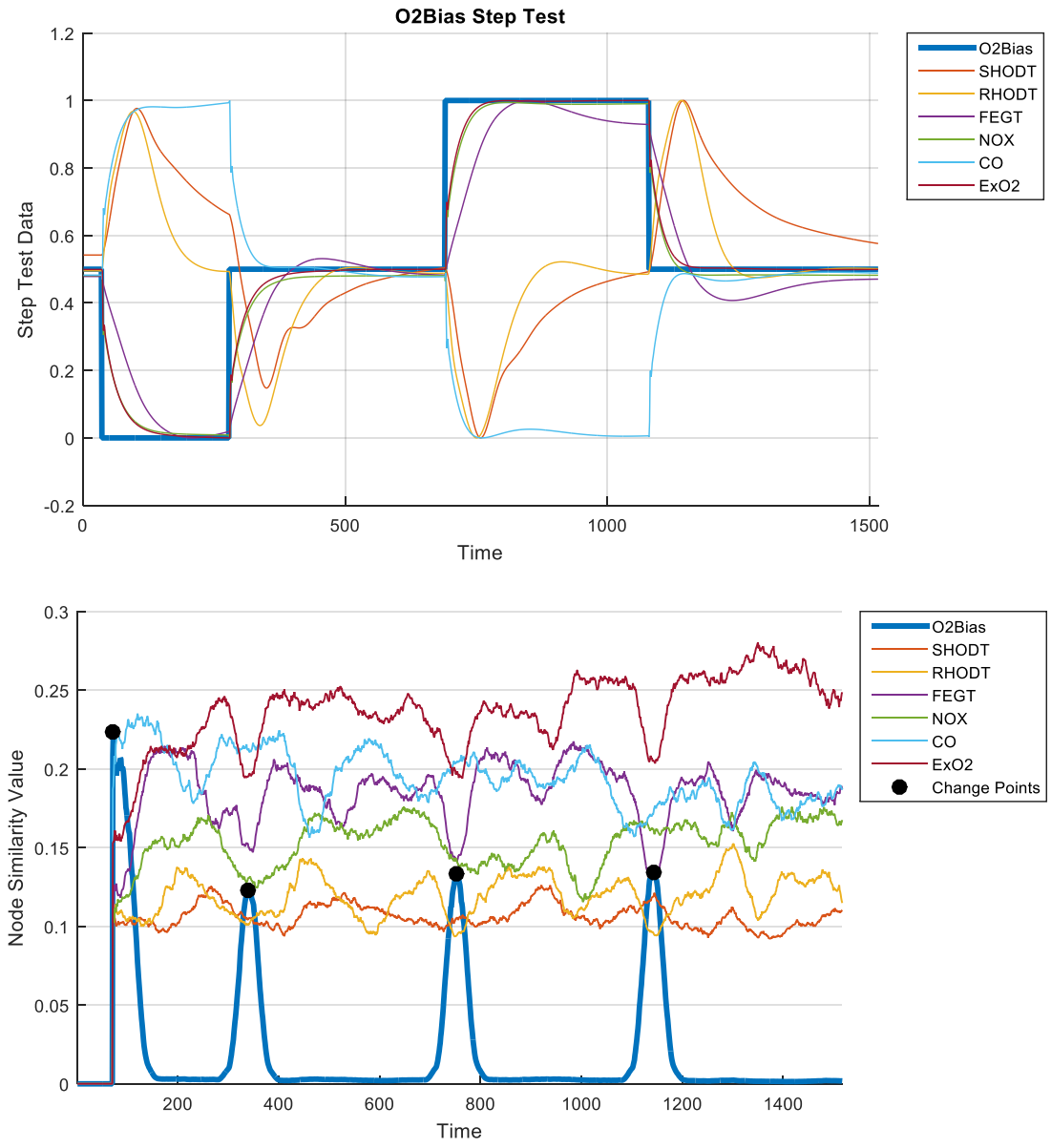


Figure 3.118: Change Detection For Normalized O2Bias Step Test Data

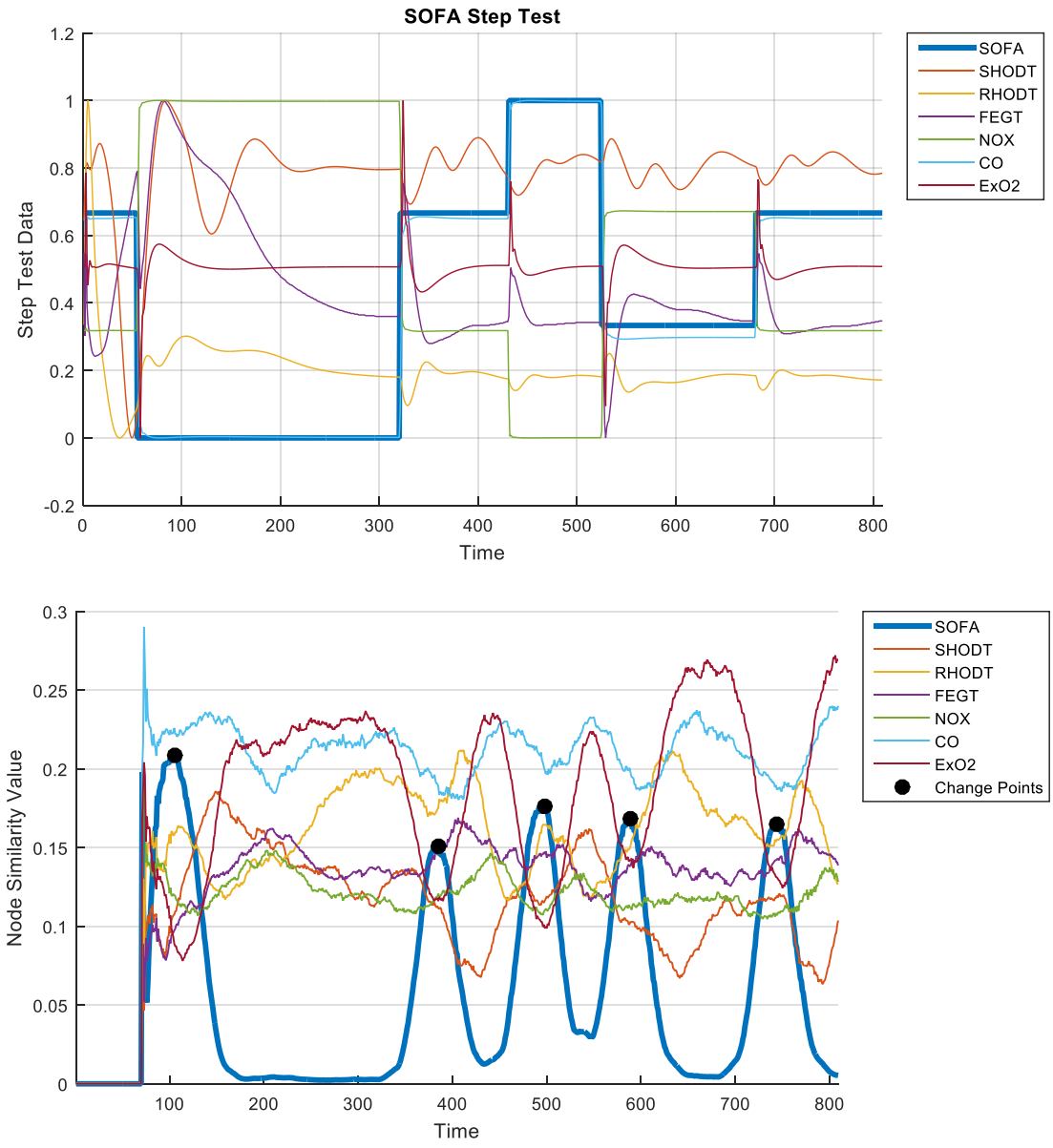


Figure 3.119: Change Detection For Normalized SOFA Step Test Data

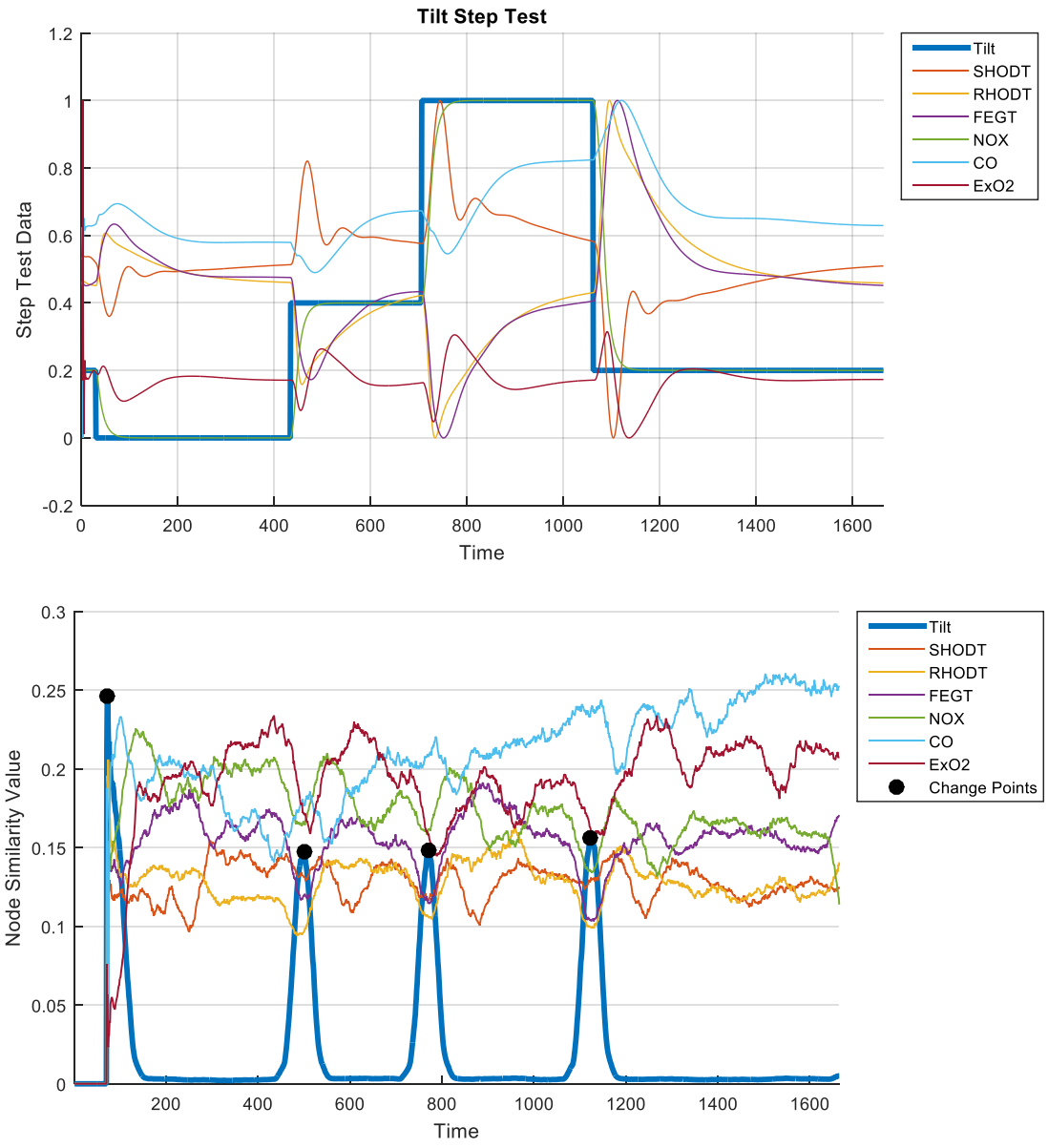


Figure 3.120: Change Detection For Normalized Tilt Step Test Data

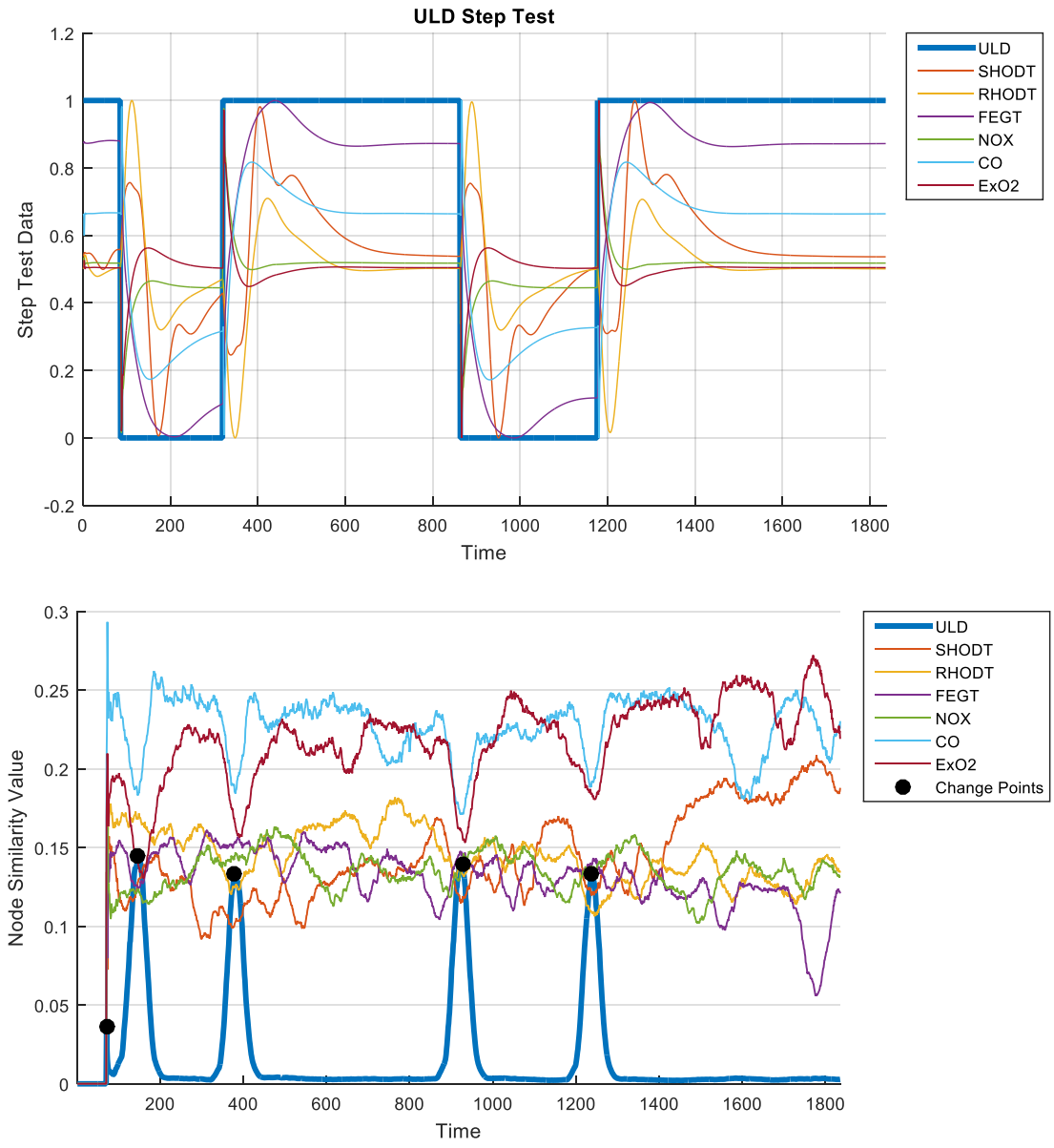


Figure 3.121: Change Detection For Normalized ULD Step Test Data

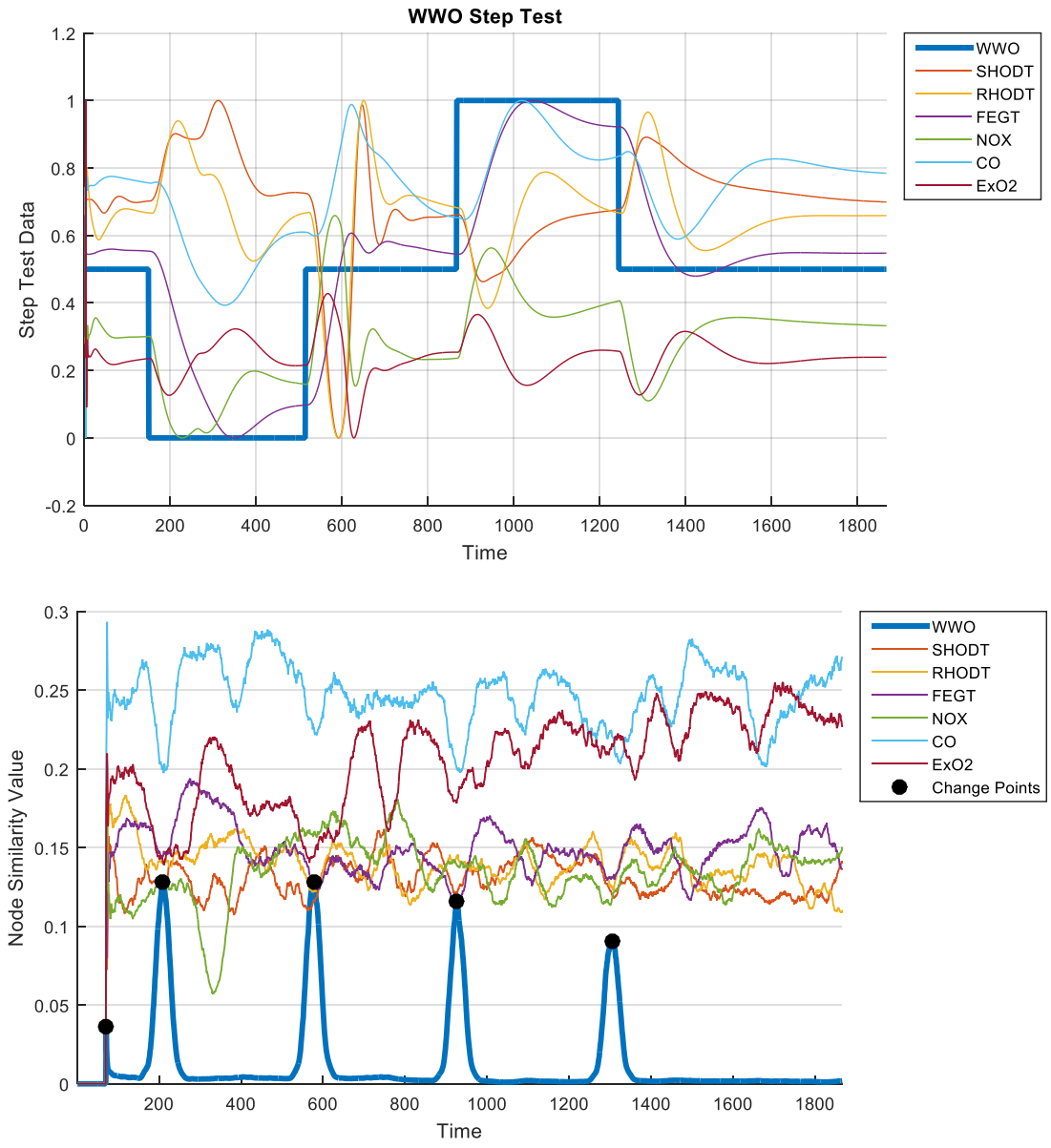
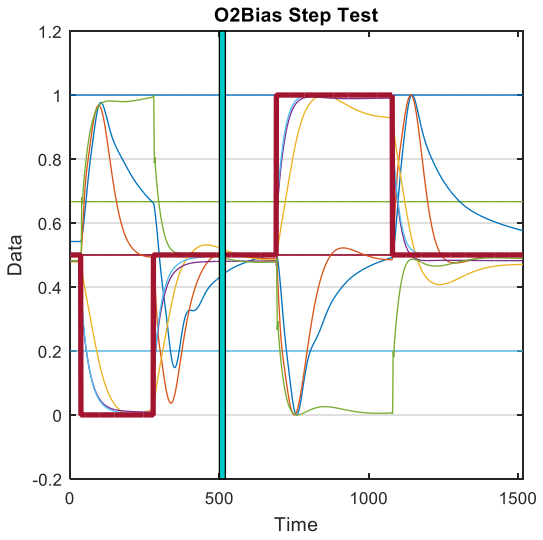
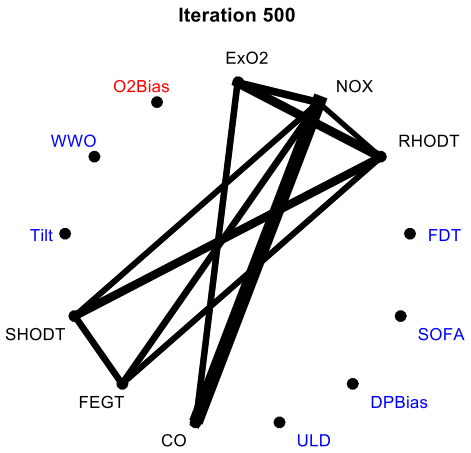
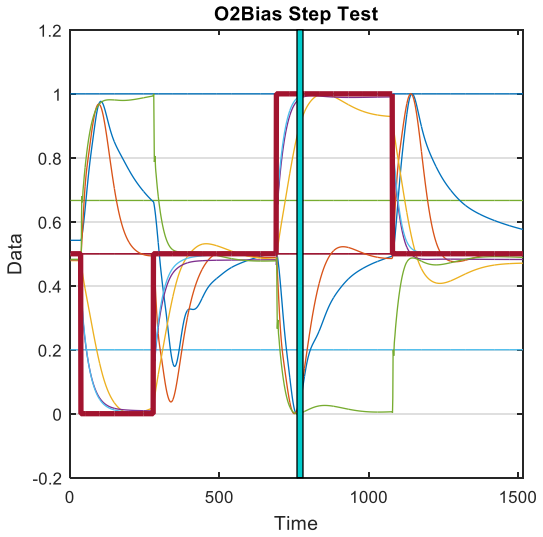


Figure 3.122: Change Detection For Normalized WWO Step Test Data

Next, we evaluate the performance of our proposed algorithm, along with the change detection technique using the node similarity measure, in detecting the active input signal and its change points. In this case, we consider a network consisting of 13 nodes, 7 input nodes and 6 output nodes. The step test data are applied to the network without knowing exactly which step test is selected. The intrinsic communication topologies are then extracted from the network at different time windows and the results are passed through the change detection algorithm to detect the active input and its change points. Figure 3.123 and Figure 3.124 show the information connectivity structures for two different time windows of O2Bias and FDT step test data, respectively. In the connectivity structure on the right side of these figures, input and output nodes are shown in blue and black. The thickness of the line between each two nodes in the discovered topology is proportional to the communication strength between those two nodes. As there are 13 nodes in this network and 156 possible connections between them, only strong connections are shown in these figures. Note that the red line represents the connectivity between input and output nodes.







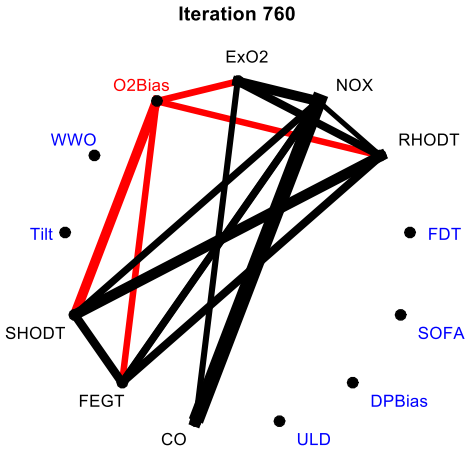
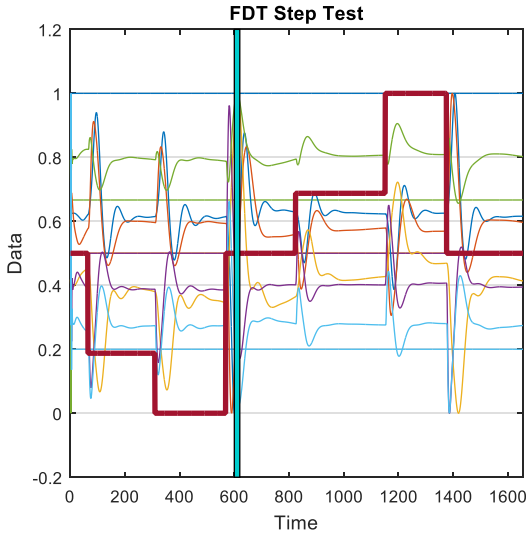
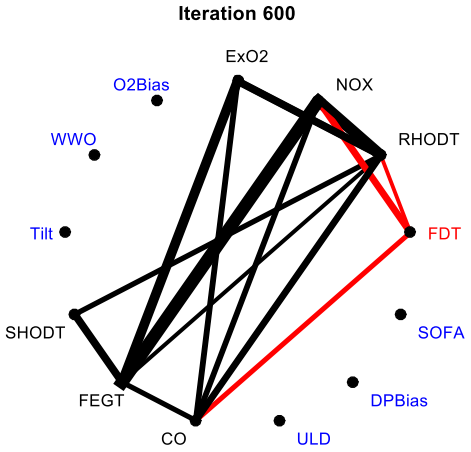


Figure 3.123: Connectivity Structure for O2Bias Step Test Data at Different Times





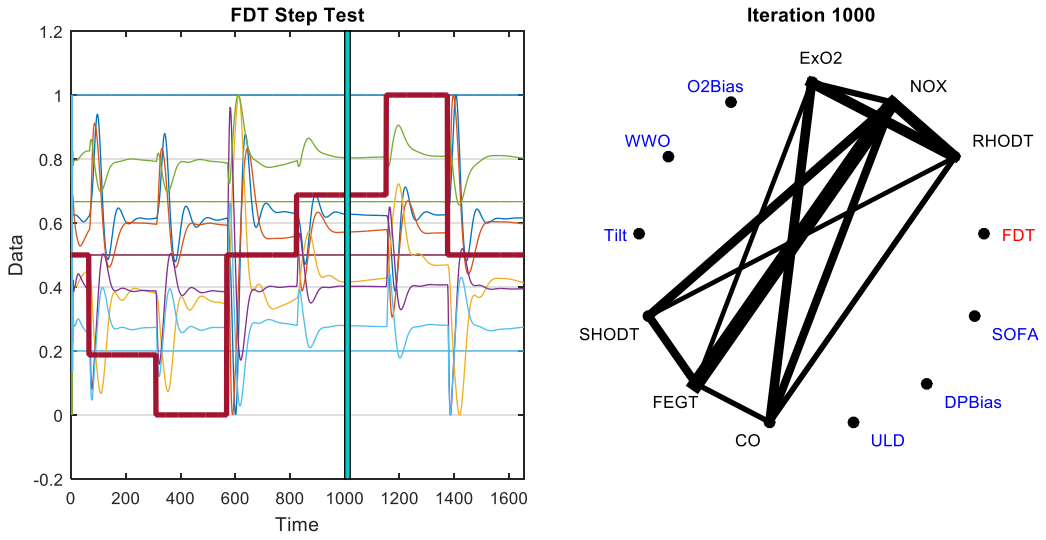


Figure 3.124: Connectivity Structure for FDT Step Test Data at Different Times

We can see that the red lines only appear after a change in the status of the operating point. For example, in Figure 3.124, these lines appear at iteration number 600 that is a little after the FDT signal has changed from 0 to 0.5. Therefore, by looking at the node similarity measure, we can detect the changes in the operating point

.

Now, let us take a look at the node similarity measure for O2Bias and FDT step test data for both input and output nodes in Figure 3.125 and Figure 3.126. We can see that the similarity score for output nodes are changing between 0.05 and 0.3 smoothly. Input node similarities, on the other hand, have a different pattern. They are all close to zero, except for one node where its similarity score changes radically at some points during the step test. These changes are rather big comparing to the changes of output nodes similarity values. By looking at these figures, one can see that the peaks are around the time where input signal transits from one level to another. This pattern is observed for all other step test data. Therefore, in order to detect the active input node, we can simply take a look at the node similarities of input nodes. The active input node is the node with some big peaks. After detecting the active input, we can simply extract change points by detecting peaks of the node similarity value for that input node as shown in Figure 3.127.

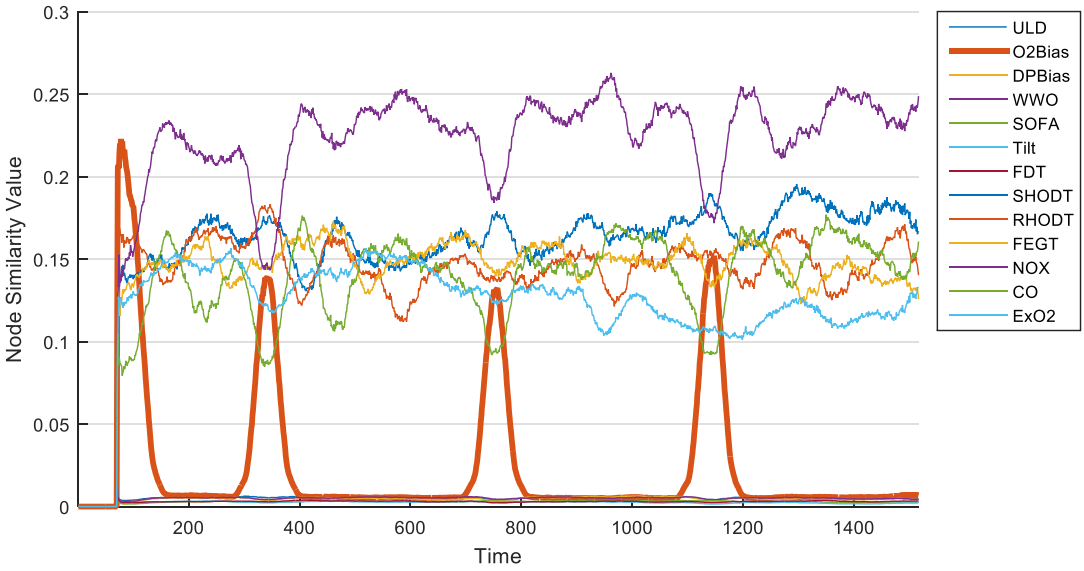


Figure 3.125: Node Similarity Measure For O2Bias Step Test Data

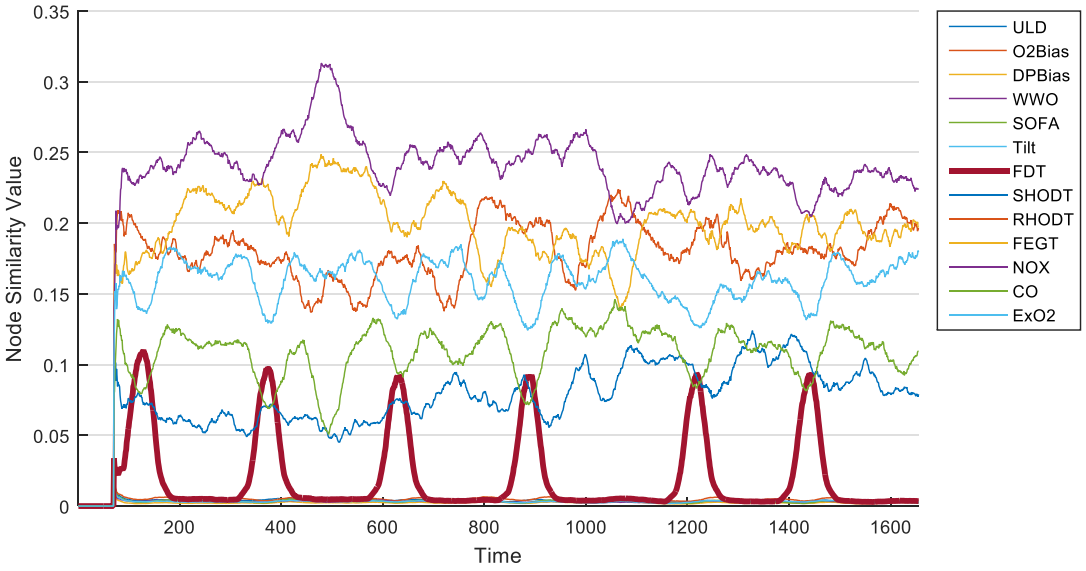


Figure 3.126: Node Similarity Measure For FDT Step Test Data

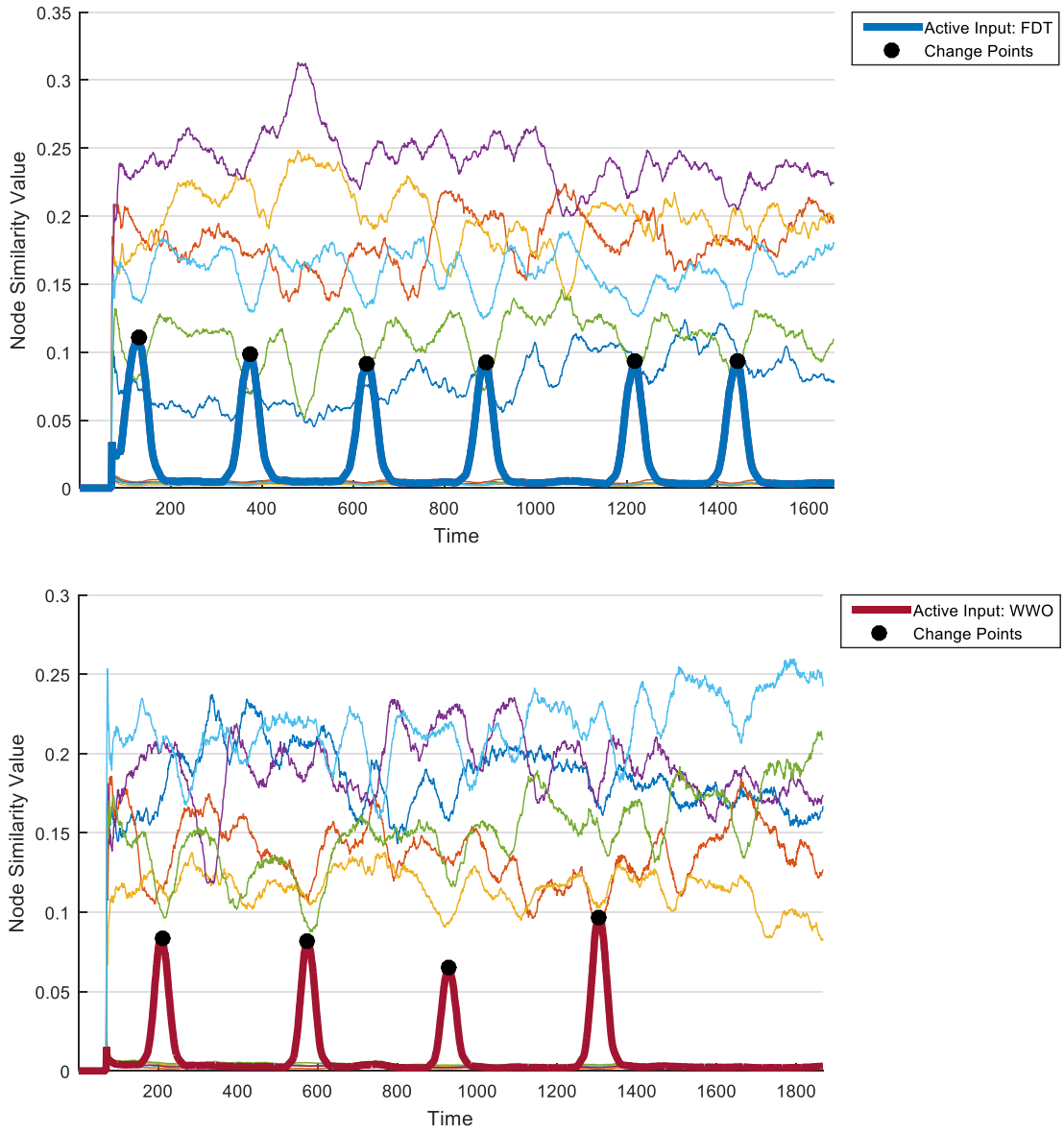


Figure 3.127: Active Input And Change Points Detection For Two Step Test Data

Table 3.**12** compares the original change points with the detected change points for all step test data. We can see that the algorithm detects the change points with a delay of around 50 iterations. This is the size of the sliding window which agents stay at a node, record data at that node and then grab the recorded data and take it to their next node. We can conclude that our proposed algorithm works great in detecting the active operating point, extracting the information connectivity structure of power generation plant for different operating point values and detecting the change points of the operating point.

Table 3.12: Change Point Detection Comparison

Step Test	Original Change Points	Detected Change Points	Difference
O2Bias	35 278 690	78 344 757	43 66 67

	1079	1148	69
DPBias	39 195 407 661	74 361 468 _	35 166 61 _
FDT	64 309 568 823 1151 1375	129 374 629 893 1218 1442	66 65 61 70 67 67
SOFA	54 320 430 524 680	98 379 489 582 739	44 59 59 58 59
Tilt	29 433 705 1061	75 498 773 1126	46 65 68 65
ULD	85 319 862 1176	145 384 929 1245	60 65 67 69
WWO	149 514 866 1243	210 575 929 1306	61 61 63 63
		Average	**52.27**

The proposed algorithm uses ant foraging behavior concepts to discover the information connectivity between elements of a system. We have reported on the performance using Gaussian exemplary system data and power plant simulation data with correlation coefficient calculations to discover the intrinsic communication topology. We applied graph similarity measures and introduced three change point detection techniques, spectral graph distance, graph diameter distance and node similarity measure, for detecting the changes in the system structure based on the discovered topologies from our algorithm. Next we conclude this section by considering a Non-Gaussian exemplary system and apply mutual information calculations to extract the system topology and detect the changes in the system.

Our proposed algorithm uses information measures for discovering the intrinsic communication topology of dynamic systems. These intrinsic communications between the elements of the system manifest as the 'mutual information' between data measured at these elements and can be used for extracting the system's intrinsic communication topology.

Mutual information measures the mutual dependency between two random variables. Let X and Y be two discrete random variables, the mutual information is then defined as:

$$I(\mathrm{X};Y) = \sum_{y \in Y} \sum_{x \in X} p(x,\mathrm{y}) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

where $p(x,y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y, respectively. Mutual information measures how much information two random variables share. It captures the statistical relationship between random variables even for nonlinearly correlated variables. This is unlike the statistical measures such as Correlation Coefficient, which captures the linear relationship between random variables. For random variables X and Y with respected expected values of μ_X and μ_Y and standard deviations of σ_X and σ_Y, correlation coefficient is quantified as:

$$\rho_{X,Y} = \mathrm{corr}(X,\mathrm{Y}) = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E\left[(\mathrm{X}-\mu_X)(Y-\mu_Y)\right]}{\sigma_X \sigma_Y}$$

Mutual information calculates the statistical relationship between both linearly and nonlinearly correlated random variables. In large systems, mutual information calculations can be complicated and time consuming. On the other hand, correlation coefficient captures the linear relationship between random variables and is computationally less complex than mutual information. In previous reports, we considered Gaussian exemplary systems with correlation coefficient calculations. In this report, we consider a Non-Gaussian exemplary system with mutual information calculations.

Consider a Non-Gaussian distribution and use mutual information calculations for extracting the connectivity structure of the network. The exemplary system considered here is shown in Figure 3.128. The Non-Gaussian distribution is generated from a Gaussian distribution that has been passed through a nonlinear function.

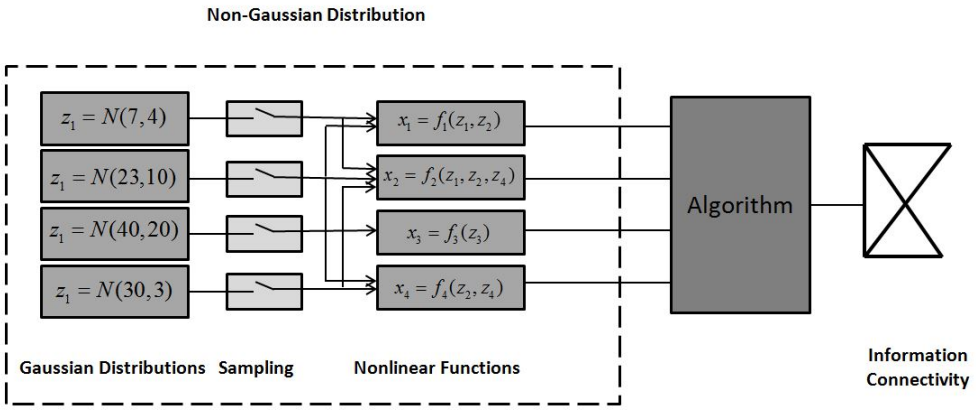


Figure 3.128 - Non-Gaussian Exemplary System Representation

For simulation, 1000 data points are sampled from the Gaussian distributions z_i and passed through the nonlinear function f_i. The proposed algorithm is then applied to the Non-Gaussian distribution and the connectivity structure is extracted. In order to evaluate the performance of our algorithm using mutual information calculations, we calculate the actual mutual information between all the nodes.

The mutual information matrix for the Gaussian distribution x, is a square matrix I_X whose elements are the mutual information between network nodes. For the Non-Gaussian distribution x of Figure 3.128, the mutual information matrix is given below, a we expect stronger connectivity between the pairs < 2, 4 >, < 1, 2 >, and < 1, 4 >.

$$I_X = \begin{bmatrix} 1 & 0.082 & 0.029 & 0.069 \\ 0.082 & 1 & 0.024 & 0.135 \\ 0.029 & 0.024 & 1 & 0.026 \\ 0.069 & 0.135 & 0.026 & 1 \end{bmatrix}$$

The simulation is initiated with 6 agents with lifespan of 181 at each node for a total of 24 agents over the entire network. The agents traverse the network and discover the intrinsic communication topology of the network. At the end of the simulation, there are approximately 4500 agents traveling throughout the network. The simulation is run with no initial pheromone present on the ground and a pheromone decay rate of 0.2 units/iteration. A sliding window with a length of 100 is passed over the data stream and the intrinsic communication topology for each window of data is discovered. Two nodes are considered correlated if their mutual information value is higher than 0.1. The final extracted intrinsic communication topology is shown in Figure 3.129. This topology supports the actual mutual information matrix.

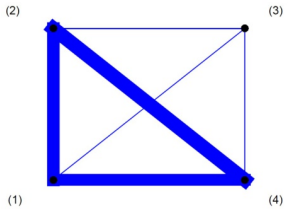


Figure 3.129: Final Topology

We repeat the simulation with different Non-Gaussian distributions. Our proposed algorithm performs well in extracting the intrinsic communication topology. However, the threshold for desired mutual information value should be selected carefully. If the threshold is high, the agents traversing the network remain forward agents and deposit small amount of pheromone on their path. This leads to a fully connected topology as the extracted intrinsic communication structure. While smaller thresholds give us better estimation of the system connectivity structure, they are time consuming and computationally inefficient for larger systems. For the same network with the same simulation specifications, simulation time for mutual information calculation is 106.16 minutes. However, correlation coefficient calculation takes only 29.64 minutes to extract the communication topology.

Next we consider how this can be applied to condition monitoring, discuss n more detail in Section 5.0, using a system containing a switch as shown in Figure 3.130. We examine the performance of our algorithm using mutual information calculation to detect the switch from one Non-Gaussian distribution to another. The system starts sampling from one Non-Gaussian distribution and after some time, it is switched to sampling from another Non-Gaussian distribution, which is different from the first distribution.

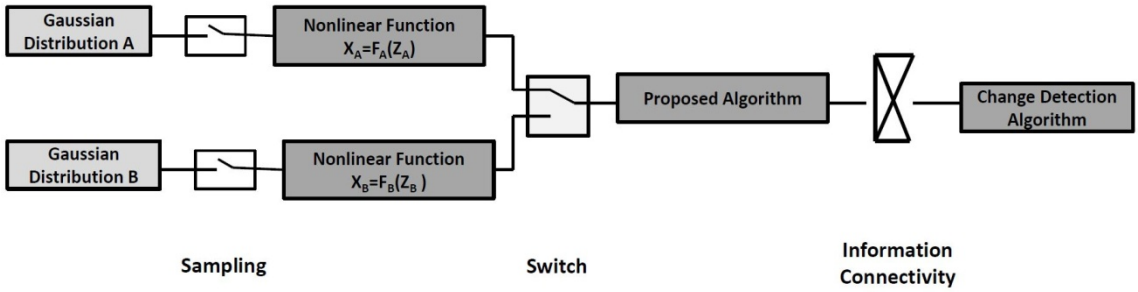


Figure 3.130: Non-Gaussian Exemplary System Representation with Switch

The Gaussian distributions and nonlinear functions for the considered distributions are given as follows:

Gaussian Distribution A

$$\begin{cases} z_{A,1} : N(7,4) \\ z_{A,2} : N(23,10) \\ z_{A,3} : N(40,20) \\ z_{A,4} : N(30,3) \end{cases}$$

Nonlinear Functions

$$\begin{bmatrix} x_{A,1} \\ x_{A,2} \\ x_{A,3} \\ x_{A,4} \end{bmatrix} = \begin{bmatrix} z_{A,1} \cdot z_{A,2} \\ {z_{A,1}}^2 \cdot z_{A,2} + {z_{A,2}}^3 \cdot z_{A,4} \\ {z_{A,3}}^4 \\ {z_{A,4}}^4 \cdot z_{A,2} \end{bmatrix}$$

Gaussian Distribution B

$$\begin{cases} z_{B,1} : N(40,15) \\ z_{B,2} : N(20,2) \\ z_{B,3} : N(5,8) \\ z_{B,4} : N(10,13) \end{cases}$$

Nonlinear Functions

$$\begin{bmatrix} x_{B,1} \\ x_{B,2} \\ x_{B,3} \\ x_{B,4} \end{bmatrix} = \begin{bmatrix} z_{B,2} \cdot z_{B,3} \\ {z_{B,4}}^3 \cdot z_{B,1} + {z_{A,3}}^3 \cdot z_{B,2} \\ {z_{B,3}}^2 + {z_{B,1}}^2 \\ z_{B,2} \cdot {z_{B,1}}^2 \end{bmatrix}$$

For simulation, 1000 data points were sampled from each Gaussian distribution and passed through the nonlinear function. At time t = 1000s, the system is switched from Non-Gaussian distribution x_A to Non-Gaussian distribution x_B. It is assumed that initially no pheromone is present on the ground, a pheromone decay parameter of 0.2 and window sizes of 100 data points. Every 20 seconds, the information connectivity structure of the network is extracted and stored as a graph. The graph similarity measures are then applied to consecutive graphs to detect the change from x_A to x_B.

The results for mutual information threshold of 0.1 are shown in Figure 3.131. It can be seen that all the graph similarity measures can detect the switch between the distributions.

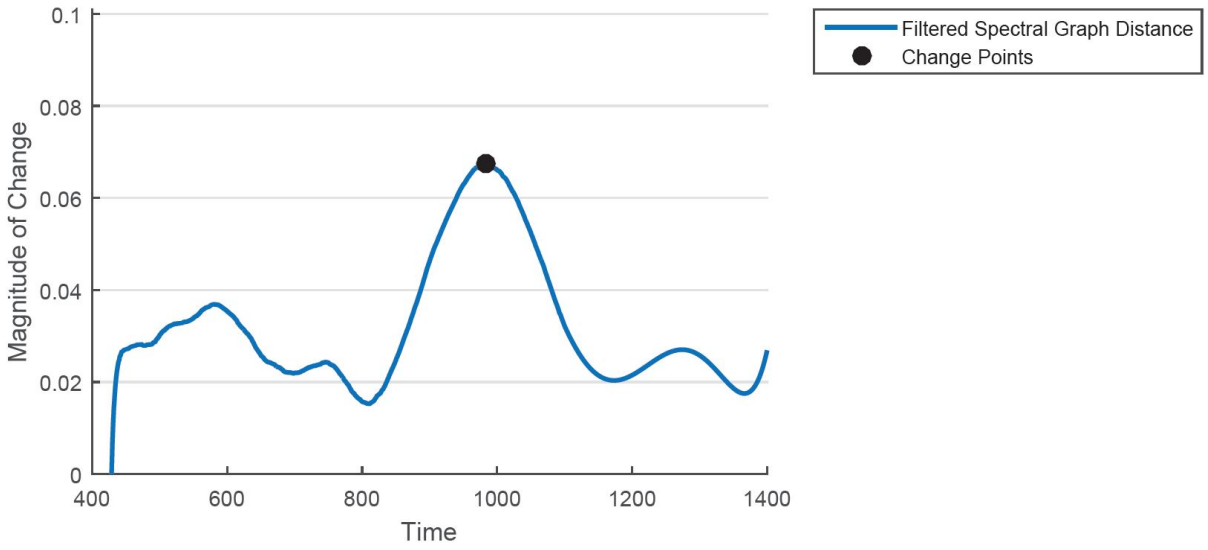


Figure 3.131: Spectral Graph Distance

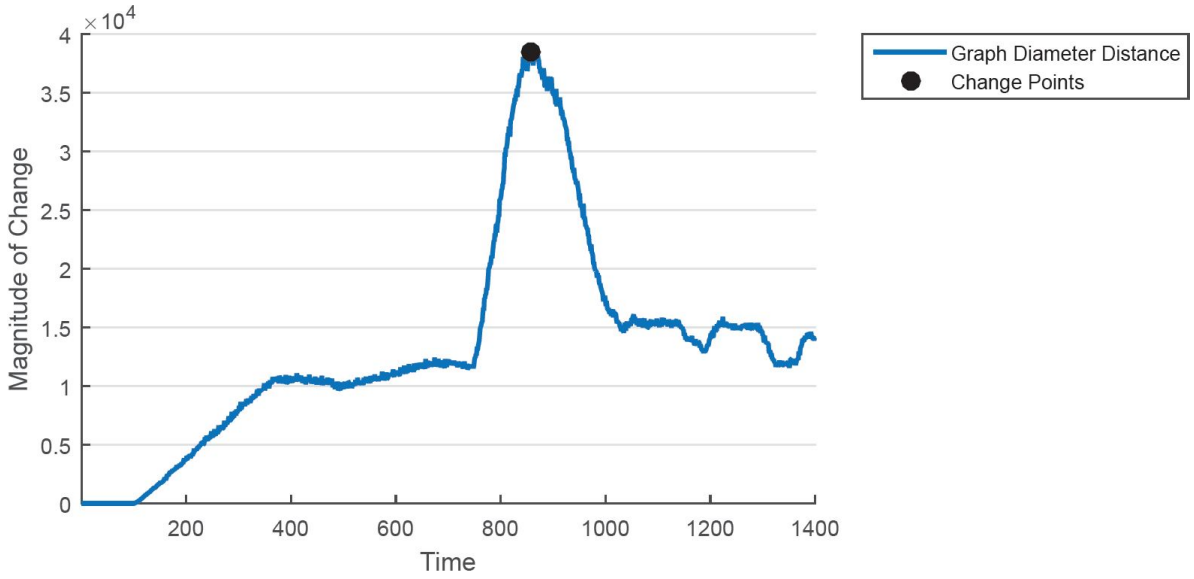


Figure 3.132: Graph Diameter Distance

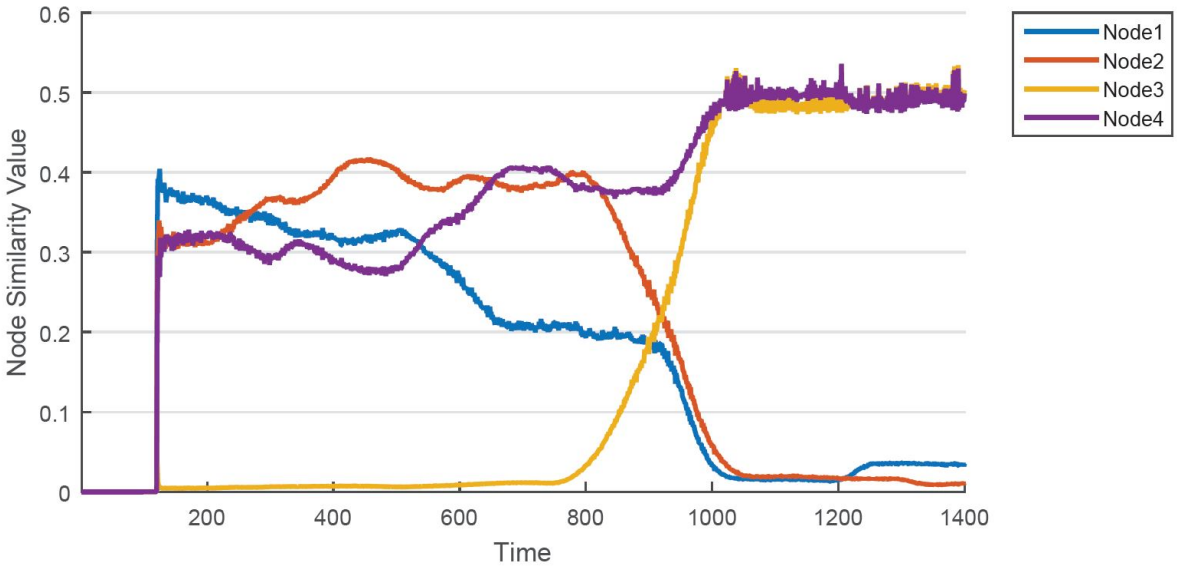


Figure 3.133 - Node Similarity Measure

If we increase the mutual information threshold to 0.4, the extracted communication topology does not reflect the actual intrinsic connectivity structure for both distributions. This stresses the importance of selecting the proper value for threshold. The same can be said for correlation coefficient threshold. We repeat the simulation for the same distributions, but with correlation coefficient calculations instead of mutual information. The threshold for correlation coefficient is considered similar to mutual information: 0.1 and 0.4. The system extracts the correct communication topology for 0.4 threshold, but fails to do the same for 0.1 threshold. We can see that the performance of our algorithm is highly dependent on the value chosen for the threshold.

References for Section 3.0:

Achtert, E., Böhm, C., & Kröger, P. (2006). DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking. *PAKDD'06 Proceedings of the 10th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining* (pp. 119-128). Springer-Verlag Berlin.

Achtert, E., Böhm, C., Kröger, P., & Zimek, A. (2006). Mining Hierarchies of Correlation Clusters. *Scientific and Statistical Database Management, 2006. 18th International Conference on*, (pp. 119 - 128). Vienna.

Achtert, E., Böhm, C., Kriegel, H.-P., Kröger, P., Müller-Gorman, I., & Zimek, A. (2007). Detection and Visualization of Subspace Cluster Hierarchies. In *Advances in Databases: Concepts, Systems and Applications* (pp. 152-163). Springer Berlin Heidelberg.

Achtert, E., Böhm, C., Kriegel, H.-P., Kröger, P., Müller-Gorman, I., & Zimek, A. (2006). Finding hierarchies of subspace clusters. In *Knowledge Discovery in Databases: PKDD 2006* (pp. 446-453). Springer Berlin Heidelberg.

Akyol, B., Haack, J., Ciraci, S., Carpenter, B., Vlachopoulou, M., & Tews, C. (2012). VOLTTRON: An Agent Execution Platform for the Electric Power System. *In Proceedings of the 3rd International Workshop on Agent Technologies for Energy Systems.* Valencia, Spain.

Alexander, S., Bernasconi, J., Schneider, W. R., & Orbach, R. (1989). Excitation dynamics in random one-dimensional systems. *Reviews of Modern Physics , 53* (2), 175-198.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering Points To Identify the Clustering Structure. *roceedings ACM SIGMOD International Conference on Management of Data.* Philadelphia.

Badi, R., & Politi, A. (1997). *Complexity: Hierarchical structures and scaling in physics* (Vol. 6). Cambridge, UK: Cambridge Universiy Press.

Barrat, A., Barthelemy, M., & Vespignani, A. (2008). *Dynamical Processes on Complex Networks.* Cambridge, U.K.: Cambridge University Press.

Bentley, P. J., Gordon, T., Kim, J., & Kumar, S. (2001). New Trends in Evolutionary Computation. *Proceedings of the 2001 Congress on Evolutionary Computation*, (pp. 162-169).

Bernasconi, J., & Schneider, W. R. (1982). Diffusion in a one-dimensional lattice with random asymmetric transition rates. *J. Physics A , 15*, L729.

Bilchev, G., & Parmee, I. C. (1995). The Ant Colony Metaphor for Searching Contiunous Design Spaces. In T. C. Fogarty (Ed.), *AISB Workshop on Evolutionary Computing, Lecture Notes in Computer Science. 993*, pp. 25-39. Berlin: Springer-Verlag.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence, from Natural to Artificial Systems.* Oxford.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems.* New York, NY: Oxford University Press.

Bonabeau, E., Sobkowski, A., Theraulaz, G., & Deneubourg, J.-L. (1996). Quantitative Study of the Fixed Threshold Model for the Regulation of Division of Labour in Insect Societies. *Proc. Royal Society of London B , 263*, 1565-1569.

Bouchaud, J.-P., & Mezard, M. (1994). Self induced quenched disorder: a model for the spin glass transition. *J. Phys. I , 4*, 1109.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (1999). OPTICS-OF: Identifying Local Outliers. *Lecture Notes in Computer Science , 1704*, 262-270.

Campello, R. J., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery in Databases. 7819*, pp. 160-172. Advances in Knowledge Discovery and Data Mining Lecture Notes in Computer Science .

Colorni, A., Dorigo, M., & Maniezzo, V. (1992). An Investigation of Some Properties of an Ant Algorithm. In R. Manner, & B. Manderick (Ed.), *Proc. 1992 Parallel Problem Solving From Nature Conf.* (pp. 509-520). Amsterdam: Elsevier.

Conant, R. C. (1975). Laws of Information which Govern Systems. *IEEE Transactions on Systems, Man and Cybernetics , SMC-6* (4), 240-255.

Cover, T. M., & Thomas, J. A. (2012). *Elements of Information Theory* (Second ed.). Hoboken, NJ: John Wiley & Sons.

Davies, D. L., & Bouldin, D. W. (1979). A Cluster Separation Measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on , PAMI-1* (2), 224 - 227.

Defays, D. (1977). An efficient algorithm for a complete link method. *The Computer Journal , 20* (4), 364-366.

Deneubourg, J., Goss, S., Franks, N., & Pasteels, J. (1989). The blind leading the blind: Modeling chemically mediated army ant raid patterns. *Journal of Insect Behavior , 2*, 719-725.

Deneubourg, j.-L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., & Chretien, L. (1991). The Dynamics of Collective Sorting: Robot-Like Ant and Ant-Like Robot. In J. A. Meyer, & S. W. Wilson (Ed.), *Proc. 1st Conference on Simulation of Adaptive Behavior: From Animals to Animats* (pp. 356-365). Cambridge, MA: MIT Press.

Deneubourg, J.-L., Goss, S., Pasteels, J. M., Fresneau, D., & Lachaud, J.-P. (1987). Self-Organization Mechanisms in Ant Societies (II): Learning in Foraging and Division of Labour. *Experimentia Suppl. , 54*, 177-196.

Derrida, B. (1983). Velocity and diffusion constant of a periodic one-dimensional hopping model. *J. Statistical Physics , 31*, 433-450.

Derrida, B., & Pomeau, Y. (1982). Classical diffusion on a random chain. *Physical Review Letters , 48*, 627-630.

Di Carlo, G., & Dorigo, M. (1998). AntNet: distributed stigmergic control for communications networks. *J. Art. Int. Res. , 9*, 317-365.

Di Caro, G. A., Ducatelle, F., & Gambardella, L. M. (2008). Theory and practice of Ant Colony Optimization for routing in dynamic telecommunications networks. In N. S. Orsucci (Ed.), *Reflecting interfaces: the complex coevolution of information technology ecosystems* (pp. 185-216). Idea Group, Hershey, PA, USA.

Di Caro, G., & Dorigo, M. (1998). AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research , 9*, 317-365.

Dorigo, M. (1994). Learning by Probabilistic Boolean Networks. *IEEE International Confernece on Neural Networks*, *2*, pp. 887-891.

Dorigo, M., Maniezzo, V., & Colorni, A. (1992). *Positive Feedback as a Search Strategy.* Politecnico di Milano, Milan, IT.

Dorigo, M., Trianni, V., Sahin, E., Gross, R., Labella, T. H., Baldassarre, G., et al. (2004). Evolving Self-Organizing Behaviors for a Swarm-bot. *Autonomous Robots , 17*, 223-245.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. *Proceedings of the 12th International Conference on Machine Learning.* San Francisco, CA: Morgan Kaufmann Publishers.

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics , 3* (3), 32-57.

Dussutou, A., Nicolis, S. C., Shephard, G., Beekman, M., & Sumpter, D. J. (2009). The Role Of Multiple Pheromones In Food Recruitment By Ants. *The Journal of Experimental Biology* , 2337-2348.

Englebrecht, A. P. (2005). *Fundamentals of Computational Swarm Intelligence.* West Sussex, U.K.: John Wiley and Sons, Ltd.

Ester, M., Kriegel, H.-p., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, (pp. 226–231).

Fall, K., & Varadhan, K. (2011). The ns Manual.

Farooq, M., & Di Caro, G. A. (2008). Routing Protocols for Next Generation Networks Inspired by Collective Behaviors of Insect Societies: An Overview. In C. B. Merckle (Ed.), *Swarm Intelligence: Introduction and Applications.* Springer.

Feng, X., & Loparo, K. A. (1997). Active Probing for Information in Control Systems with Quantized State Measurements: A Minimum Entropy Approach. *IEEE Transactions on Automatic Control , 42* (2), 216-238.

Feng, X., Loparo, K. A., & Fang, Y. (1997). Optimal state estimation for stochastic systems: an information theoretic approach. *IEEE Transactions on Automatic Control , 42* (6), 771-785.

Fischer, K. H., & Hertz, J. A. (1993). *Spin Glasses.* Cambridge, U.K.: Cambridge University Press.

Haack, J., Akyol, B., Carpenter, B., Tews, C., & Foglesong, L. (2013). Volttron: An Agent Platform for the Smart Grid. *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems.* St. Paul, MN, USA.

Hamerly, G., & Elkan, C. (2002). Alternatives to the k-means algorithm that find better clusterings. *Proceedings of the eleventh international conference on Information and knowledge management* , 600-607.
Hartigan, J., & WongReviewed , M. (1979). A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) , 28* (1), 100-108.
Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning.* New York: Springer.
Haus, J. W., & Kehr, K. W. (1987). Diffusion in regular and disordered lattices. *Physics Reports , 150* (5-6), 263.
Havlin, S., & Ben-Avraham, D. (1987). Diffusion in disordered media. *Advances in Physics , 36*, 695.
He, Z., Chiang, H. D., Li, C., & Zeng, Q. (2009). Fault-section Estimation in Power Systems Based on Improved Optimization Model and Binary Particle Swarm Optimization. *IEEE Power and Energy Society General Meeting*, (pp. 1-8).
Huang, K. (1987). *Statistical Mechanics* (Second ed.). Singapore: Wiley.
Janjarasjitt, S., & Loparo, K. A. (2008). An approach for characterizing coupling in dynamical systems. *Physica D: Nonlinear Phenomena , 237* (19), 2482-2486.
Janjarasjitta, S., & Loparo, K. A. (2008). An approach for characterizing coupling in dynamical systems. *Physica D: Nonlinear Phenomena , 237* (19), 2482-2486.
Kao, Y., & Cheng, K. (2006). An ACO-Based Clustering Algorithm. In *Ant Colony Optimization and Swarm Intelligence* (pp. 340-347). Springer Berlin Heidelberg.
Kirkpatrick, S., Gelatt Jr., C. D., & Vecchi, M. P. (1983). Optimizing by simulated annealing. *Science , 220*, 671.
Kolacinski, R. (2003). A Behavioral Model-Based Approach to Constructing Models of Swarm Behaviors. *Workshop on Agent/Swarm Programming 2003 (WASP'03).* Cleveland, OH.
Kolacinski, R. M. (2003). *Swarming Algorithms to Enhance Connectivity and Performance of Mobile Ad hoc Networks (MANETs).* DARPA Final Report.
Kolacinski, R. (2011). Power System Model Identification via the Thermodynamic Formalism. *Proc. IFCA World Congress.* Milan, IT.
Kolacinski, R., Kanchanaharuthai, A., & Loparo, K. (2011). A General Mathematical Framework for Power System Security and Control. *Proc. IEEE EnergyTech 2011.* Cleveland, OH.
Kotsianis, S., & Kanellopoulos, D. (2006). Discretization Techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering , 32* (1), 47-58.
Kriegel, H.-P., Kröger, P., Sander, J., & Zimek, A. (2011). Density-based Clustering. *WIREs Data Mining and Knowledge Discovery , 1* (3), 231-240.
Kuntz, P., Layzell, P., & Snyers, D. (1997). A Colony of Ant-Like Agents for Partitioning in VLSI Technology. In P. Husbands, & I. Harvey (Ed.), *Proc. 4th European Conference on Artificial Life* (pp. 417-424). Cambridge, MA: MIT Press.
Liui, L., & Cartes, D. A. (2006). Particle Swarm Optimization for Automatic Diagnosis of PMSM Stator Fault. *Proceedings of the American Control Conference*, (pp. 3026-3031).
Lloyd, S. P. (1982). Least squares quantization in PCM. *Information Theory, IEEE Transactions on , 28* (2), 129-137.
Lumer, E. D., & Faieta, B. Diversity and adaptation in populations of clustering ants. *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3: from animals to animats 3*, (pp. 501-508).

Lumer, E., & Faieta, B. (1994). Diversity and Adaptation in Populations of Clustering Ants. *Proc. 3rd International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3* (pp. 499-508). Cambridge, MA: MIT Press.

Mann, T. P. (2006). Numerically stable hidden Markov model implementation. 1-8.

Marinari, E., Parisi, G., & Ritort, F. (1994). Replica field theory for deterministic models (II): a non random spin glass with glassy behaviour. *J. Phys. A , 27*, 7647.

Marinari, E., Parisi, G., & Ritort, F. (1994). Replica field theoy for deterministic models: binary sequences with low autocorrelation. *J. Phys. A , 27*, 7615.

Mezard, M., Parisi, G., & Virasoro, M. (1986). *Spin glass theory and beyond.* Singapore: World Scientific.

Mirollo, R. E., & Strogatz, S. H. (1990). Synchronization of Pulse-coupled Biological Oscillators. *SIAM J. App. Math. , 50* (6), 1645-1662.

Mitchell, R., & McKim, J. (2002). *Design by Contract: by example.* Addison-Wesley.

Murthy, K. P., & Kehr, K. W. (1989). Mean first-passage time of random walks on a random lattice. *Physical Review A , 40* (4), 2082.

Nock, R., & Nielsen, F. (2006). On Weighting Clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on , 28* (8), 1223-1235.

Oida, K., & Kataoka, A. (1999). Lock-free AntNet and its evaluation adaptiveness. *Journal of IEICE B (in Japenese)* , 1309-1319.

Oprisan, S. A., Holban, V., & Moldoveanu, B. (1996). Functional Self-Organization Performing Wide-Sense Stochastic Processes. *Phys. Lett. A , 216*, 303-306.

Palmer, D. W., Hantak, C. M., & Kovacina, M. A. (1999). Impact of Behavior Influence on Decentralized Control Strategies for Swarms of Simple, Autonomous Mobile Agents. *Proc. Workshop: Biomechanics Meets Robotics Modeling and Simulation of Motion.* Heidelberg, Germany.

Palmer, R. G. (1982). Broken ergodicity. *J. Adv. Phys. , 31*, 669.

Parpinelli, R. S., Lopes, H. S., & Freitas, A. A. (2002). An Ant Colony Algorithm for Classification Rule Discovery. In H. Abass, & R. Sarker (Ed.), *In Data Mining: A Heuristic Approach* (pp. 191-208). London, UK: Idea Group Publishing.

Parpinelli, R. S., Lopes, H. S., & Freitas, A. A. (2002). Data Mining with an Ant Colony Optimization Algorithm. *IEEE Trans. On Evolutionary Computation , 6* (4).

Pikovsky, A., Rosenblum, M., & Kurths, J. (2001). *Synchronization: A universal concept in nonlinear sciences* (Vol. 12). Cambridge, U.K.: Cambridge University Press.

Roy, S., & Bhattacharyya, D. (2005). An Approach to Find Embedded Clusters Using Density Based Techniques. *LNCS , 3816*, 523 – 535.

Saleem, M., Di Caro, G. A., & Farooq, M. (2011). Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions. *Information Sciences , 181* (20), 4597-4624.

Santos, D. S., & Bazzan, A. L. (March 30 2009-April 2 2009). A Biologically-Inspired Distributed Clustering Algorithm. *Swarm Intelligence Symposium, 2009. SIS '09. IEEE*, (pp. 160-167).

Schneider, J., & Vlachos, M. (2013). Fast parameterless density-based clustering via random projections. *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, (pp. 861-866). New York.

Schoonderwoed, R., Holland, O., Bruten, J., & Rothkrantz, L. (1996). Ant-based load balancing in telecommunications networks. *Adaptive Behavior , 5* (2), 169-207.

Schoonderwoerd, R., Bruten, J. L., Holland, O. E., & Rothkrantz, L. J. (1996). Ant-Based Load Balancing In Telecommunications Networks. *Adaptive Behavior , 5* (2), 169-207.

Schoonderwoerd, R., Holland, O., & Bruten, J. (1997). Ant-like agents for load balancing in telecommunications networks. *Proceedings of the First International Conference on Autonomous Agents*, (pp. 209-216).
Scott, D. W. (1979). On Optimal and Data-based Histograms. *Biometrika , 66* (3), 605-610.
Shelokar, P., Jayaraman, V., & Kulkarni, B. (2004). An ant colony approach for clustering. *Analytica Chimica Acta , 509* (2), 187–195.
Sibson, R. (1972). SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal , 16* (1), 30-34.
Stein, D. L., & Newman, C. M. (2013). *Spin Glasses and Complexity*. Princeton, NJ: Princeton University Press.
Steward, S., & Appleby, S. (1994). Mobile Software Agents For Control Of Distributed Systems Based On Principles Of Social Insect Behaviour. *Singapore ICCS* (pp. 549-553). IEEE.
Teh, Y. W. (2010). Dirichlet Processes. In C. Sammut, & G. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 280-287). Springer.
Tonguz, O. K. (2011). Biologically Inspired Solutions to Fundamental Transportation Problems. *IEEE Communications Magazine , 49*, 106-115.
Toulouse, G. (1977). Theory of the frustration effect in spin glasses I. *Communication Physics , 2*, 15.
Tyrell, A., Auer, G., & Bettstetter, C. (2006). Fireflies as Role Models for Synchronization in Ad hoc Networks. *Proc. 1st int. conf. on Bio inspired models of network, information and computing systems (BIONETICS 2006).* Cavalese, Italy: ACM, New York, NY.
Vizine, A. L., De Castro, R. N., Hruschka, E. R., & Gudwin, R. R. (2005). Towards Improving Clustering Ants: An Adaptive Ant Clustering Algorithm. *Informatica , 29*, 143--153.
Wilson, E. O. (1971). *The Insect Societies.* The Belkap Press of Harvard University Press.
Wodrich, M. (1996). *B. Sc. Thesis, Ant Colony Optimization.* South Africa: University of Capetown, Dept. of Electrical and Electronic Engineering.
Yang, S., Lou, X., Neuschaefer, C., Boisson, P., & Bories, R. (2013). Model Predictive Control for Alstom's 1000 MWe Ulrtra-Supercritical Boiler and Steam Plant. *56th Annual ISA POWID Symposium.*
Yang, Y., & Kamel, M. (2003). Clustering ensemble using swarm intelligence. *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, (pp. 65 - 71).
Yang, Y., & Kamel, M. S. (2006). An aggregated clustering approach using multi-ant colonies algorithms. *Pattern Recognition , 39* (7), 1278-1289.
Zhang, Z., Long, K., Jianping, J., & Falko, F. (2014). On Swarm Intelligence Inspired Self-Organized Networking: Its Bionic Mechanisms, Designing Principles and Optimization Approaches. *IEEE Communications Surveys & Tutorials*, *16*, pp. 513 - 537.
Zhou, L., & Franklin, S. (1994). Character Recognition Agents. *rtificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems* (pp. 301-306). MIT Press.

Section 4.0: Fault Simulation for Steam Power Plant

Summary:
This section summarizes the fault simulation work using GE's dynamic model for its typical 1000 MW steam power plant design, and the fault simulation work is aimed to evaluate a set of new algorithms for fault detection. The list of possible fault scenarios and their corresponding model revisions in APROS are described and a set of biases and noises have been added to the Apros model to simulate the predefined boiler fault scenarios (sensor, actuator and process faults) to best support the testing of the fault detection methods. The simulation data including the dynamic responses of concerned process variables are illustrated for each scenario examined.

Purpose and scope

This section summarizes the fault simulation work using GE's dynamic model for its typical 1000 MW steam power plant design. A list of possible fault scenarios and their corresponding model revisions in APROS is given. A set of biases and noises have been added to the Apros model to simulate the predefined boiler fault scenarios (sensor, actuator and process faults) to best support the testing of the fault detection methods.

In order to fulfill the purpose, the following tasks have been planned:

1. Identify and list the fault conditions and scenarios which are feasible for simulation using the Apros dynamic model;
2. Modify and update Apros model to be ready for fault detection testing;
3. Run the proposed fault scenarios on Apros model and generate data for the university team to test the new fault detection algorithms;
4. Review, report development and consulting support.

Dynamic model description

In this section, the dynamic models along with closed-loop controls used in the simulation tool developed are described. Figure 4.1 shows the schematic for the 1000 MW coal-fired steam power plant.

Process models of the Boiler

The dynamic boiler process model determines the pressures, temperatures, and mass flow rates of water/steam and of air/flue gas at various points in the system as a function of time. The system is converted to a network of nodes and branches. At each time step the conservation equations for mass, energy and momentum for the network are solved. In addition, heat transfer equations between fluid and metal parts and heat conduction within metal parts are solved.

The model includes the physical arrangement data for all the important components and systems of the boiler that are of interest, e.g., the furnace, waterwalls, superheater, reheater, economizer, backpass, pipes, pumps and valves. The process components are modeled with sufficient details and fidelity to assure realistic and accurate results. The air heater system, the steam turbine system, the condensate and feedwater system upstream of the feedwater pump have not been modeled. Therefore, transient parameters such as the air temperature at the furnace entrance, steam temperature at the reheater inlet, and the feedwater temperature at the feedwater pump inlet, have been modeled as boundary conditions related to

the unit load demand. Since the fault simulation in this project mainly occurs in the boiler, in particular the superheater heat exchangers, the detailed description of the boiler is shown below. It should be noted that this work can be further applied to the whole plant model.

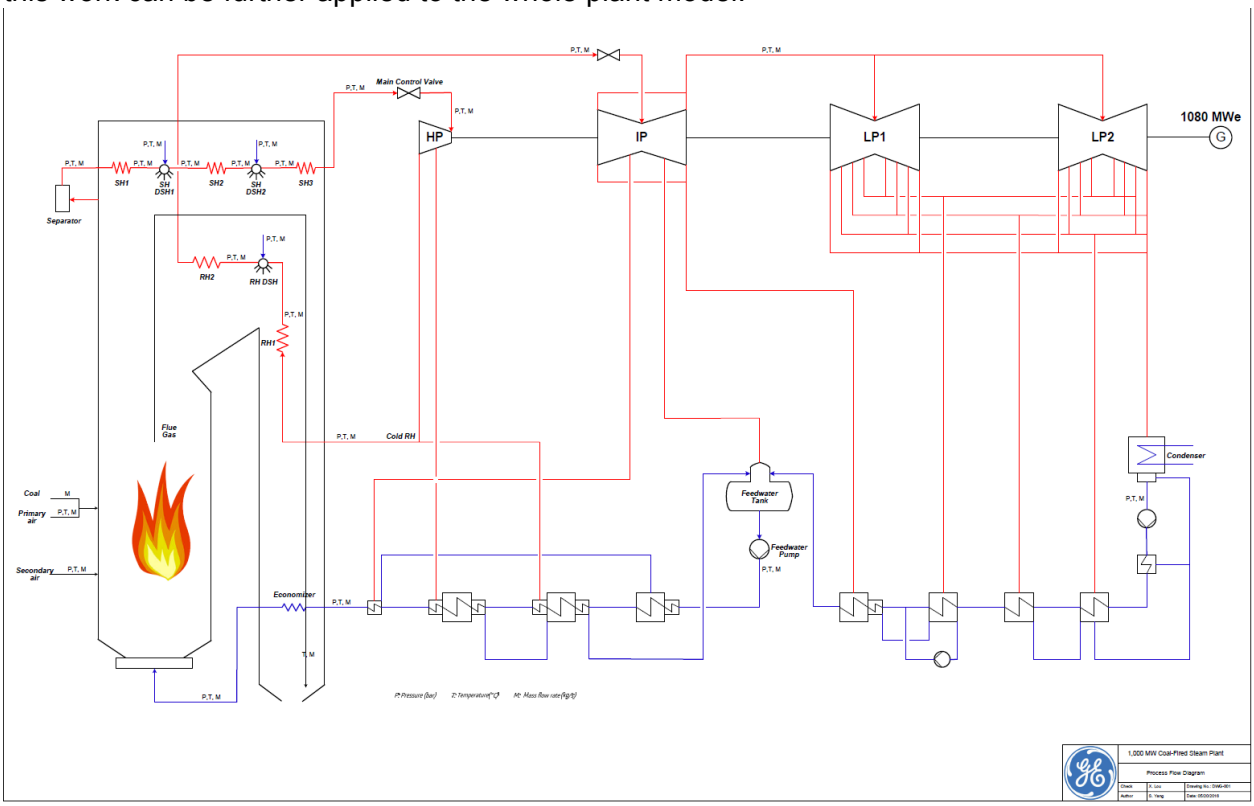


Figure 4.1: 1000MW Coal-fired steam plant schematic

Water/Steam

Feedwater coming from the HP heaters is sent to the boiler and passes through sequentially a series of heat exchangers to be evaporated. The feedwater first enters the economizer for warming up and then goes into the waterwalls, where most radiant heat is absorbed by the water. For once-through operation of this model, the water changes continuously to steam in the waterwall sections. The steam then goes through the separator and absorbs some additional radiant heat by passing the roof heat exchangers. After the radiant roof section, the steam enters the superheaters. The superheater sections comprise of SH screen, cavity, panel, platen and final superheater. There are two stages of SH sprays with the primary spray installed between panel and platen, and the final spray installed between platen and final superheaters. Both sprays come from the downstream of feedwater pump. The main steam outlet the final superheater goes to the turbine sections. The SH section diagram is shown in Figure 4.2.

The steam from the HP turbine is sent back to the boiler for reheating. The reheater sections are divided into reheater horizontal, pendant and final. There is one stage of RH spray with its pressures and temperatures specified as boundary conditions.

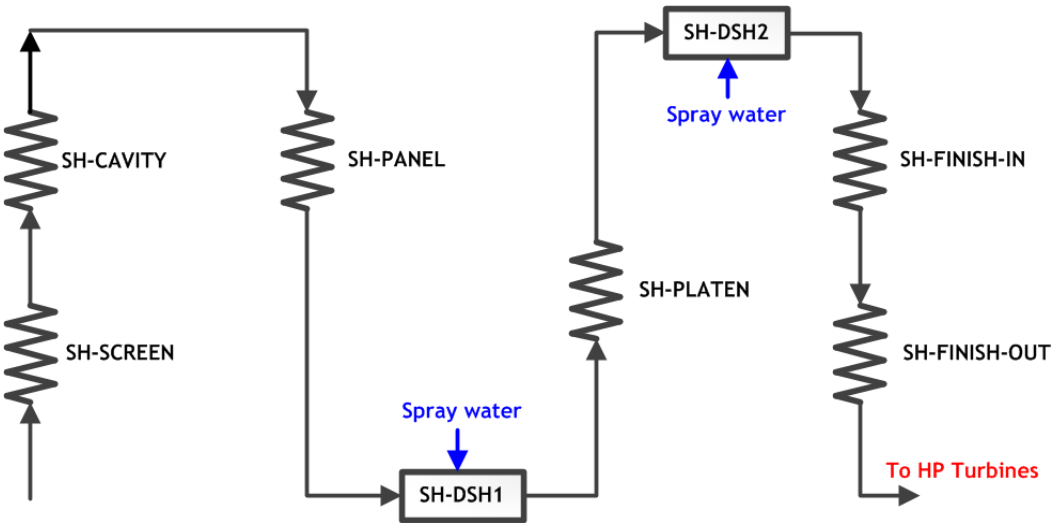


Figure 4.2: Steam – Superheater section

Description of main steam temperature controls

As two stages of SH spray are used, the SH steam temperature control consists of the primary spray control and the final spray control, shown in Figure 4.3. The control structure of the final spray control is a cascade arrangement where the first PI controller acts on the superheater outlet temperature as compared with the setpoint dependent on the boiler master. The resulting master output provides the setpoint to a second controller, which acts on this setpoint as compared to steam temperature measured at the superheater final desuperheater outlet. The slave controller output provides the valve position demand for the final spray.

The primary spray control is also configured as a cascade arrangement. The first PI controller acts on the final spray differential temperature as compared with the load dependent DT setpoint. The resulting master output provides the setpoint to a second controller, which acts on this setpoint as compared to the steam temperature measured at the primary spray outlet. The second PI controller output provides the valve position demand for the primary spray. The control demand of the primary and final valve positions are both limited by its associated saturation protection logic.

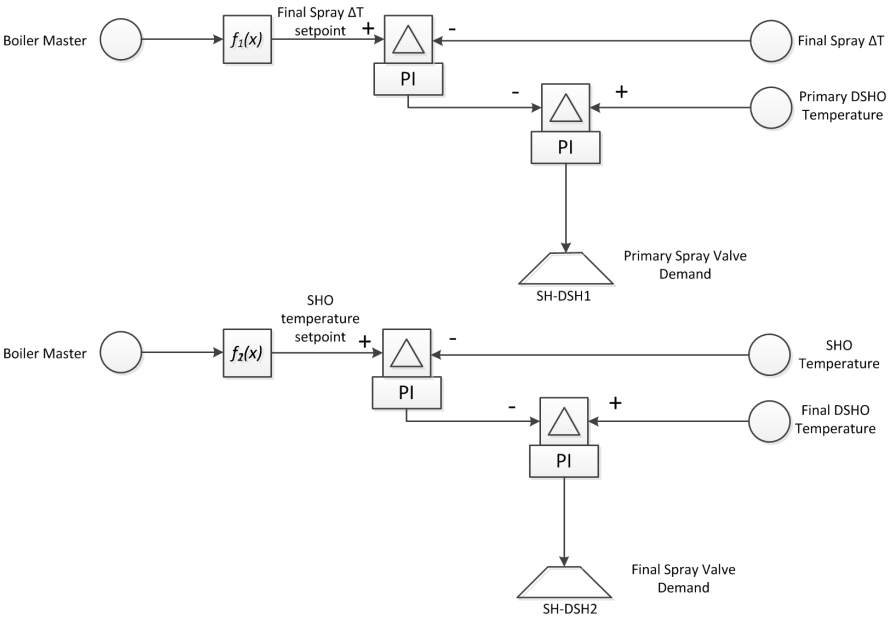


Figure 4.3: SH steam temperature control (2 stages)

Fault scenario and modeling action

Three types of faults are included in this work: process fault, sensor fault and actuator fault.

Process fault

Process item: superheater heat exchanger (SH1, SH2 and SH3)

Fault type: fouling/slagging in the superheater, which causes gradual change of the heat transfer coefficient.

Modeling action: apply predetermined ramping curve of the heat transfer coefficient to the superheater and obtain the data after the model reaches steady state. The ramping curve should refer to the field data to reflect realistic situation.

Process item: reheater heat exchangers (RH1 and RH2)

Fault type: fouling/slagging in the reheater, which causes gradual change of the heat transfer coefficient.

Modeling action: apply ramping curve for the heat transfer coefficient to the reheater.

Sensor fault

Sensor item: temperature sensors for the two stage de-superheaters (SH-DSH1 and SH-DSH2)

a) Fault type: sensor bias of various levels (both negative and positive)

Modeling action: apply bias to the temperature measurement. The bias applied to the model should refer to field data.

b) Fault type: out-of range sensor failure

Modeling action: apply bias to the temperature measurement to exceed the bound of the sensor measurement.

Sensor item: temperature sensor at Separator outlet
Fault type: sensor bias of various levels (both negative and positive)

Modeling action: apply bias to the temperature measurement. The bias applied to the model should refer to field data.

Actuator faults

Actuator item: water spray valve actuator for the two stage de-superheater (SH-DSH1 and SH-DSH2)

a) Fault type: actuator bias of various levels (both negative and positive)

Modeling action: apply bias to the position of the water spray valves. The bias applied to the model should refer to field data.

b) Fault type: out-of range actuator failure

Modeling action: apply bias to the position of the water spray valves to exceed the maximum position of the valves.

Boiler fault simulation study
A series of modeling efforts have been performed to simulate the listed faults associated with boiler operations. Three attempts have been carried out to generate fault simulation and shared with the team at Case Western Reserve University. First, all the fault scenarios have been simulated separately, including 8 cases with sensor fault simulation, 1 case with process fault simulation and 5 cases with actuator simulation. Secondly, to support off-line testing of the fault detection algorithm, a normal data sequence of all three types of faults with longer normal time and short ramping time was generated. Lastly, additional simulation experiment was generated per university request, including repeating the same fault after some time, and simulating the fault sequence with different fault intervals (time between two faults occur).

Sensor fault simulation
Eight individual scenarios were simulated for sensor faults, including step bias of ±5ºC for temperature measurement for primary desupheater SH-DSH1/final desuperheater SH-DSH2 and ramp bias of ±5ºC for temperature measurement for primary desupheater SH-DSH1/final desuperheater SH-DSH2. Figure 4.4 presents the transient profile of the spray water mass flow of the primary and final spray system in response to the step bias. Step bias of +5ºC is applied to the temperature sensor for the final de-superheater SH-DSH2 at time 30 minutes. After the bias is applied, the water spray mass flow for de-superheater SH-DSH2 increases and the water spray mass flow for de-superheater SH-DSH1 decreases.

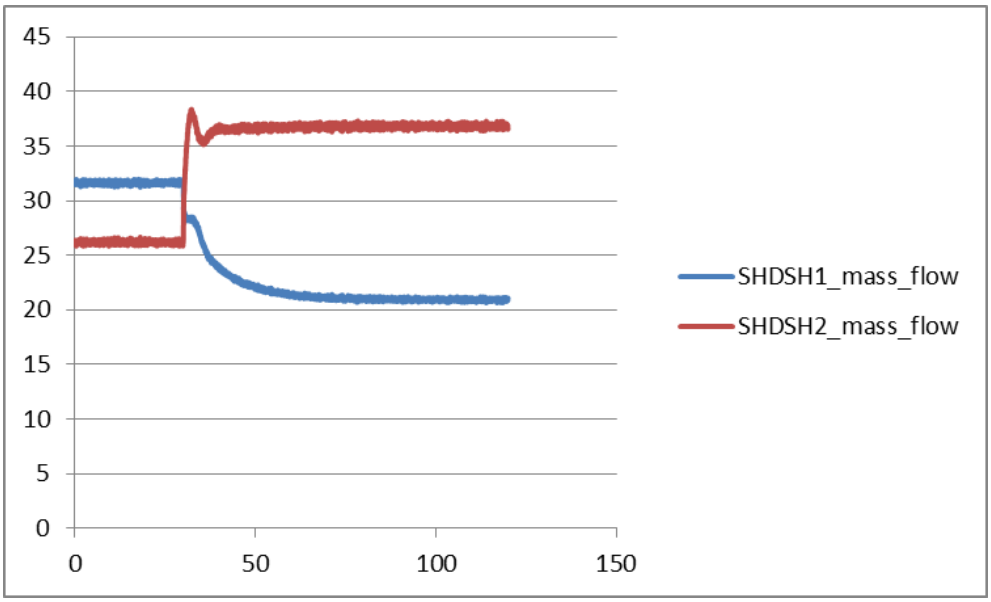


Figure 4.4: Transient profile of the spray water mass flows of the primary and final spray system. Step bias of +5ºC is applied to the temperature sensor for the final de-superheater SHDSH2.

Process fault simulation

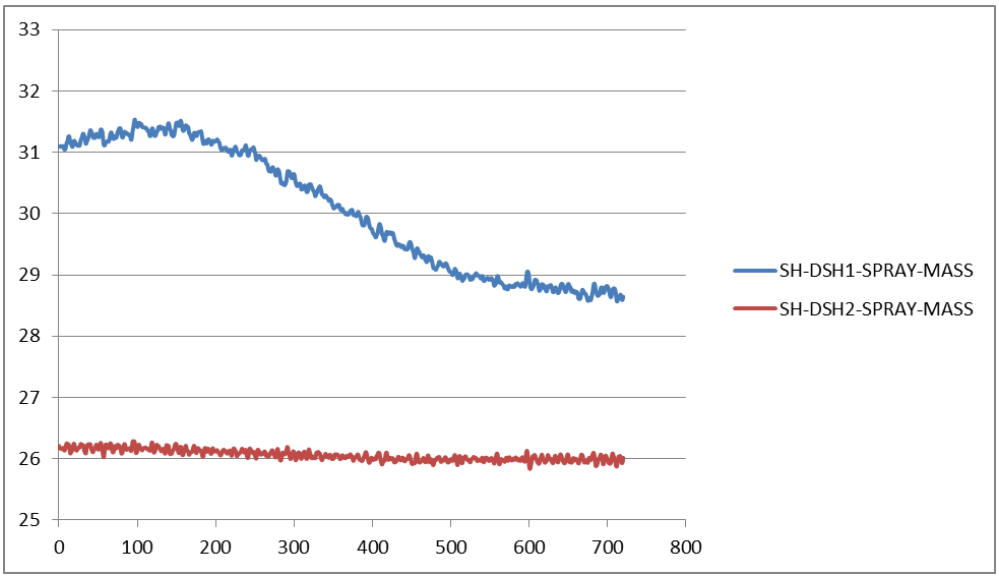


Figure 4.5: Transient profile of the spray water mass flow of the primary and final spray system. Ramping curve of - 5% change is applied for heat transfer coefficient of all the superheater heat exchangers (time when bias is applied: 2 hours; ramping duration: 6 hours).

One scenario is simulated for the process fault. Fouling/slagging in the superheater heat exchangers would cause gradual change of the heat transfer coefficient. A ramping curve of decreasing the heat transfer coefficient by 5% has been applied to all the superheater heat exchangers. Figure 4.5 presents the transient profile of the spray water mass flow of the primary and final spray system. The bias is applied at the time of 2 hours, and the ramping duration is 6 hours. Since less heat is transferred to the water side, the amount of the spray water mass flow gradually decreases to maintain the superheater outlet temperature.

Actuator fault simulation

Five scenarios are simulated for the actuator fault, including step bias/ramp bias of ±0.05 of the actuator for final desuperheater SH-DSH2 valve, and one case of ramp bias of +0.5 to reach lower bound of actuator. Figure 4-3 presents the transient profile of the final stage slave controller output and final stage valve position. Step bias of +0.05 is applied to the actuator for final stage desuperheater SH-DSH2 valve at time 60 minutes. Figure 4-4 presents the transient profile of the final stage slave controller output and final stage valve position. A ramp bias of +0.5 is applied to the actuator for final stage desuperheater SH-DSH2 valve at time 120 minutes for 6 hours.

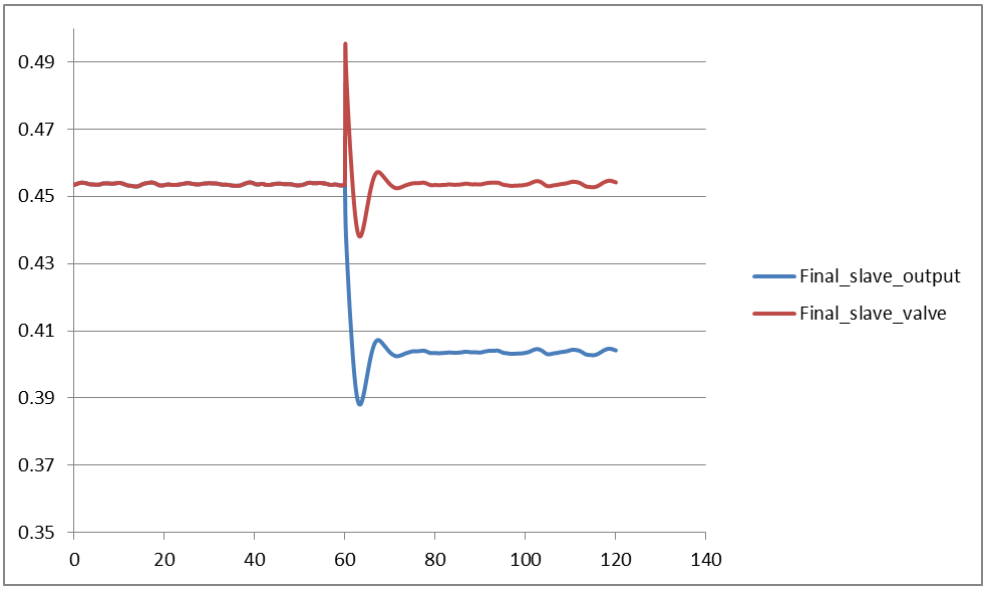


Figure 4.6: Transient profile of the final stage slave controller output and final stage valve position. Step bias of +0.05 is applied to the actuator for final stage desuperheater SH-DSH2 valve.

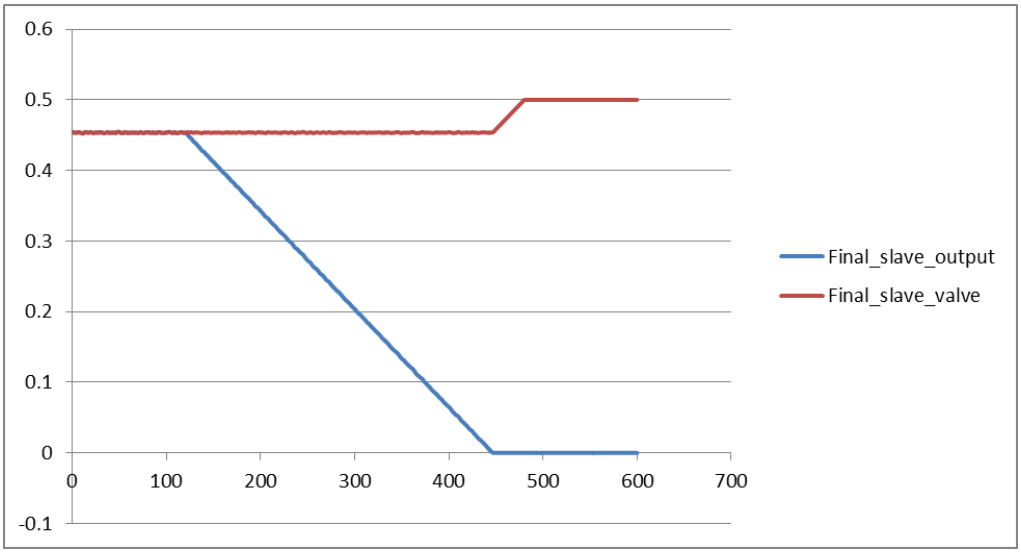


Figure 4.7: Transient profile of the final stage slave controller output and final stage valve position. Ramp bias of +0.5 is applied to the actuator for final stage desuperheater SH-DSH2 valve (time when bias is applied: 2 hours; ramping duration: 6 hours).

Conclusions

The fault simulation work based on the dynamic model are summarized. Three types of faults including sensor fault, process fault and actuator fault are simulated to support the testing of the new fault detection algorithm developed by the university. Firstly, all the fault scenarios have been simulated separately for the algorithm testing; secondly, a normal sequence of all three types of fault with longer normal time and short ramping time was generated; lastly, additional simulation experiment was performed including repeating the same fault after some time, and simulating the fault sequence with different fault intervals. The simulation data have been shared with the team at Case Western Reserve University for the fault detection algorithm testing. The off-line testing results have been shared with DOE NETL. DOE NTEL acknowledged this collaborated project work and indicated that a continued project should be planned between the company and the university on continued fault detection method evaluation using simulation and further industrial pilot tests and/or field tests.

References

- Venkatasubramanian, et al. A review of process fault detection and diagnosi Part I: Quantitative model-based methods, Computers and Chemical Engineering, 27 (2003) 293-311
- Venkatasubramanian, et al. A review of process fault detection and diagnosi Part II: Qualitative model-based methods and search strategies, Computers and Chemical Engineering, 27 (2003) 313-326
- Venkatasubramanian, et al. A review of process fault detection and diagnosi Part III: Process history based methods, Computers and Chemical Engineering, 27 (2003) 327-346
- Agharazi. A swarm Intelligent Approach to Condition Monitoring of Dynamic Systems. PhD thesis. Case Western Reserve University, 2016
- Agharazi, et al. An ant foraging behavior approach to condition monitoring of power generation plants. 59th annual ISA POWID Symposium, Charlotte, North Carolina, 2016.
- Loparo and Kolacinski, An information Theoretic Framework and Self Organizing Agent-based Sensor Network Atchitecture for Power Plant Condition Monitoring. NETL Report, Oct, 2016

Section 5.0: Condition Monitoring System and Results

MATLAB Platform

In this section, we explain the MATLAB platform developed for the proposed algorithm. The user can upload data from the computer, define simulation specifications, and perform simulation. The results are then listed in the main window and the user selects the desired results to be shown. At the end, the results can be saved into MATLAB workspace or into an Excel file

.

The main window of the platform is shown in Figure 5.**3**. The user loads data from computer. Data should be in a single Excel file, stored in columns where the first row contains the titles. After loading data, the user can plot the data and define the following simulation specifications:

- **Calculation Approach**: The user can choose between correlation coefficient and mutual information, the method used for calculating the similarity between data.
- **Window Size**: This defines the size of the data agents carry from their home node. These windowed data are use for calculating the similarity between data at different locations of the system.
- **Desired Threshold**: The desired threshold for correlation coefficient or mutual information value that declares high similarity between system elements.

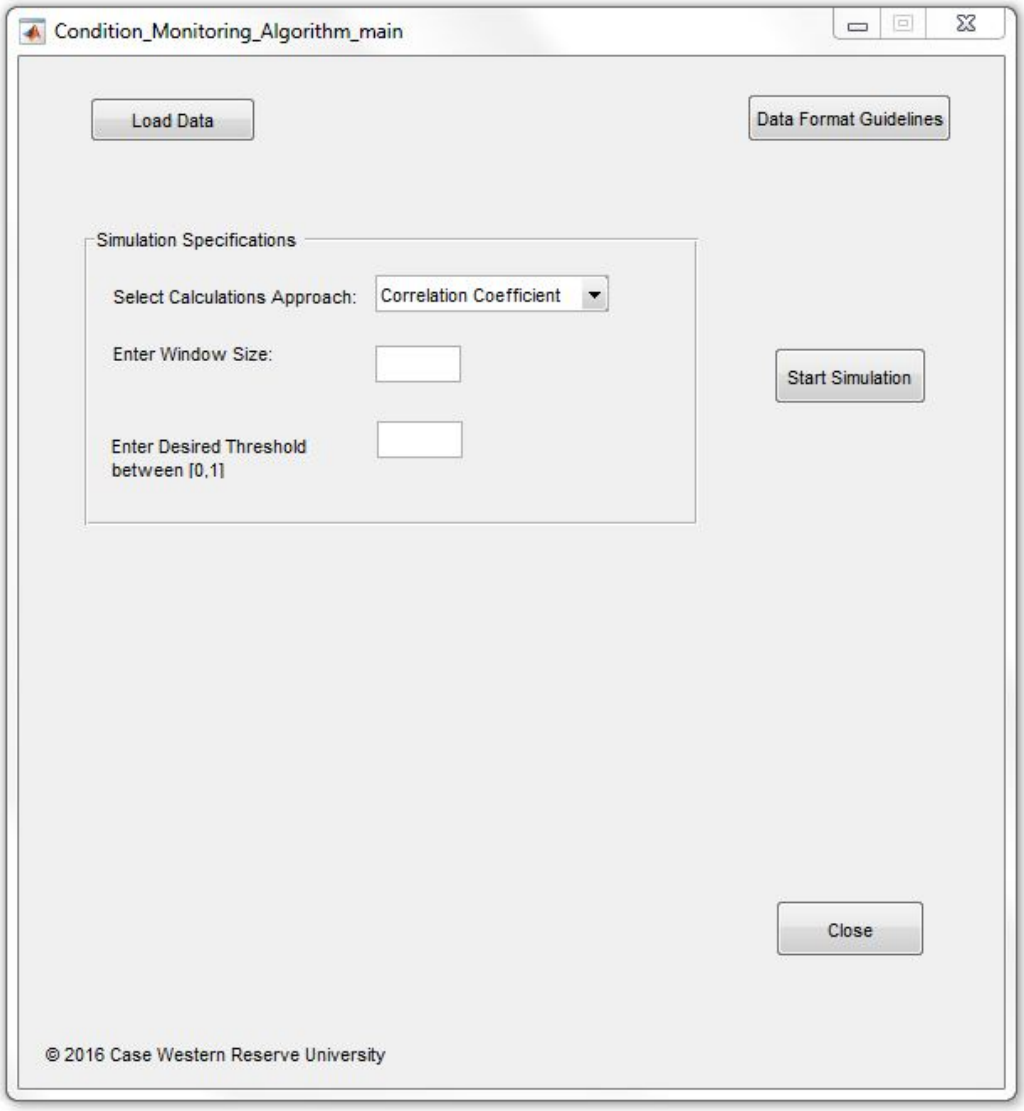


Figure 5.3: Main Window

We select a Non-Gaussian exemplary network with a switch, similar to the system described previously in Section 3.0. The data is shown in Figure 5.**4**.

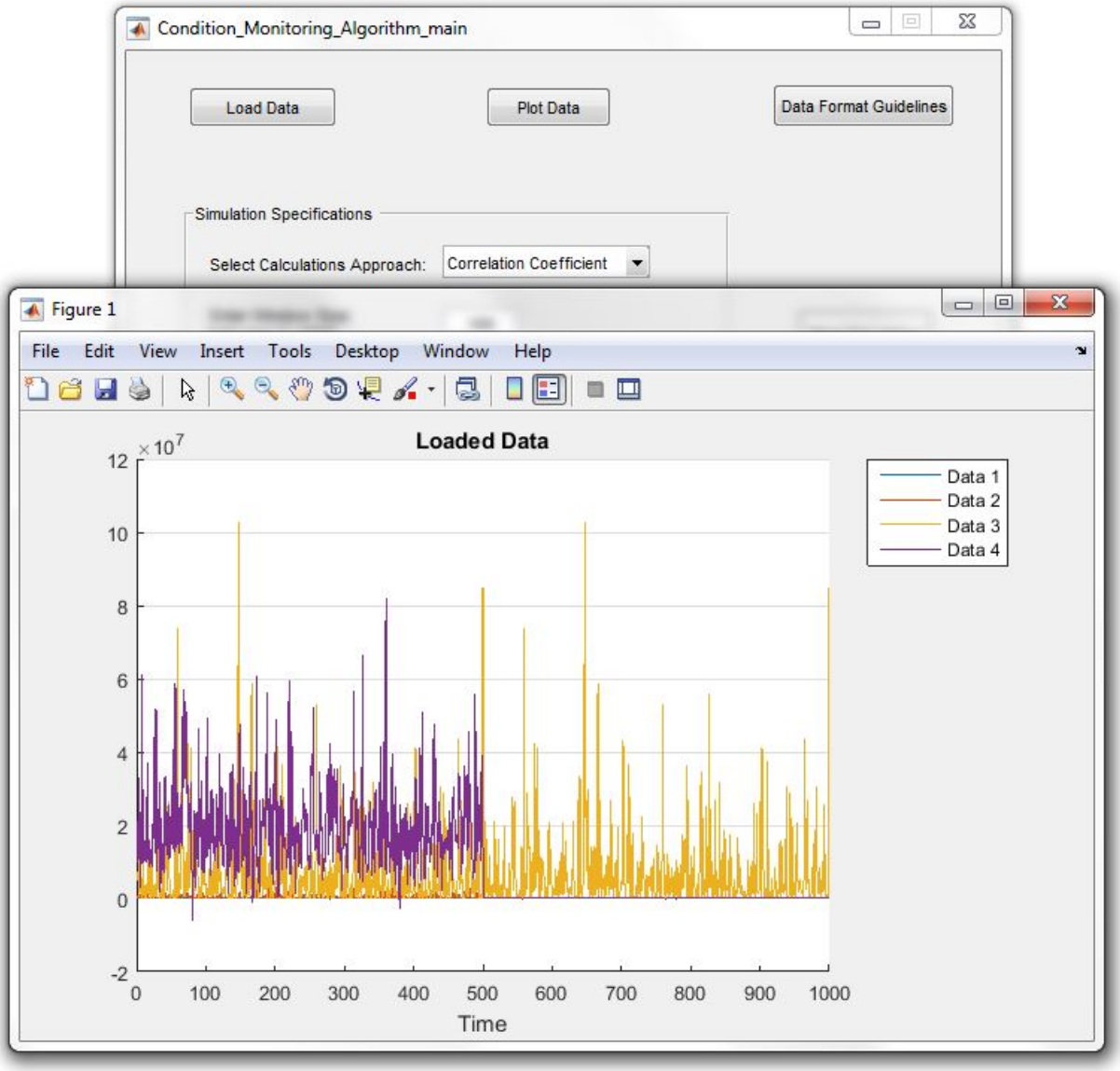


Figure 5.4: Loaded Data

Correlation Coefficient Approach

The simulation is performed for correlation coefficient approach with windows of size 100 and desired threshold of 0.4. Figure 5.**5** shows the simulation progress.

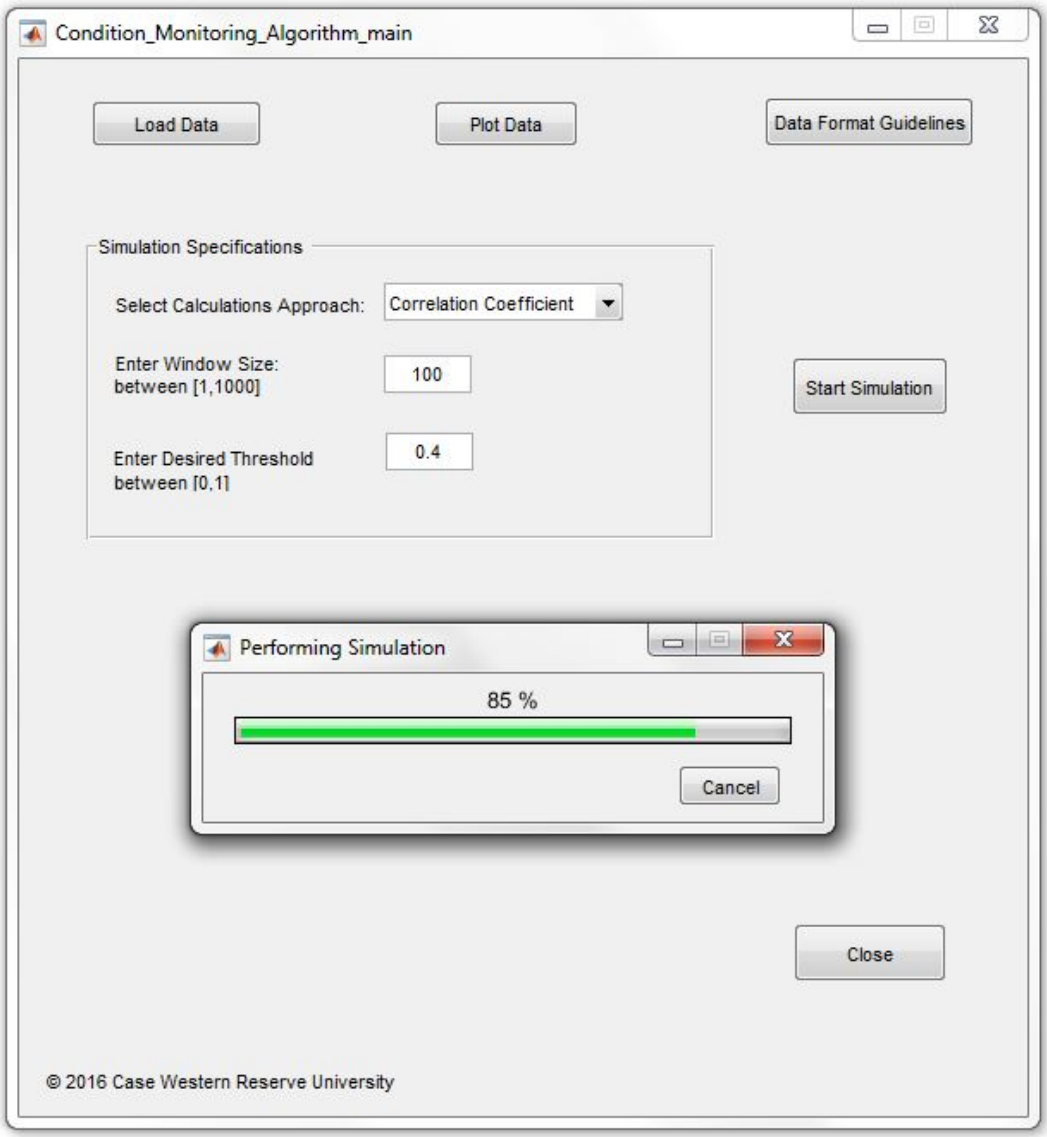


Figure 5.5: Simulation Progress

After the simulation finishes, the 'Simulation Results' panel becomes visible which shows the options for plotting the results as shown in Figure 5.**6**.

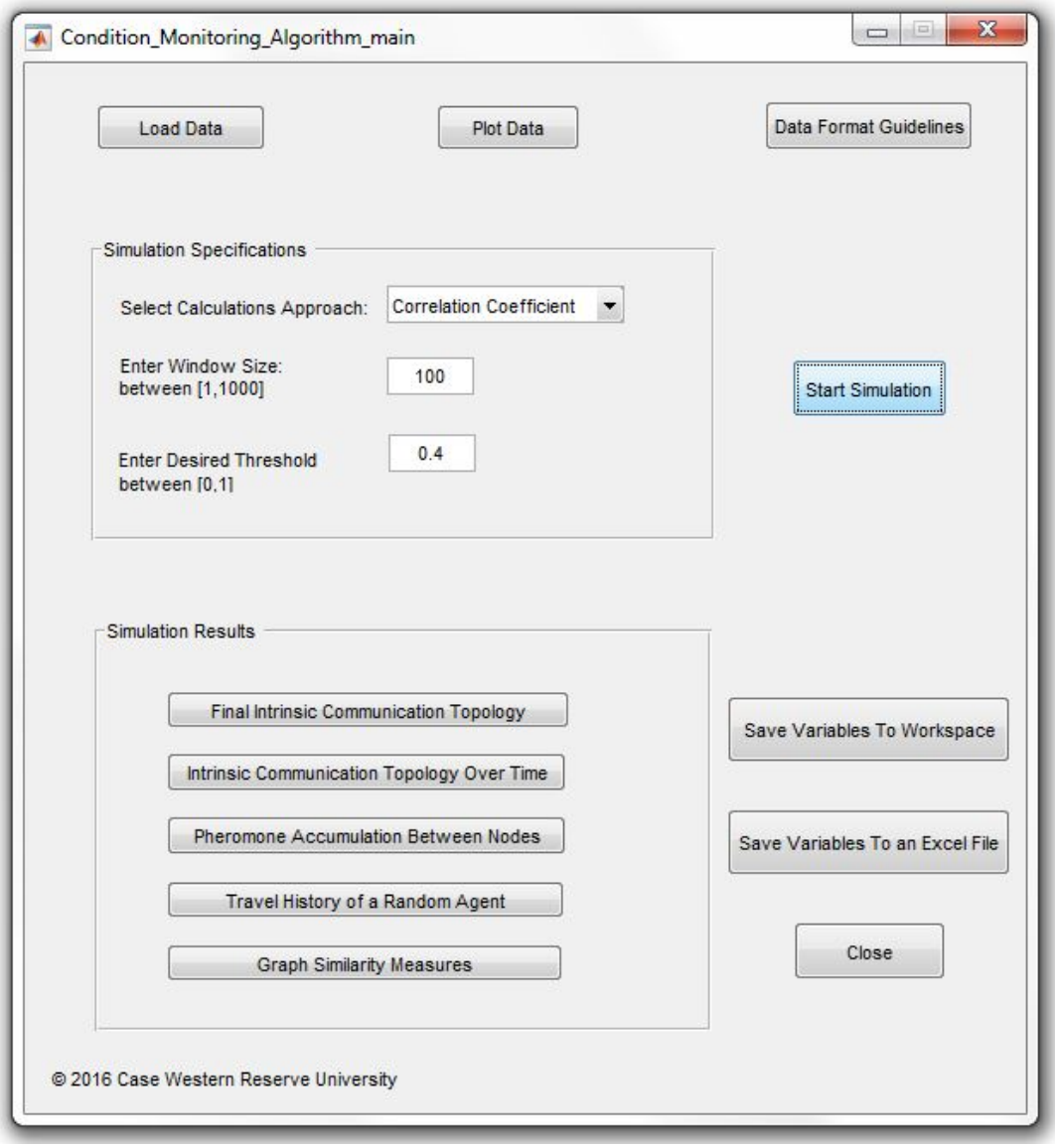


Figure 5.6: Simulation Results Panel

The simulation results are shown in Figure 5.**7** to Figure 5.**13**.

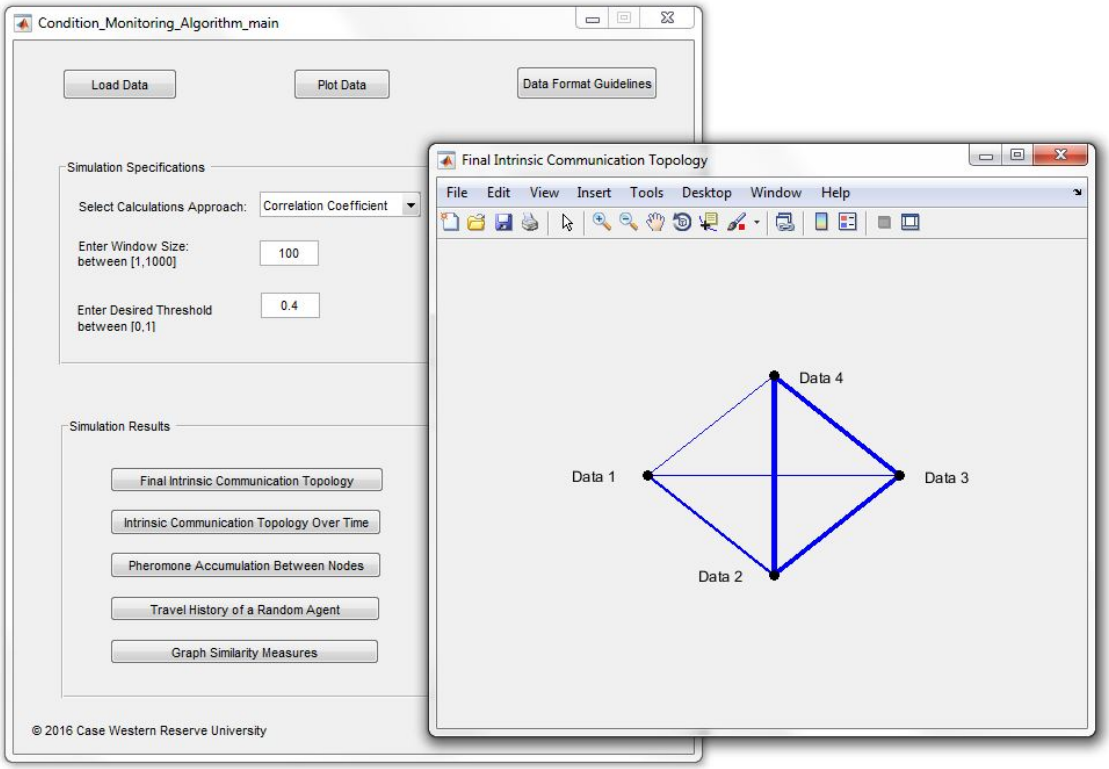


Figure 5.7: Final Intrinsic Communication Topology

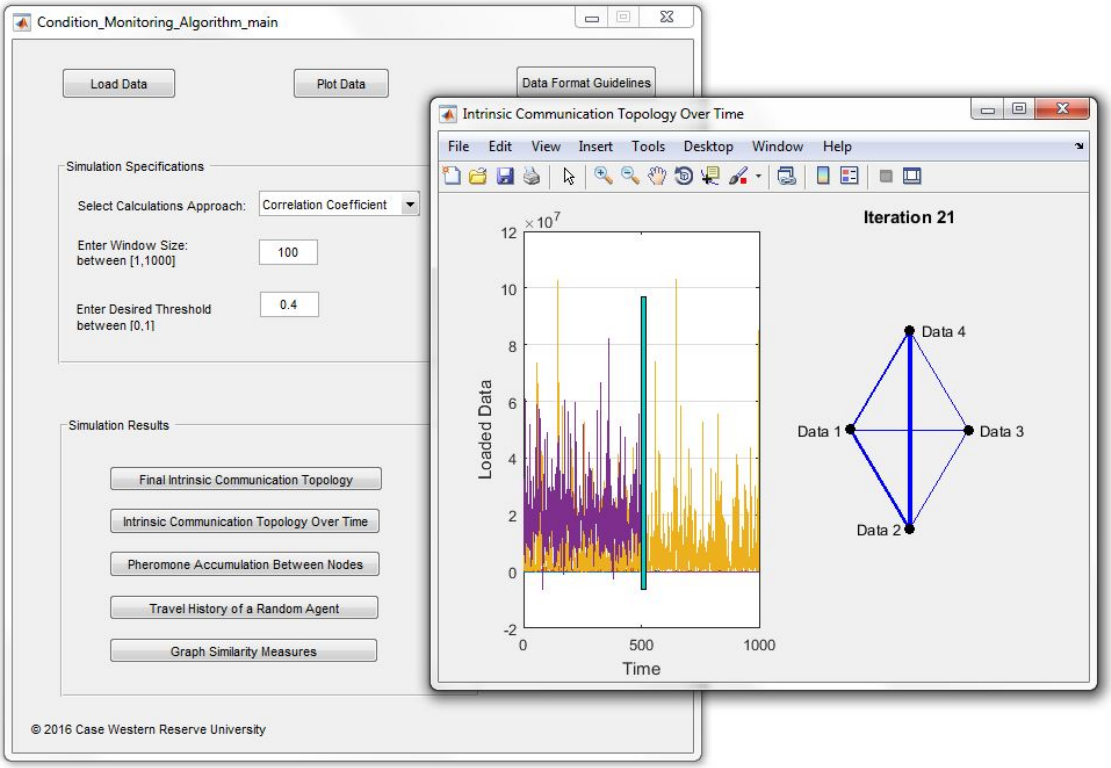


Figure 5.8: Intrinsic Communication Topology Over Time

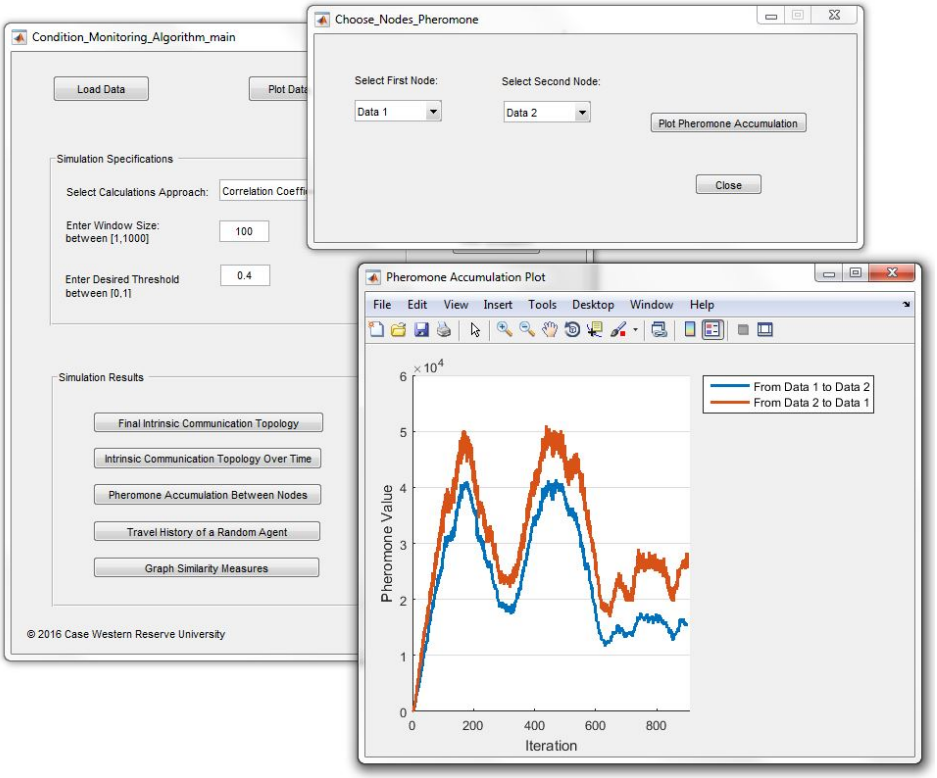


Figure 5.9: Pheromone Accumulation Between Node 1 and 2

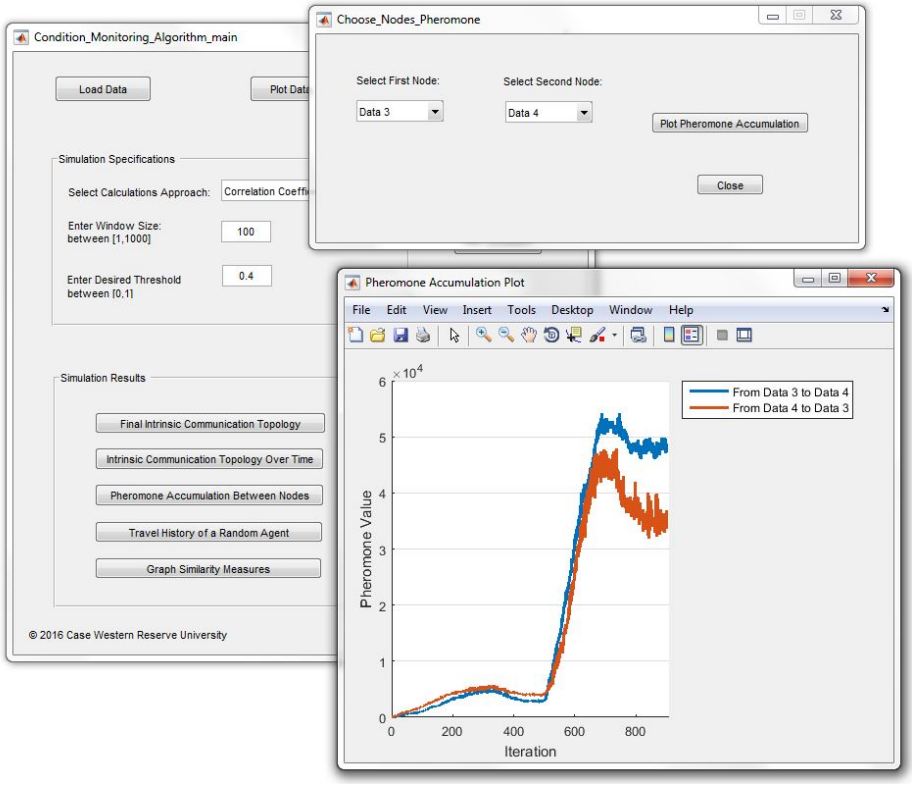


Figure 5.10: Pheromone Accumulation Between Node 3 and 4

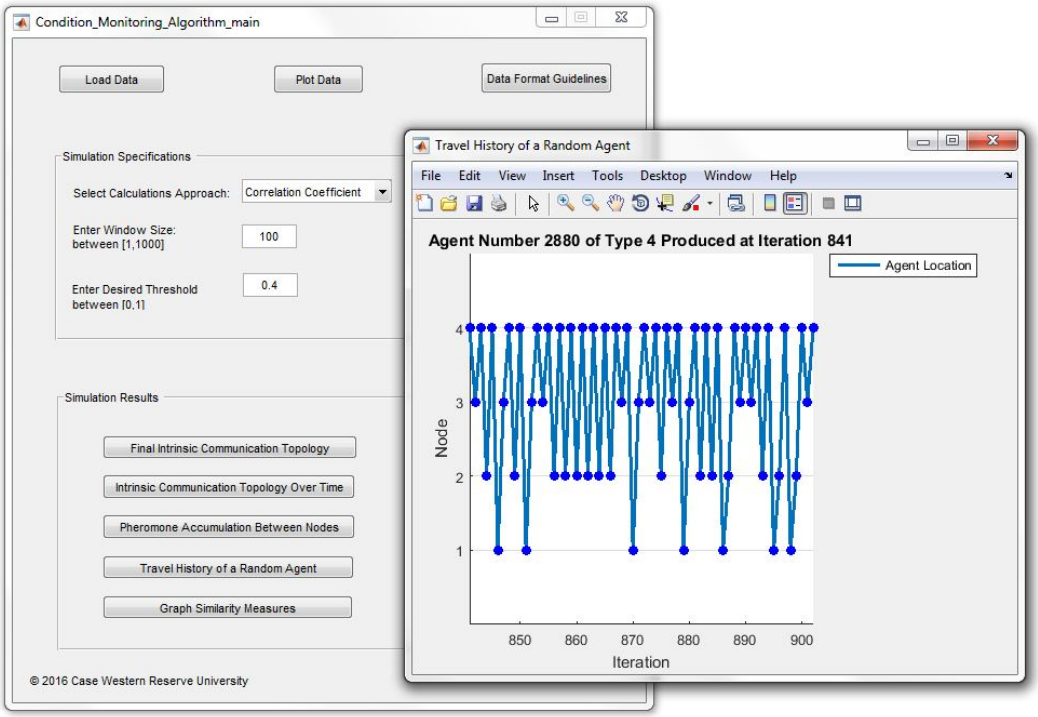


Figure 5.11: Travel History of a Random Agent

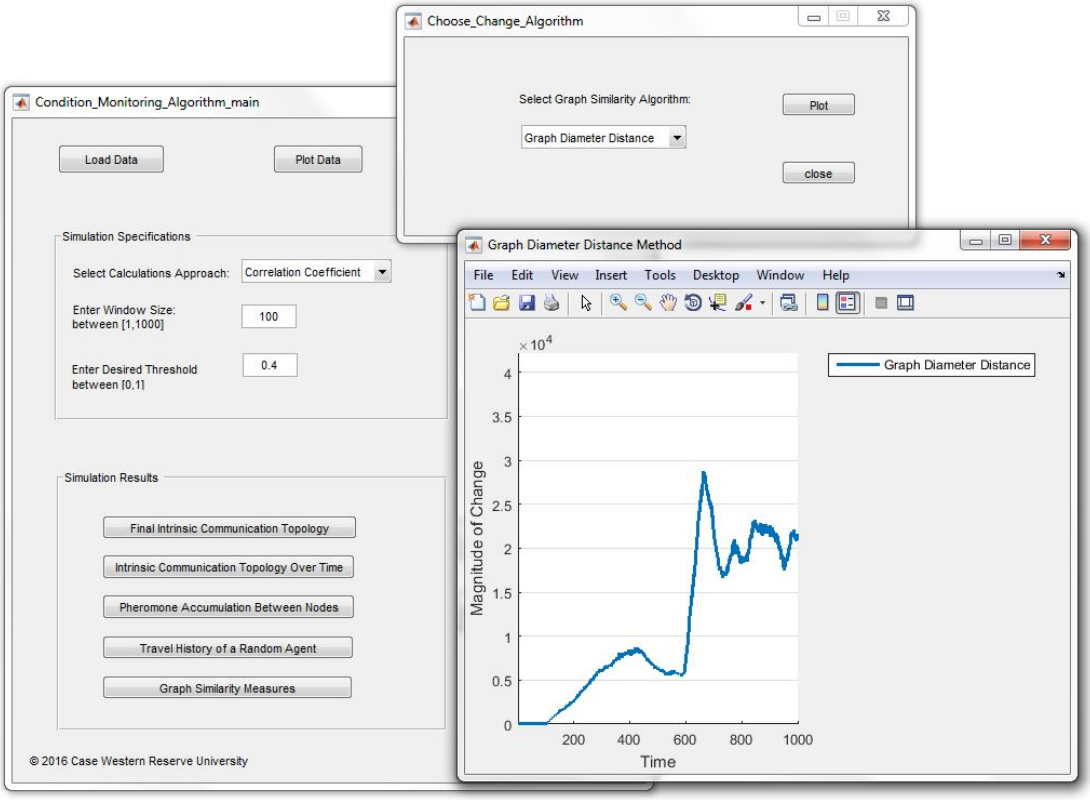


Figure 5.12: Graph Diameter Distance

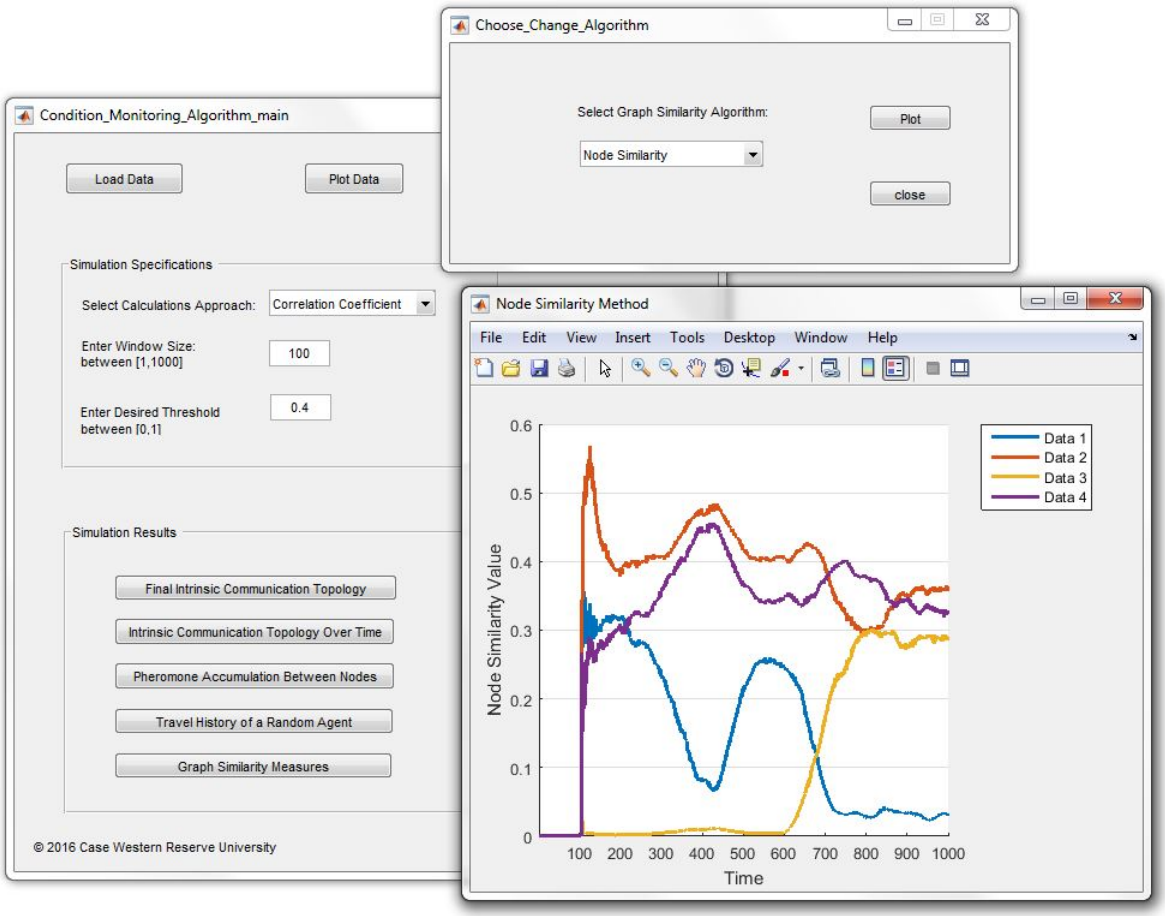


Figure 5.13: Node Similarity Measure

Mutual Information Approach

The simulation is repeated for the same data but with mutual information approach with the same windows size of 100 and desired threshold of 0.4. The results are shown in Figure 5.**14** to Figure 5.21. The simulation takes much longer than correlation coefficient approach. The results can be further improved by changing the desired threshold and window size as discussion in Section 3.0.

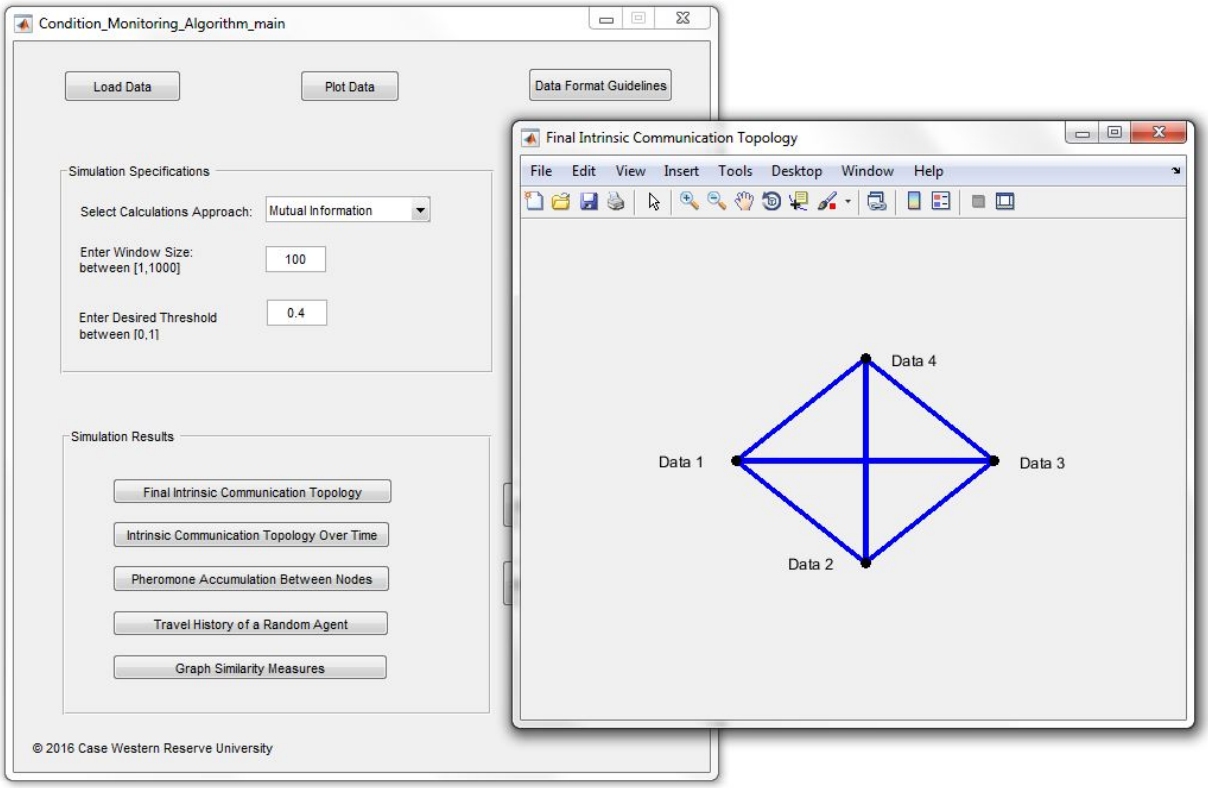


Figure 5.14: Final Intrinsic Communication Topology

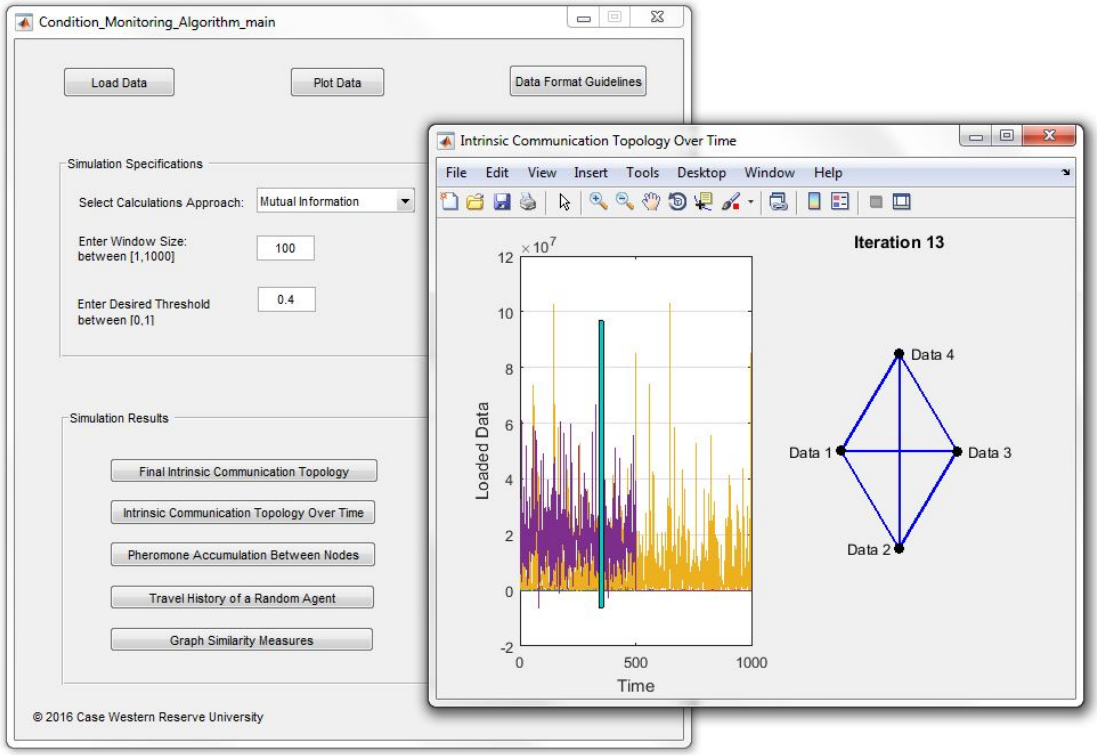


Figure 5.15: Intrinsic Communication Topology Over Time

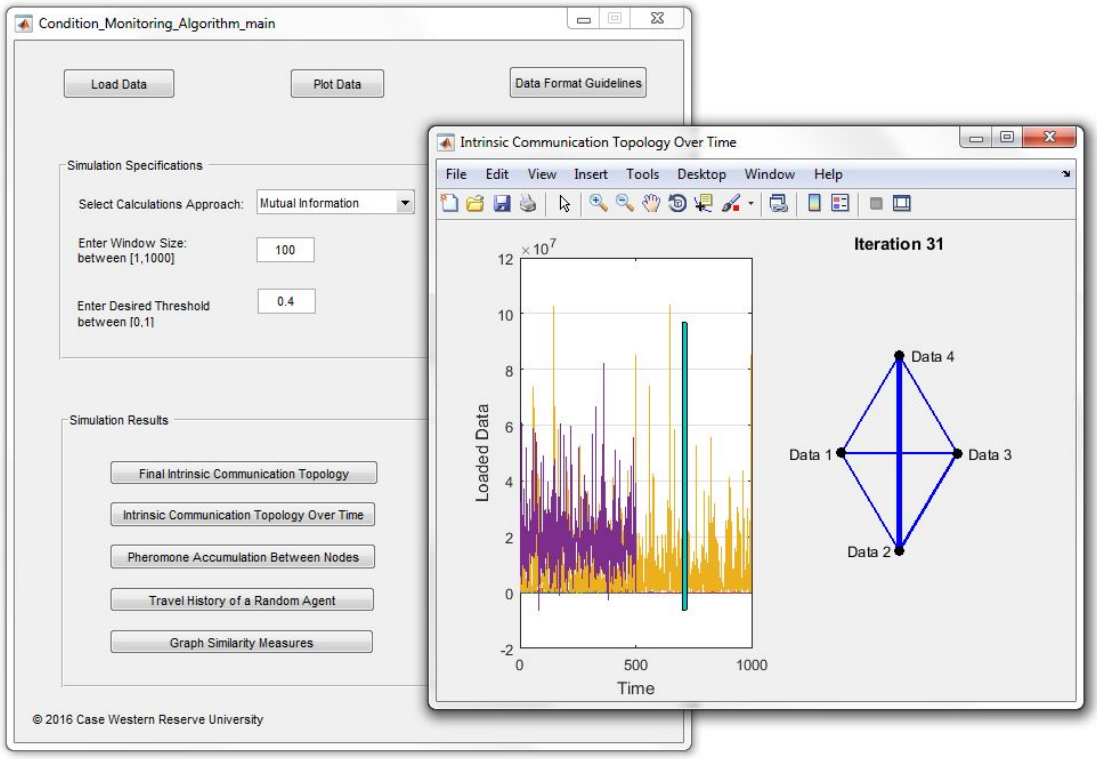


Figure 5.16: Intrinsic Communication Topology Over Time

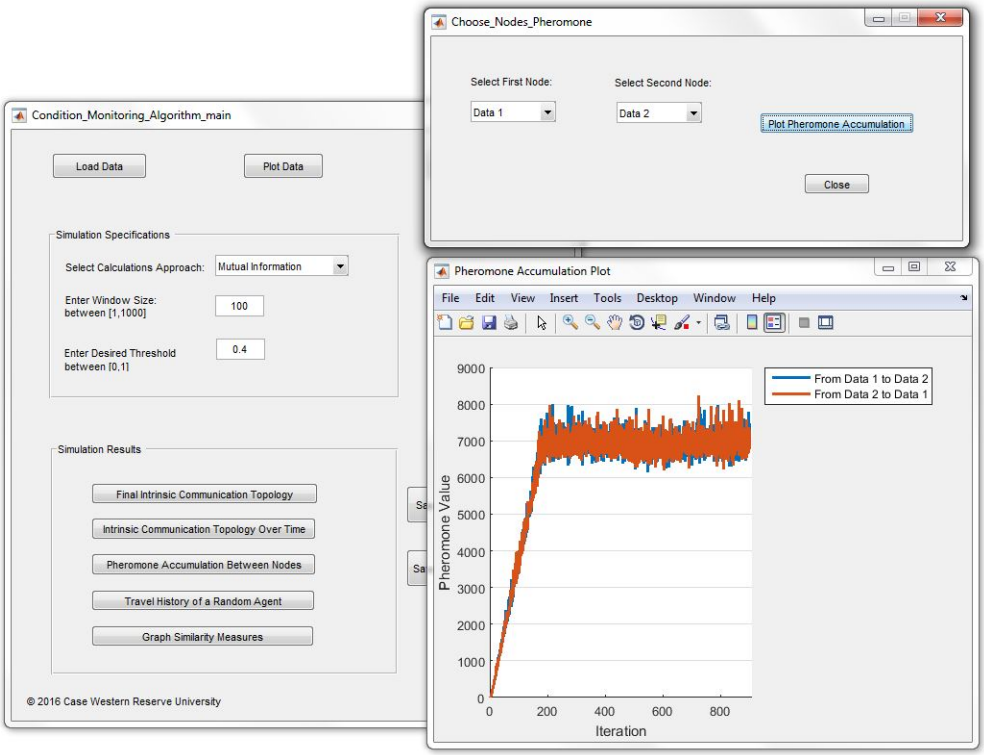


Figure 5.17: Pheromone Accumulation Between Node 1 and 2

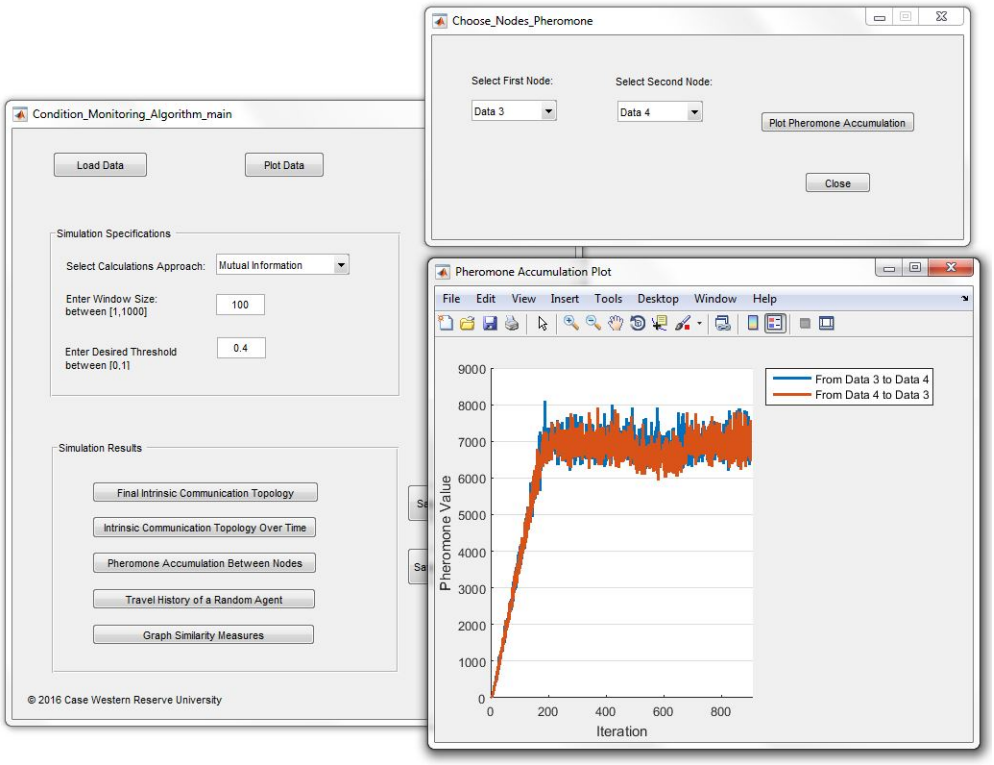


Figure 5.18: Pheromone Accumulation Between Node 3 and 4

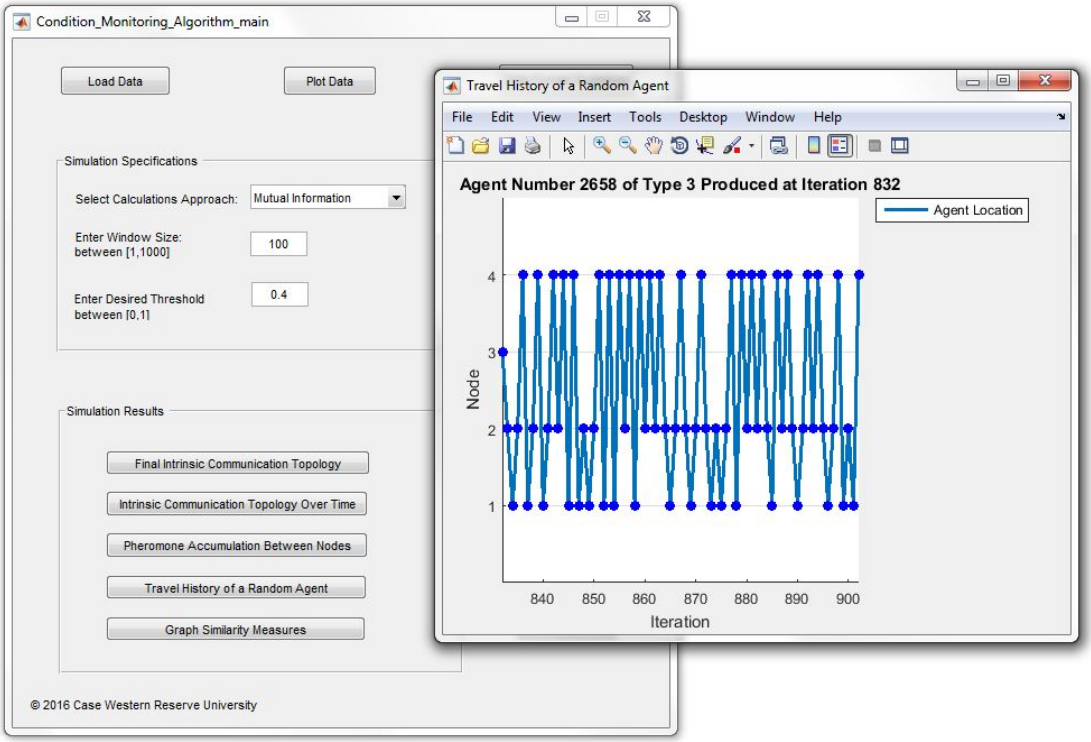


Figure 5.19: Travel History of a Random Agent

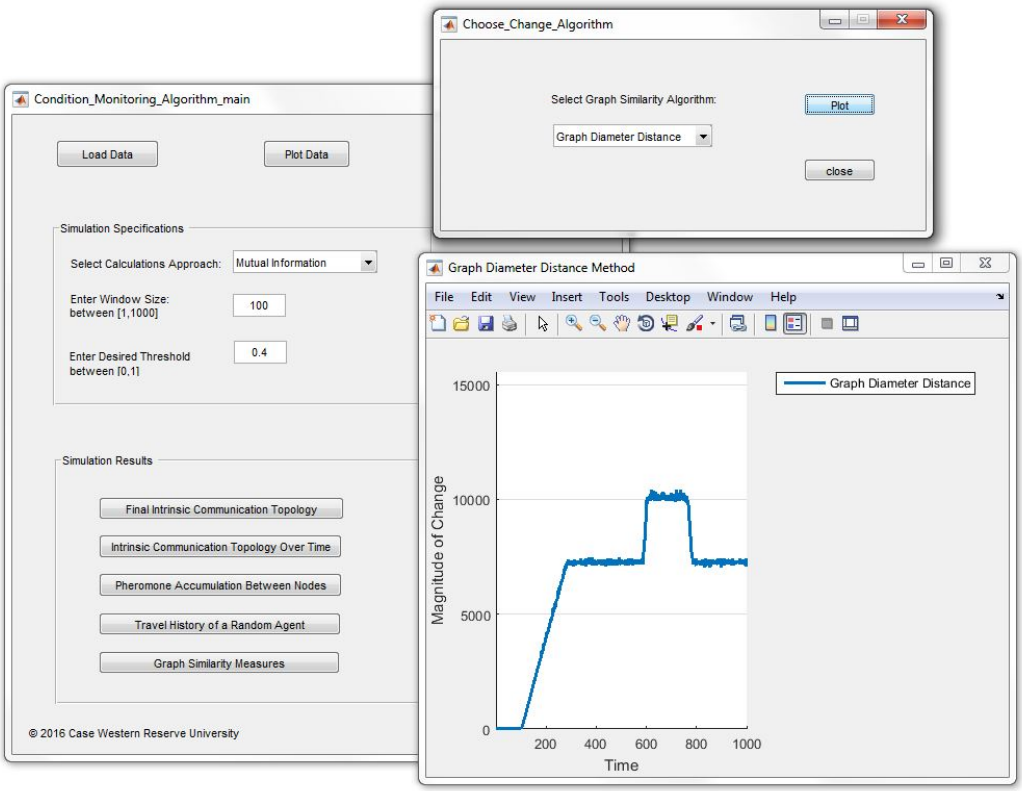


Figure 5.20: Graph Diameter Distance

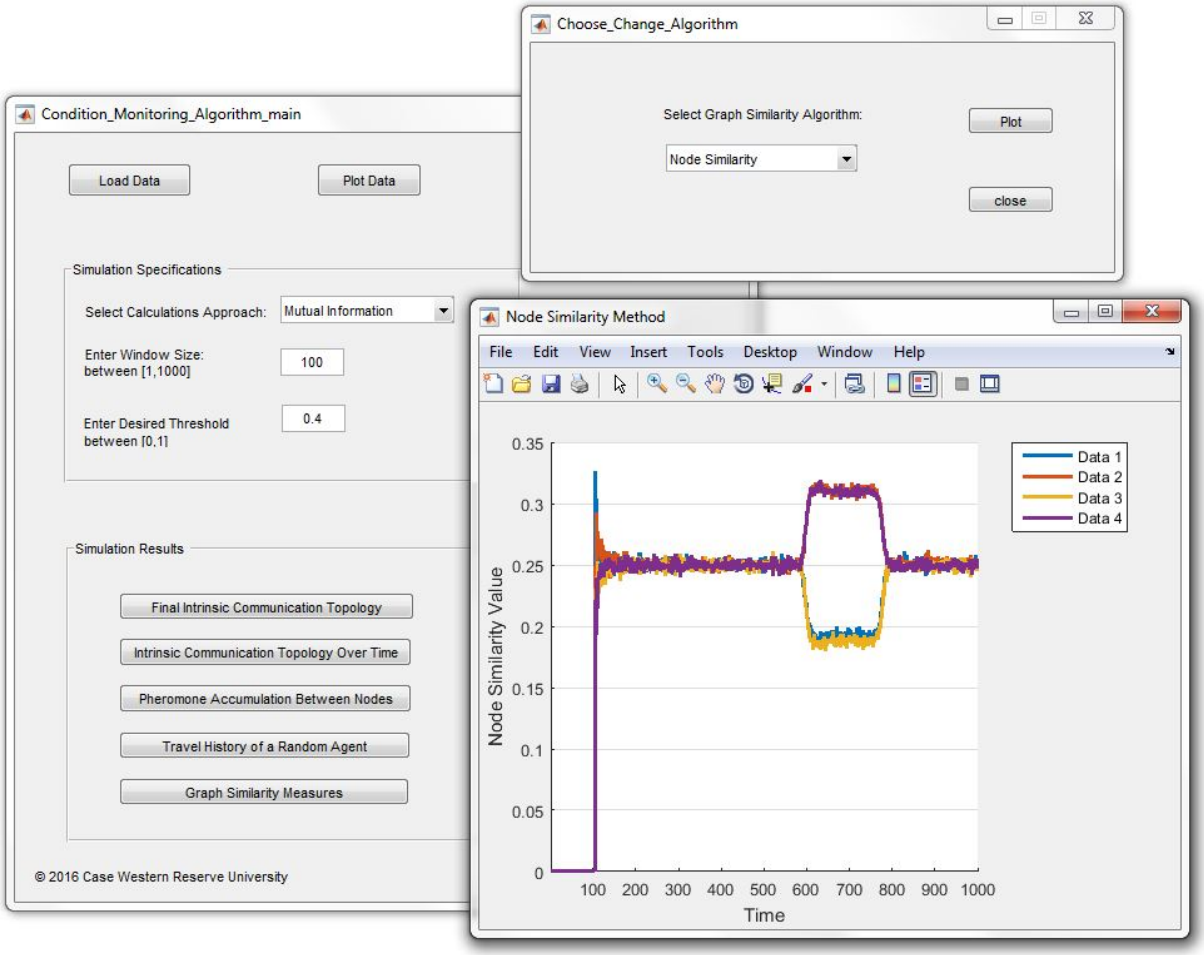


Figure 5.21: Node Similarity Measure

Discovering the Intrinsic Communication Topology of a Physical System

As discussed in Section 3.0, we developed an algorithm that emerged from the foraging behavior of ants to discover the information connectivity between elements of a system. This section extends where the algorithms we developed were applied to Gaussian, Non-Gaussian, and Input/Output power plant simulation data using correlation coefficient and mutual information calculations. We applied graph similarity measures and introduced three change point detection techniques, spectral graph distance, graph diameter distance and node similarity measure, for detecting the changes in the system structure based on the discovered topologies from our algorithm.

Next, we consider simulation data of a steam plant dynamic model in Apros from GE Power (Section 4.0). Different fault scenarios are introduced to the plant simulation model to evaluate the performance of our algorithm in detecting the faults. Three types of faults are considered: process faults, sensor faults and actuator faults. Sensor faults are studied in this report. The boiler faults for the simulation study include: process faults, sensor faults and actuator faults.

Process faults

1. Process items: superheater heat exchangers (SH1, SH2 and SH3)
   Fault type: fouling/slagging in the superheater, which causes gradual change of the heat transfer coefficient.

   Modeling action: To apply a predetermined ramping curve to simulate the gradual change in heat transfer coefficient of the superheater; and record the simulation data for the whole simulation process in the quasi-steady state condition.

   The slow changing rate of the ramping curve should reflect the heat transfer degrading process to some extent, which could be based on field data and operating experiences. Some literature survey can be conducted with support from the university side.

2. Inject Gaussian noise
   a. To add noise to the coal feed rate

Sensor faults

1. Sensor items: temperature sensors for the two stage de-superheaters (SH-DSH1 and SH-DSH2)
   a) Fault type: sensor bias of various levels (both negative and positive)
      Modeling action:
      - Apply bias to the temperature measurement. The bias applied to the model should reflect the field data to some extent. The fault levels will be varied to simulate the variation in sensor fault severity
      - Apply graduate changes to a sensor output to simulate the sensor drift fault. The fault levels will be varied to simulate the variation in sensor fault severity

   b) Fault Type: Simulate sensor performance degrading by increasing the SNR (Signal to Noise Ratio). Different levels of SNR can be planned for fault simulations.

      Measurement noise should be introduced by adding Gaussian noise signals.

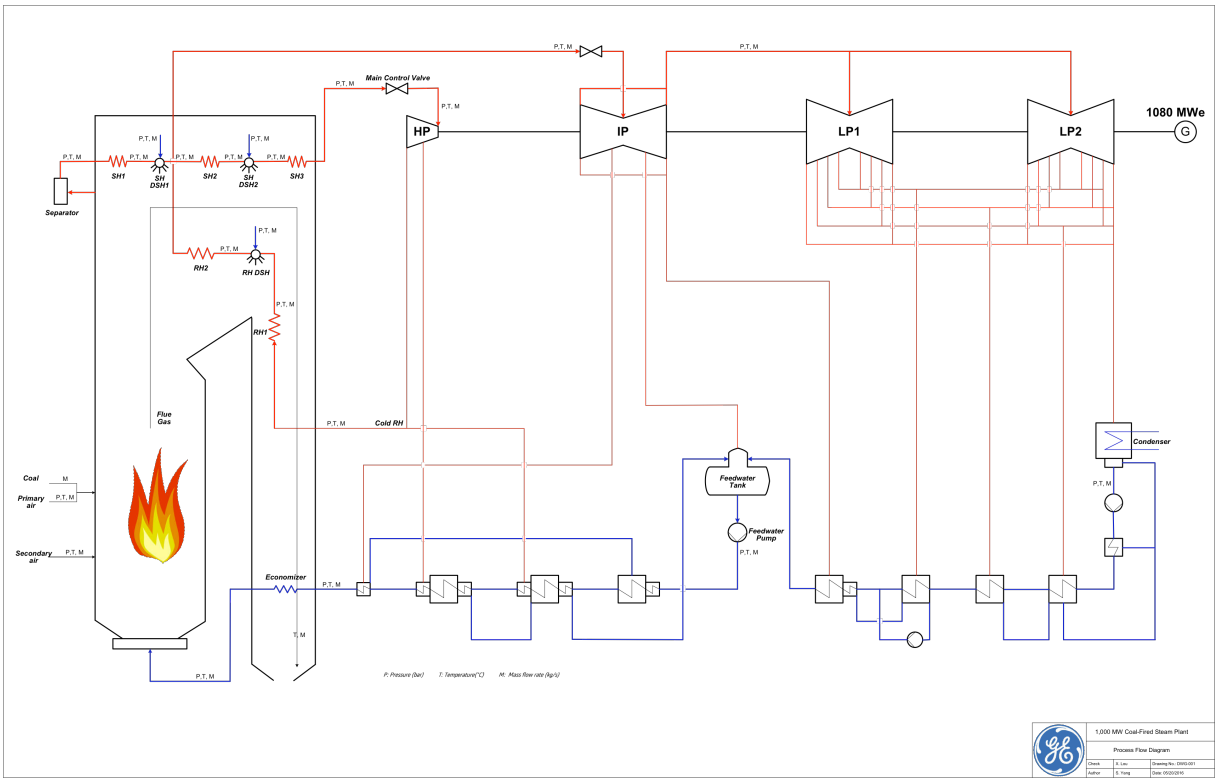
   c) Fault type: out-of range sensor failure
      Modeling action: Apply bias to the temperature measurement to exceed the bound of the sensor measurement.

Actuator faults

1. Actuator items: water spray valve actuator for the two stage de-superheater (SH-DSH1 and SH-DSH2)
   a) Fault type: actuator bias of various levels (both negative and positive)
      Modeling action: apply bias to the position of the water spray valves. The bias applied to the model should reflect the field data to some extent.

   b) Fault type: Actuator saturation (out-of range actuator failure)
      Modeling action: apply bias to the position of the water spray valves to exceed the maximum position of the valves.

A process diagram of the boiler being considered in this study is shown below.



Sensor Faults

The first step was to develop a list of possible fault scenarios that could be simulated and used to further develop and validate the condition. Temperature sensors for primary and final de-superheaters are subject to a bias fault. A positive or negative bias is applied to the temperature measurement. In this report, we consider positive and negative step bias to the temperature sensor of the primary de-superheater, SH-DSH1, as listed in Table 5.13.

Table 5.13: Sensor Fault Scenarios

Sensor	Bias	Time when bias is applied
SH-DSH1 Temperature Sensor	step bias of positive 5 c	30 mins
	step bias of negative 5 c	30 mins

Process variables, controller outputs and setpoints for the outer loop of the main steam controllers in the boiler model are recorded every 2 seconds for 2 hours. The tags are listed in Table 5.14.

Table 5.14: Taglist

Taglist	Description
SHDSH1_inlet	Primary desuperheater (SHDSH1) inlet
SHDSH1_outlet	Primary desuperheater (SHDSH1) outlet
SHDSH2_inlet	Final desuperheater (SHDSH2) inlet

SHDSH2_outlet	Final desuperheater (SHDSH2) outlet
SHDSH1_mass_flow	Primary desuperheater (SHDSH1) spray water mass flow
SHDSH2_mass_flow	Final desuperheater (SHDSH2) spray water mass flow
Primary_master_setpoint	Master controller setpoint for primary desuperheater (SHDSH1)
Primary_master_output	Master controller output for primary desuperheater (SHDSH1)
Primary_slave_output	Slave controller output for primary desuperheater (SHDSH1)
Final_master_setpoint	Master controller setpoint for final desuperheater (SHDSH2)
Final_master_output	Master controller output for final desuperheater (SHDSH2)
Final_slave_output	Slave controller output for final desuperheater (SHDSH2)

Positive Step Bias Simulation Data

The primary master temperature sensor output is shown in Figure 5.**22**. We can see that a step bias of positive 5^{c} is applied to this sensor at time 30 minutes. Figure 5.**23** to Figure 5.**31** show the simulation data for process variables, controller outputs and setpoints. We can see the change in the temperature, mass flow and the set points after the bias is applied. However, after a while the system settles and returns to its previous state. We evaluate the performance of our algorithm in detecting the bias fault.

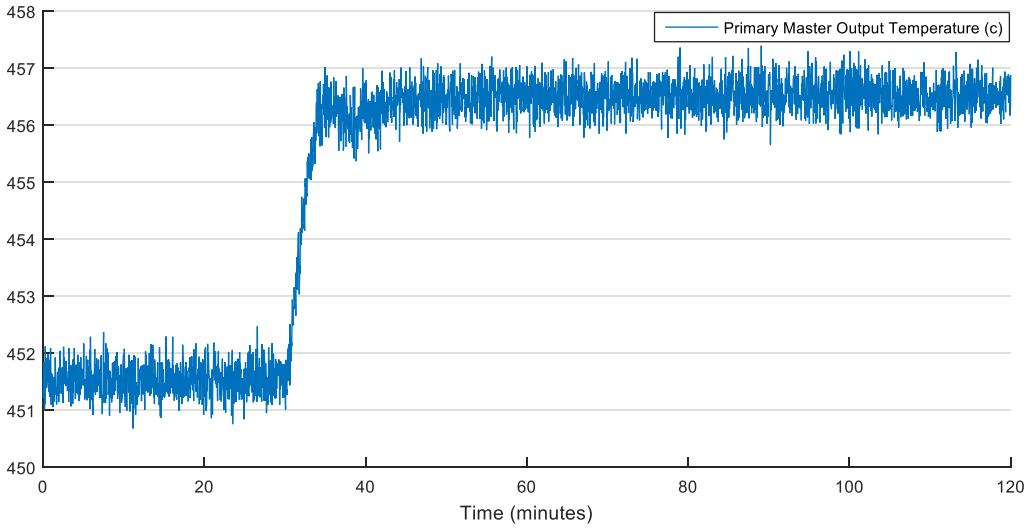


Figure 5.22: Primary Master Output Temperature

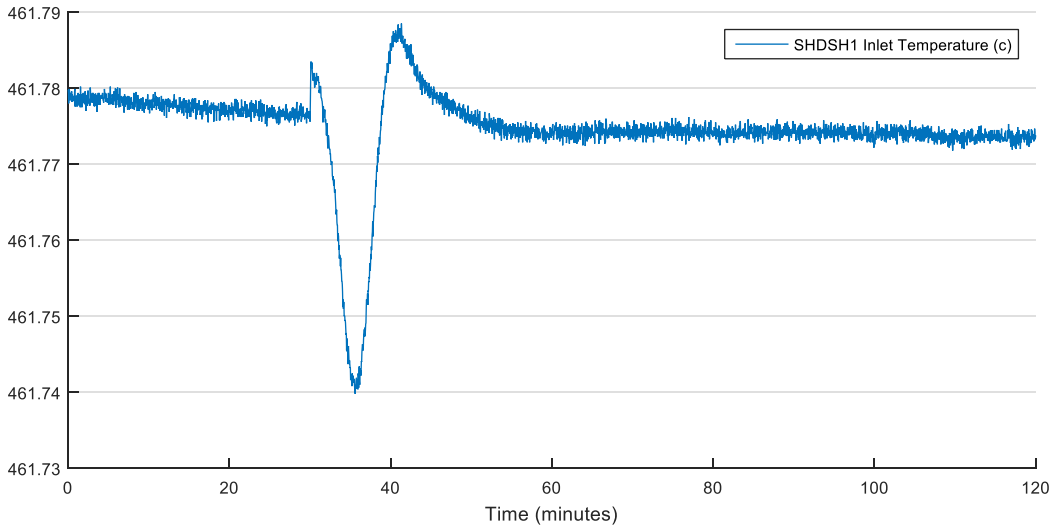


Figure 5.23: SHDSH1 Inlet Temperature

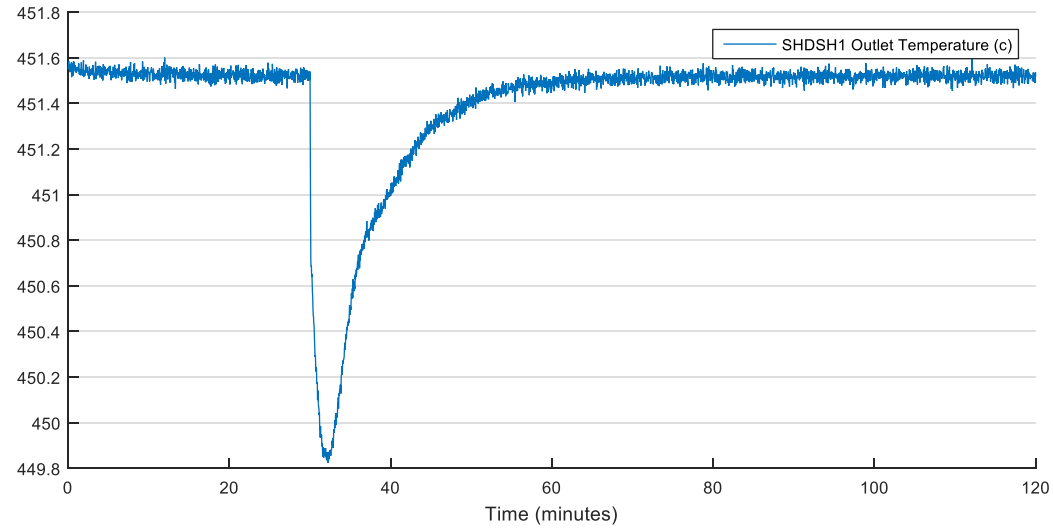


Figure 5.24: SHDSH1 Outlet Temperature

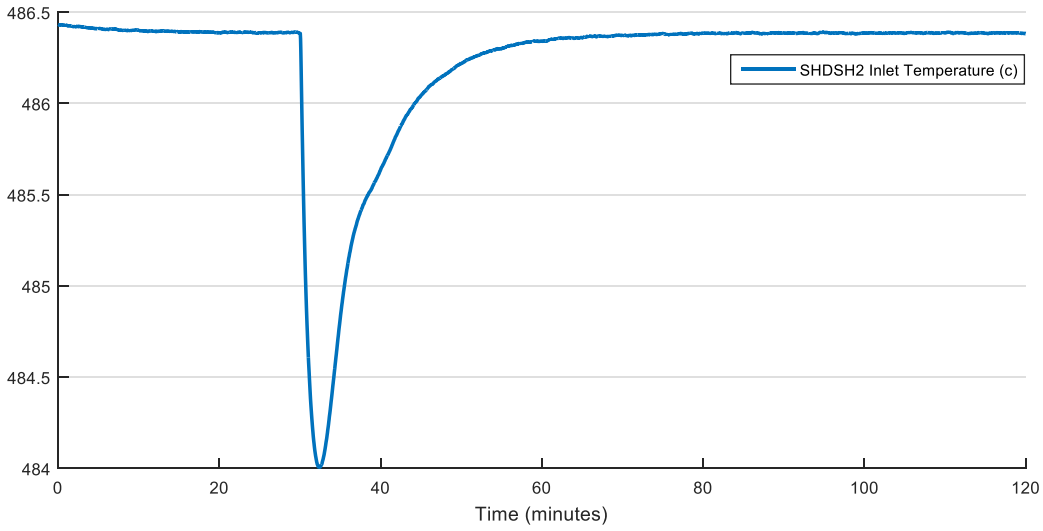


Figure 5.25: SHDSH2 Inlet Temperature

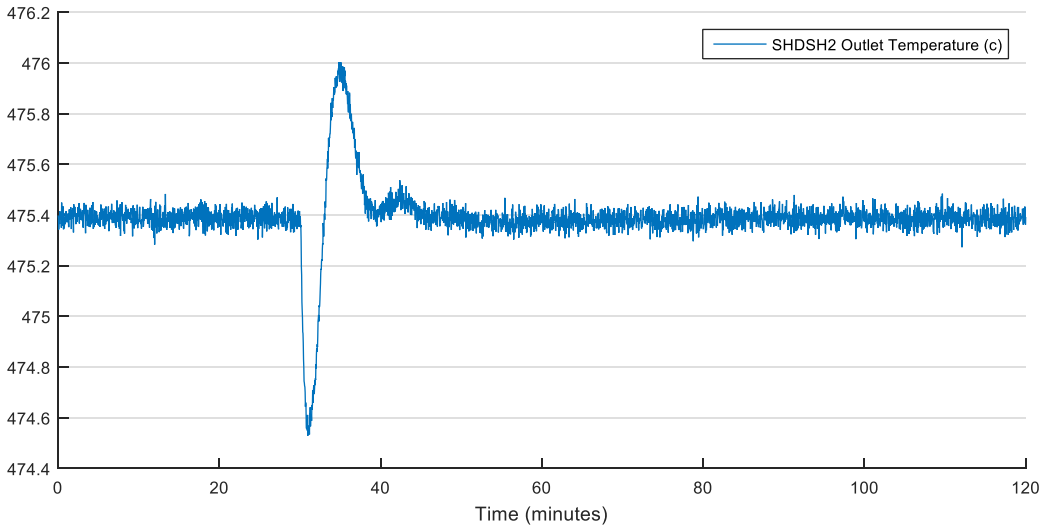


Figure 5.26: SHDSH2 Outlet Temperature

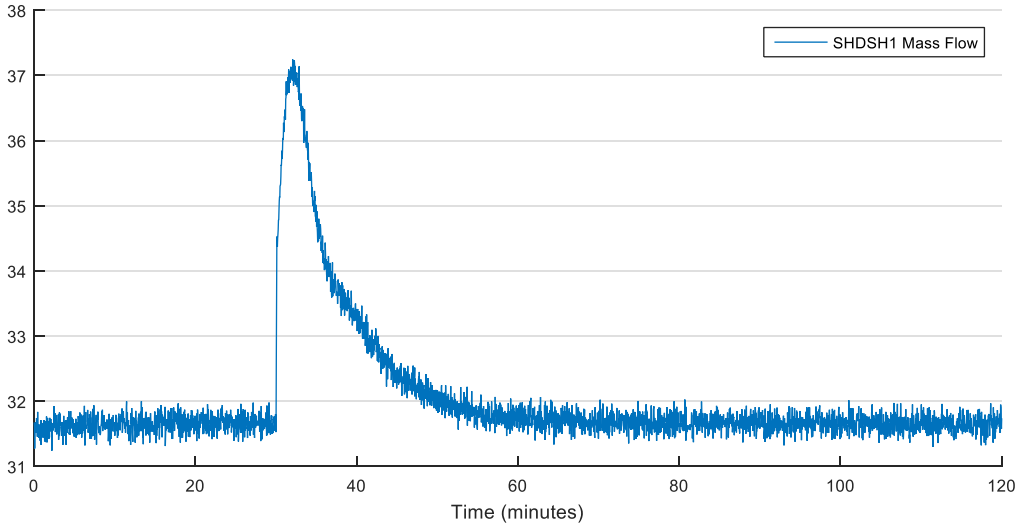


Figure 5.27: SHDSH1 Mass Flow

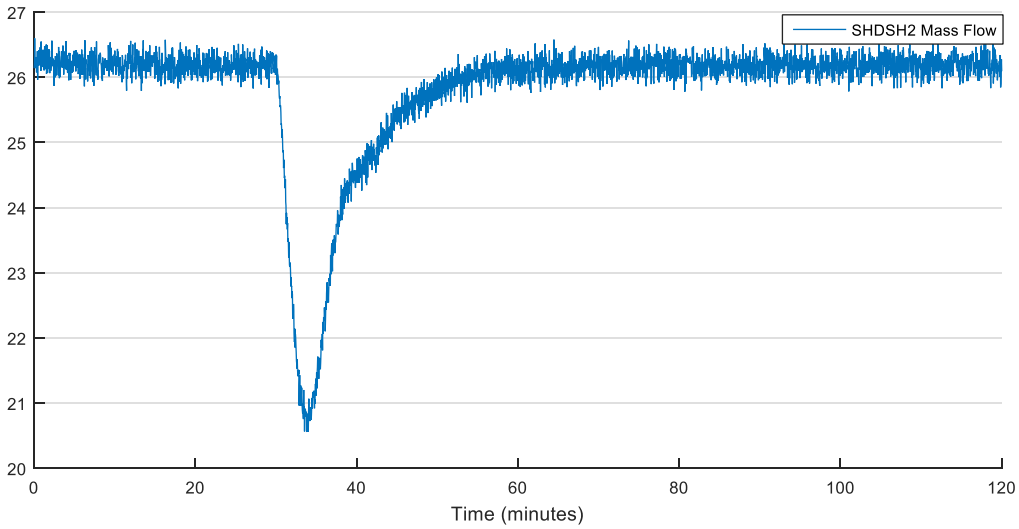


Figure 5.28: SHDSH2 Mass Flow

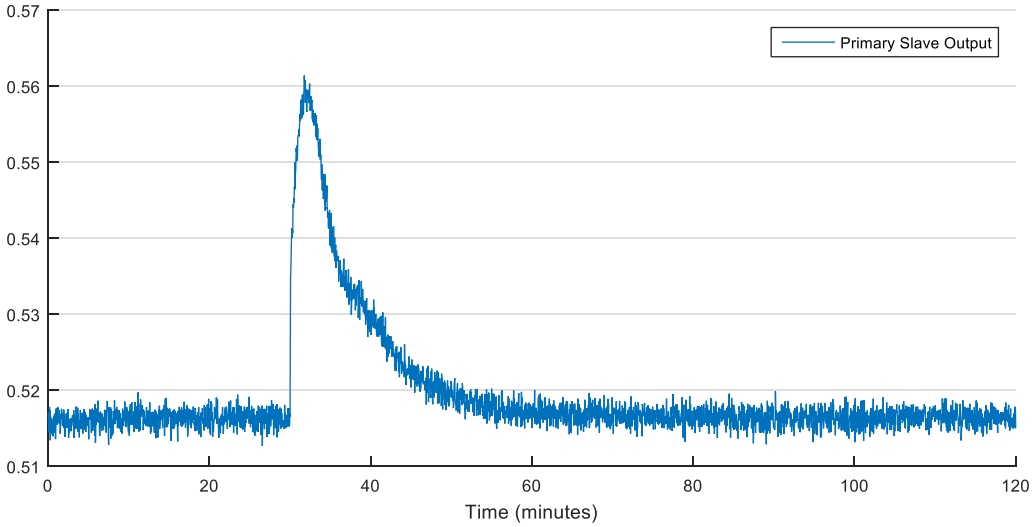


Figure 5.29: Primary Slave Output

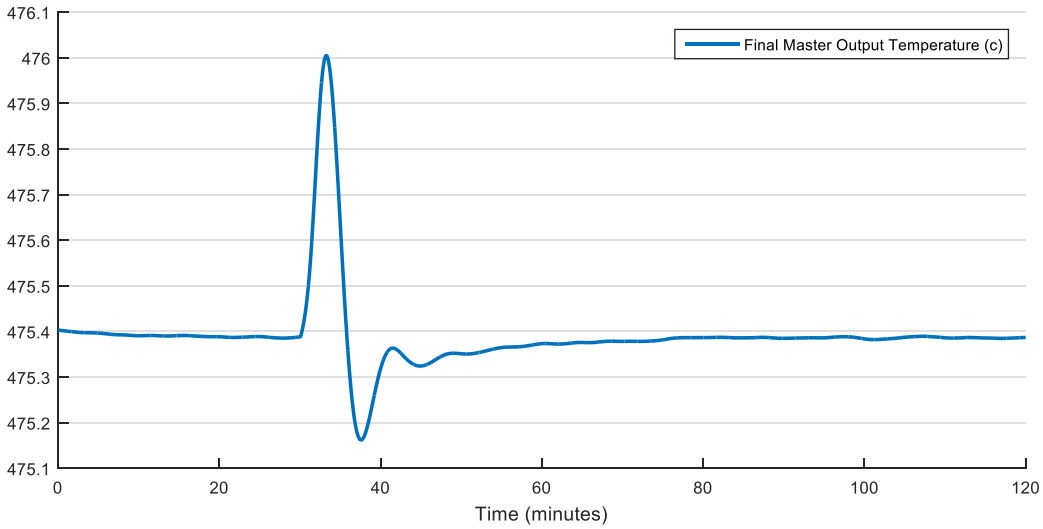


Figure 5.30: Final Master Output Temperature

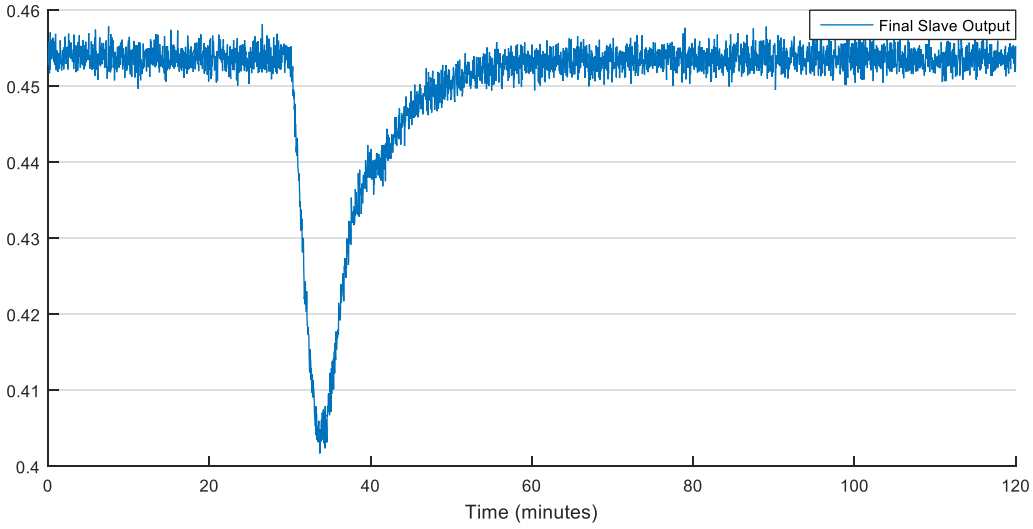


Figure 5.31: Final Slave Output

Simulation Results: The steam plant model is considered as a network of 12 nodes, as listed in **Table 5.14**. The simulation data is applied to the network. The intrinsic communication topologies are then extracted from the network over different time windows and the results are passed through the change detection algorithm to detect the fault.

Extracted Intrinsic Communication Topology: The simulation was initiated with 13 agents at each node with lifespan of 50 for a total of 156 agents over the entire network. The agents traverse the network and discover the intrinsic communication topology of the power plant. At the end of the simulation, there are approximately 78,000 agents traveling throughout the network. The simulation is run with no initial pheromone present and a pheromone decay rate of 0.2 units/iteration. A sliding window with a length of 50 is passed over the data stream and the intrinsic communication topology for each window of data is discovered.

Figure 5.**32** to Figure 5.**37** show the information connectivity structures for different time windows. The normalized simulation data is shown on the left side of these figures with the extracted connectivity structure on the right side. The thickness of the line between each two nodes in the discovered topology is proportional to the communication strength between its terminal nodes. The system starts at a stable connectivity structure as shown in Figure 5.**32**. After the introduction of the sensor bias at time 30 minutes, temporary communication links are appearing between the nodes. After around 40 minutes, the information connectivity structure converges to its stable state as shown in Figure 5.**36**.

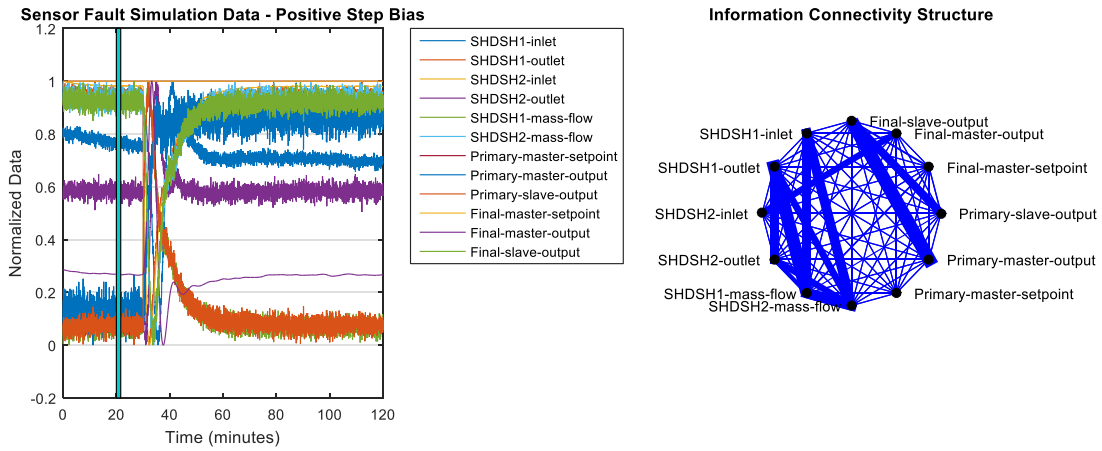


Figure 5.32: Stable Information Connectivity Structure

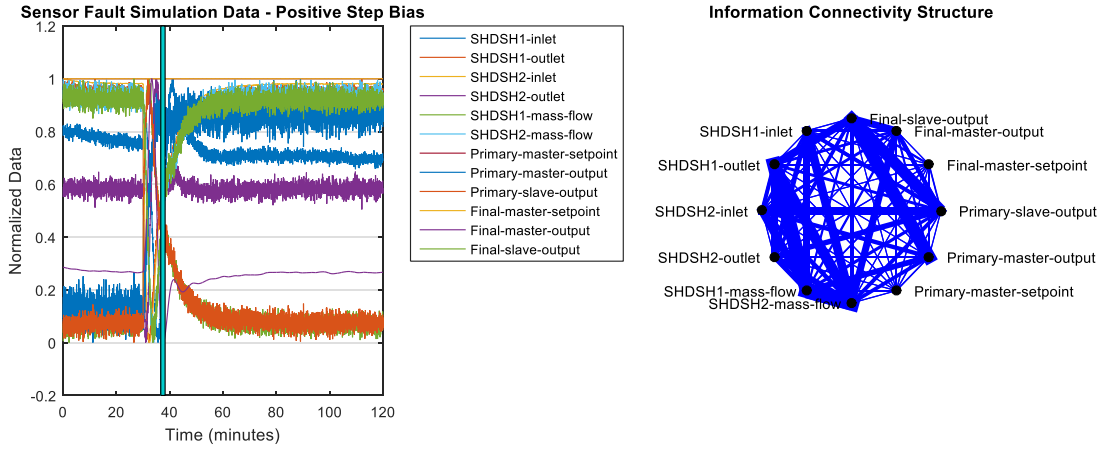


Figure 5.33: Transient Information Connectivity Structure

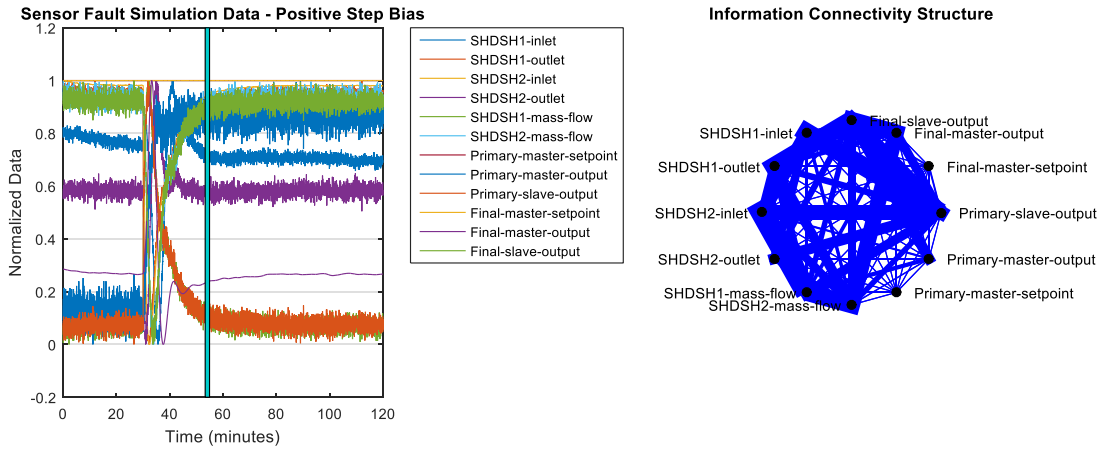


Figure 5.34: Transient Information Connectivity Structure

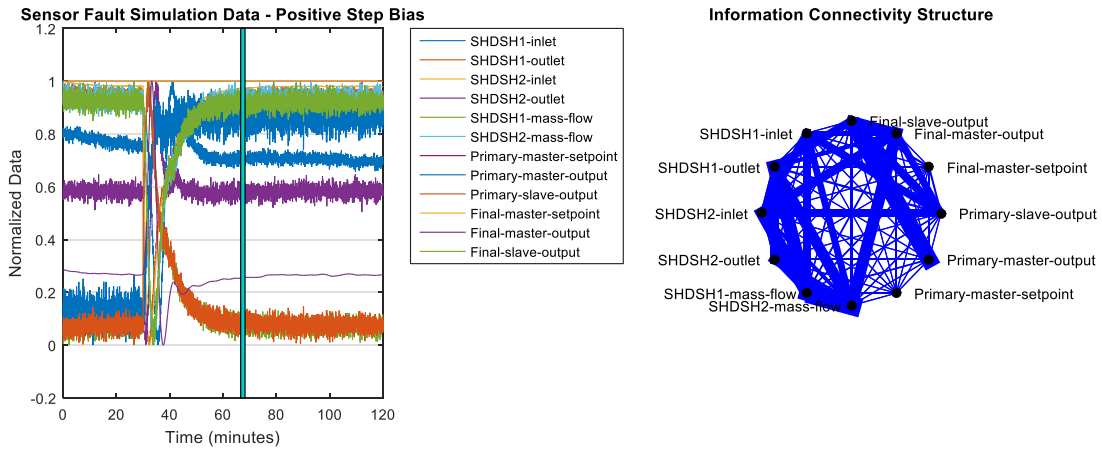


Figure 5.35: Transient Information Connectivity Structure

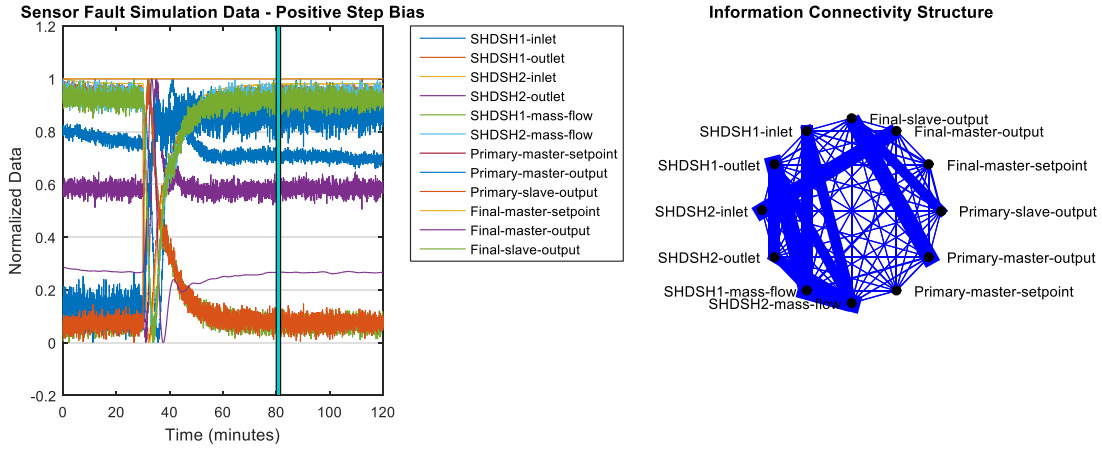


Figure 5.36: Stable Information Connectivity Structure

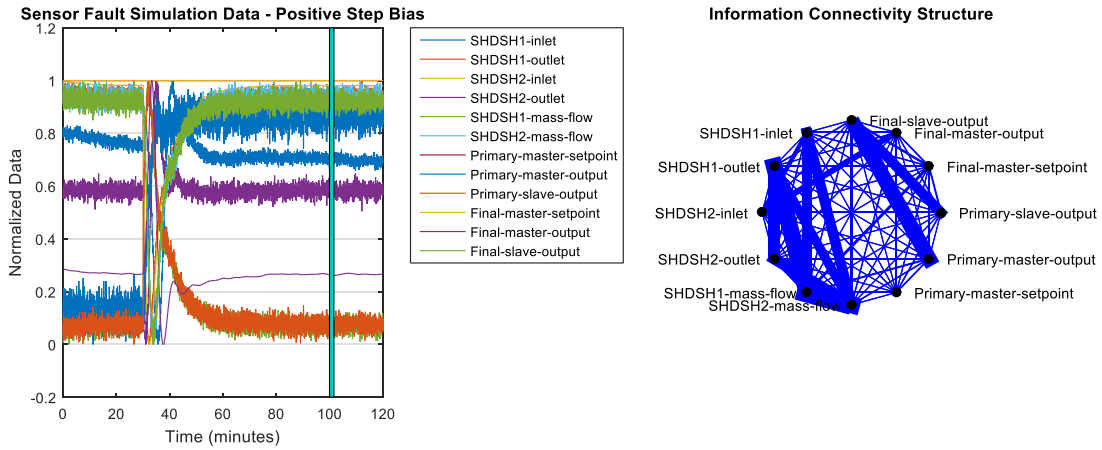


Figure 5.37: Stable Information Connectivity Structure

Introducing a bias to the output of the sensor does not change the system structure, so there are no structural changes in the extracted topologies. Therefore, the connectivity structures of Figure 5.**32** to Figure 5.**37** are the same. This structure is shown in

Figure 5.**38** and for the sake of clarity, connections whose strengths are less than 30% of the strongest connection in the topology are omitted from the graph.

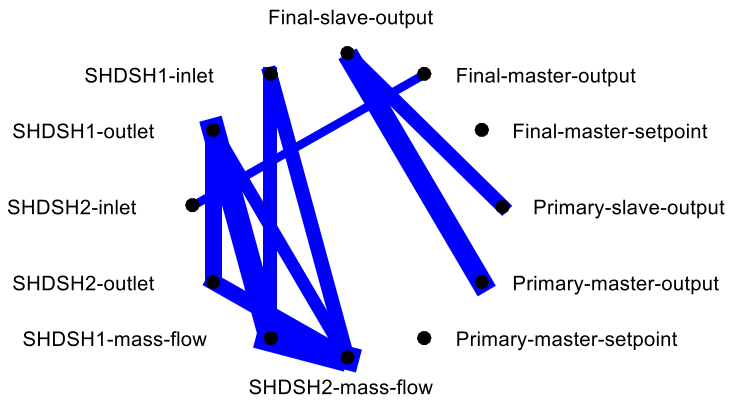


Figure 5.38: Stable Information Connectivity Structure

Change Detection: The *Node Similarity Measure* is applied to the extracted communication topologies to detect the point where the fault happens. The node similarity measure for a positive step bias is shown in Figure 5.39. The sudden change in the node similarity scores indicates a change in the operating state of the system. As the connectivity structures for stable states are similar, node similarity scores do not change that much. Therefore, we can detect changes in the operating state by looking at sudden changes in the similarity scores that are related to changes in the communication structures, i.e. the presence of the fault.

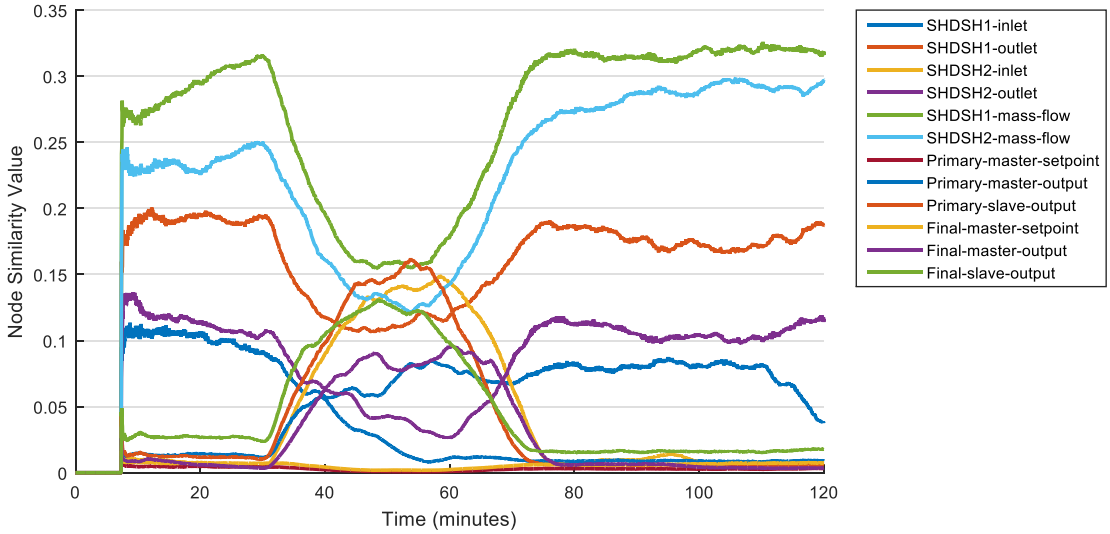


Figure 5.39: Node Similarity Measures

From Figure 5.**39** we can see that around the time 30 minutes where the bias is applied to the sensor, the similarity values of almost all of the nodes change. This indicates a

change in the structure of the system, and therefore a change in the status of the system. After around 40 minutes, the system converges to a stable operating condition and the node similarity scores return to their previous values. This shows that the connectivity structure also converges to its previous stable structure.

Negative Step Bias Simulation Data

Next, we consider simulation data for a negative step bias in the temperature sensor of the master controller output for the primary de-superheater. The normalized simulation data for negative bias is shown in Figure 5.**40**. We can see similar behavior for the negative bias as well: the temperature, mass flow and the setpoints change after the bias is applied. However, after about 40 minutes, the system settles down and returns to its previous state.

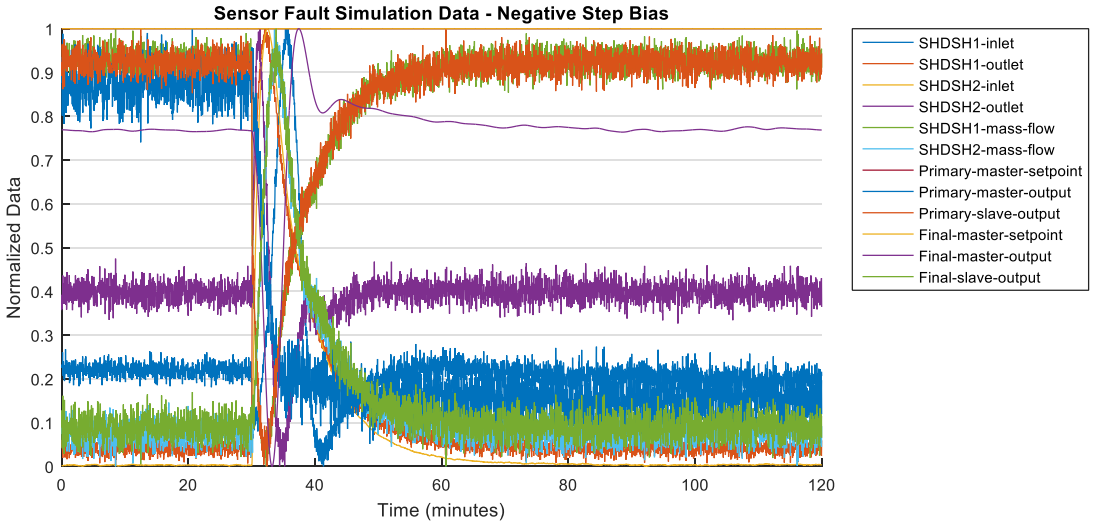


Figure 5.40: Normalized Negative Step Bias Simulation Data

Simulation Results: Similarly, the steam model is considered as a 12-node network. The simulation data is applied to the network and the intrinsic communication topologies are then extracted from the network over different time windows and the results are passed through the change detection algorithm to detect the fault.

Extracted Intrinsic Communication Topology: The simulation specifications are the same as with the positive step bias fault. The extracted information connectivity structures for stable and transient states are shown in Figure 5.41 to Figure 5.44. We can see that after the bias is applied to the sensor, the connectivity structure changes. After about 40 minutes, the system settles and its connectivity structure converges to its stable state. This stable state is shown in Figure 5.45.

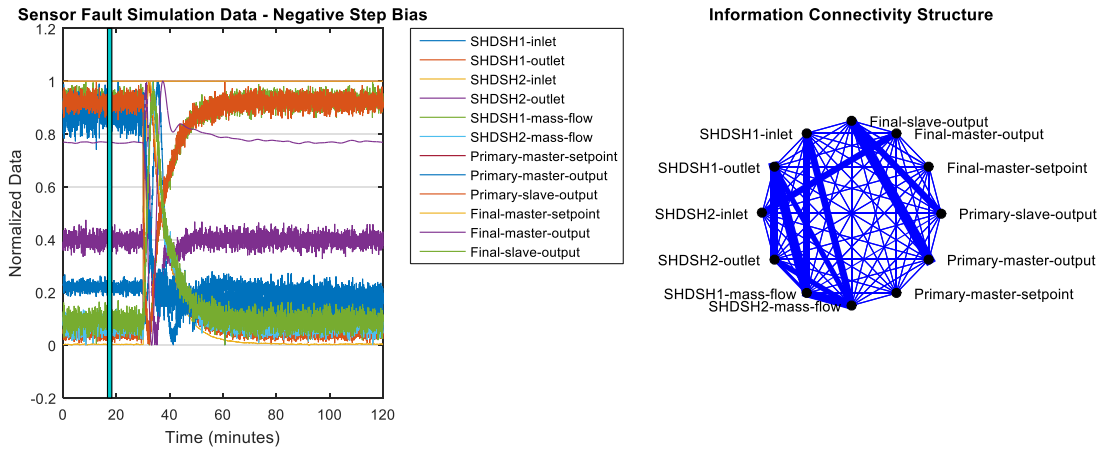


Figure 5.41: Stable Information Connectivity Structure

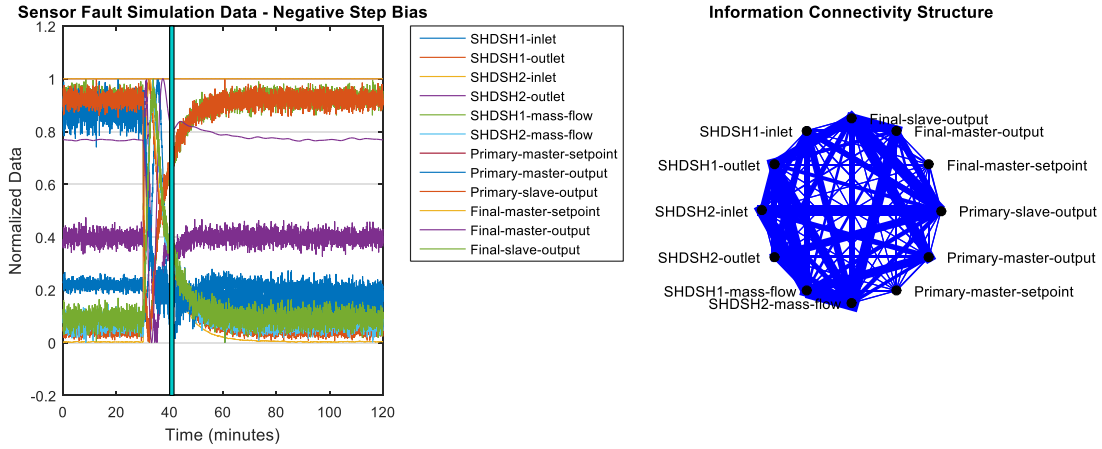


Figure 5.42: Transient Information Connectivity Structure

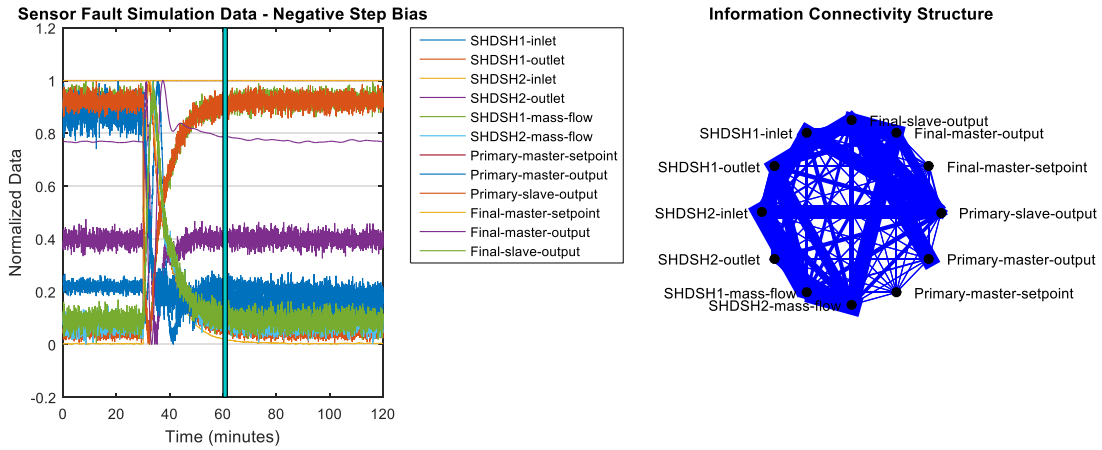


Figure 5.43: Transient Information Connectivity Structure

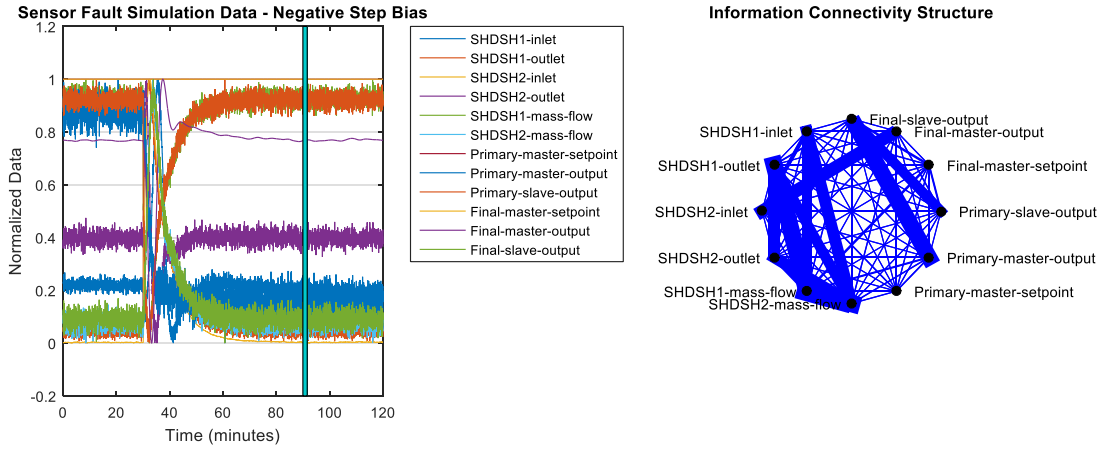


Figure 5.44: Stable Information Connectivity Structure

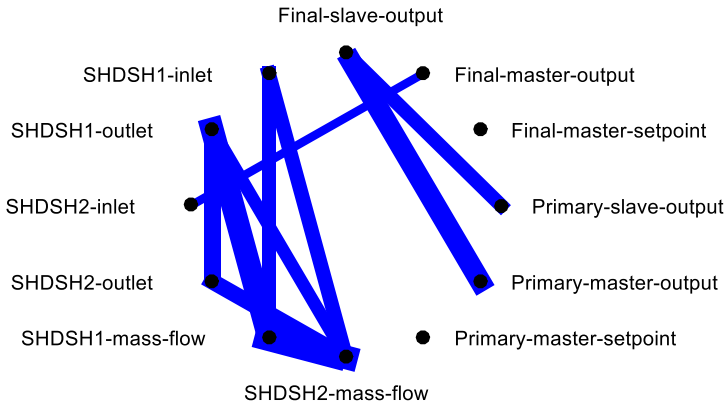


Figure 5.45: Stable Information Connectivity Structure

Change Detection: Node similarity measures for the extracted communication topologies are used to detect the presence of the fault as shown in Figure 5.**46**. We can see that when the fault happens, the node similarity score for almost all of the nodes changes. As the system converges to its stable state, the similarity scores return to their previous values.

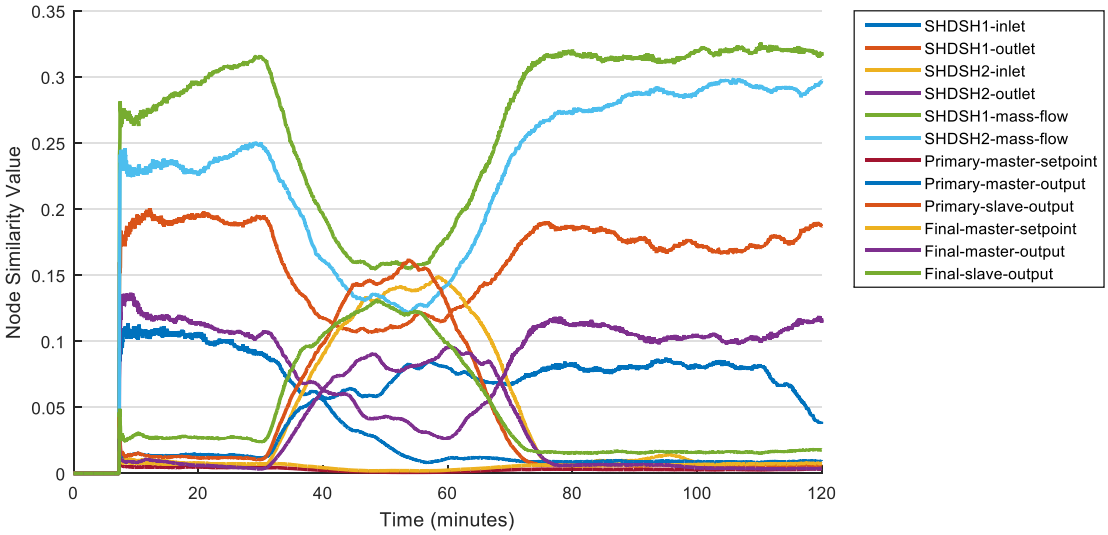


Figure 5.46: Node Similarity Measure

In this preliminary study,, we considered sensor faults in a steam power plant model. Our algorithm was used to analyze the faulty simulation data to detect the presence of a fault by extracting the intrinsic communication topology of the steam plant model and observing the changes in the communications topology. When the fault occurs in the system, the communication topology changes and the fault is detected by observing the changes in the intrinsic communications topology using a change detection algorithm such as the node similarity measure.

The algorithm was used to process to simulation output data and the communication topology of the system was identified. The communication topology of the system in the normal (or baseline) operational state was identified, and we observed that almost immediately after the step bias fault was introduced to the system, the topology computed by the algorithm changes. However, after a while, the communication topology converged back to its normal mode topology. The reason is that applying a bias to the output of the sensor does not change the internal system structure. We used a node similarity measure to detect the point where the fault occurred, and also to determine when the system settles down after the transient introduced by the fault.

Fault Simulation and Fault Detection Algorithm Testing

We continue the data analysis of the previous report section using simulation data of a steam plant dynamic model in Apros from GE Power (See Section 4.0). Different fault scenarios are introduced to the plant simulation model to evaluate the performance of our algorithm in detecting the faults. Three types of faults are considered: process faults, sensor faults and actuator faults. Sensor fault was studied in the previous report. Actuator and process faults are considered in this report. Also, we evaluate the performance of our algorithm in the case of multiple faults that occur in a single simulation run.

Single Fault Scenarios:

Actuator Fault: The actuator for the final desuperheater valve is subject to positive/negative bias or saturation faults as listed in Table 5.15.

Table 5.15: Actuator Fault Scenarios

Actuator	Bias	Time when bias is applied	Ramp Duration
Actuator for final desuperheater (SH-DSH2) valve	step bias of positive 0.05	30 minutes	N/A
	step bias of negative 0.05	30 minutes	N/A
	ramp bias of positive 0.05	2 hours	6 hours
	ramp bias of negative 0.05	2 hours	6 hours
	ramp bias of positive 0.5 to reach lower bound of actuator	2 hours	6 hours

Process variables, controller outputs and setpoints for the outer loop of the main steam controllers in the boiler model are recorded every 5 seconds. The tags are listed in Table 5.16.

Table 5.16: Actuator Faults Taglist

Taglist	Description
Primary_master_setpoint	Master controller setpoint for primary desuperheater (SHDSH1)
Primary_master_output	Master controller output for primary desuperheater (SHDSH1)
Primary_slave_output	Slave controller output for primary desuperheater (SHDSH1)
Primary_slave_valve	Primary desuperheater (SHDSH1) valve position
Final_master_setpoint	Master controller setpoint for final desuperheater (SHDSH2)
Final_master_output	Master controller output for final desuperheater (SHDSH2)
Final_slave_output	Slave controller output for final desuperheater (SHDSH2)
Final_slave_valve	Final desuperheater (SHDSH2) valve position
SH-DSH1-SPRAY-MASS	Primary desuperheater (SHDSH1) spray water mass flow
SH-DSH2-SPRAY-MASS	Final desuperheater (SHDSH2) spray water mass flow
SH-DSH1-IN-TEMP	Primary desuperheater (SHDSH1) inlet temperature
SH-DSH1-OUT-TEMP	Primary desuperheater (SHDSH1) outlet temperature
SH-DSH2-IN-TEMP	Final desuperheater (SHDSH2) inlet temperature
SH-DSH2-OUT-TEMP	Final desuperheater (SHDSH2) outlet temperature

In this report, we study saturation fault simulation data and fault detection results.

Saturation Fault Simulation Data: A ramp bias of positive 0.5 is applied to the actuator for the final desupreheater valve to reach its lower bound as shown in Figure 5.**47**.

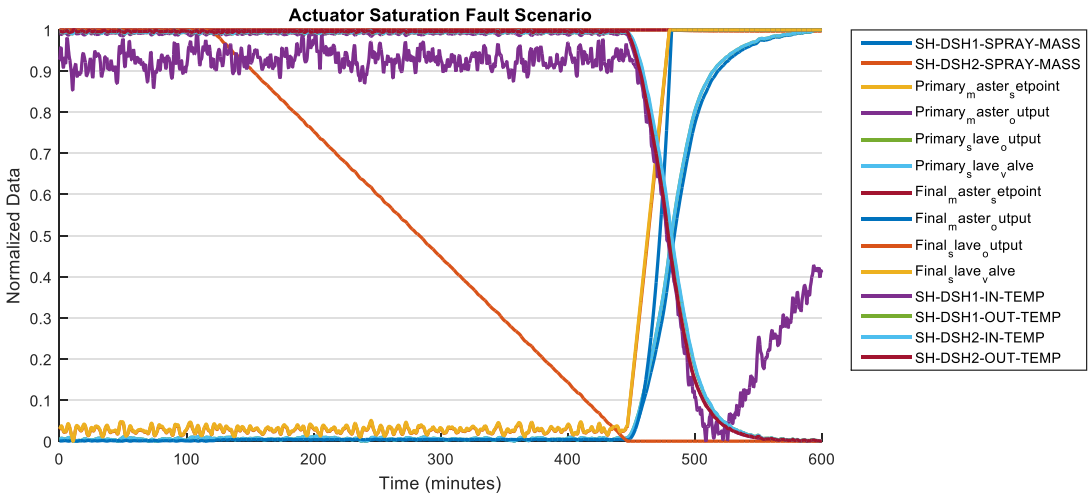


Figure 5.47: Normalized Actuator Saturation Fault Simulation Data

We can see that the final slave output changes at time 120 minutes and continues to decrease until it reaches zero at time 480 minutes. At this time, other variables start to react to the changes in the system. It is not clear from the simulation data if they have also reacted when the ramp bias starts.

Simulation Results: For consistency, we study the communication topology among the same variables considered for sensor fault scenarios that are listed in Table 5.17. In this case, we can compare the topologies and identify the differences between sensor and actuator faults. Similarly, the intrinsic communication topologies are identified from the network over different time windows and the results are passed to the change detection algorithm to detect the fault.

Table 5.17: List of Variables For Simulation

Taglist	Description
Primary_master_setpoint	Master controller setpoint for primary desuperheater (SHDSH1)
Primary_master_output	Master controller output for primary desuperheater (SHDSH1)
Primary_slave_output	Slave controller output for primary desuperheater (SHDSH1)
Final_master_setpoint	Master controller setpoint for final desuperheater (SHDSH2)
Final_master_output	Master controller output for final desuperheater (SHDSH2)
Final_slave_output	Slave controller output for final desuperheater (SHDSH2)
SH-DSH1-SPRAY-MASS	Primary desuperheater (SHDSH1) spray water mass flow
SH-DSH2-SPRAY-MASS	Final desuperheater (SHDSH2) spray water mass flow
SH-DSH1-IN-TEMP	Primary desuperheater (SHDSH1) inlet temperature
SH-DSH1-OUT-TEMP	Primary desuperheater (SHDSH1) outlet temperature
SH-DSH2-IN-TEMP	Final desuperheater (SHDSH2) inlet temperature
SH-DSH2-OUT-TEMP	Final desuperheater (SHDSH2) outlet temperature

Identified Communication Topology: The simulation was initiated with 13 agents at each node with lifespan of 50 for a total of 156 agents over the entire network. The

agents traverse the network and discover the intrinsic communication topology of the power plant. At the end of the simulation, there are approximately 78,000 agents traveling throughout the network. The simulation is run with no initial pheromone present on the ground and a pheromone decay rate of 0.2 units/iteration. A sliding window with a length of 50 is passed over the data stream and the intrinsic communication topology for each window of data is discovered. Figure 5.**48** to

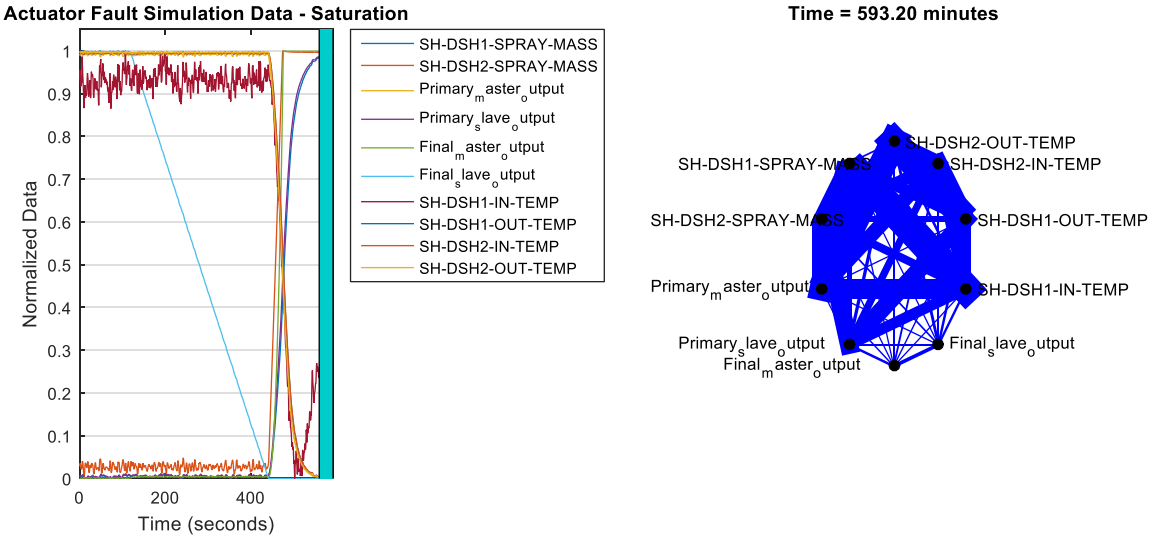


Figure 5.**54** show the information connectivity structures for different time windows. The normalized simulation data is shown on the left side of these figures with the extracted connectivity structure on the right side. The thickness of the line between each two nodes in the discovered topology is proportional to the communication strength between its terminal nodes. The system starts at a stable connectivity structure as shown in Figure 5.**48**. After the introduction of actuator bias at time 120 minutes, temporary communication links are appearing between some nodes, but the overall structure is almost the same. After time 480 minutes the actuator is saturated, the information connectivity structure changes and we can see more temporary communication links. The system does not stabilize as there is not enough time left before the end of the simulation.

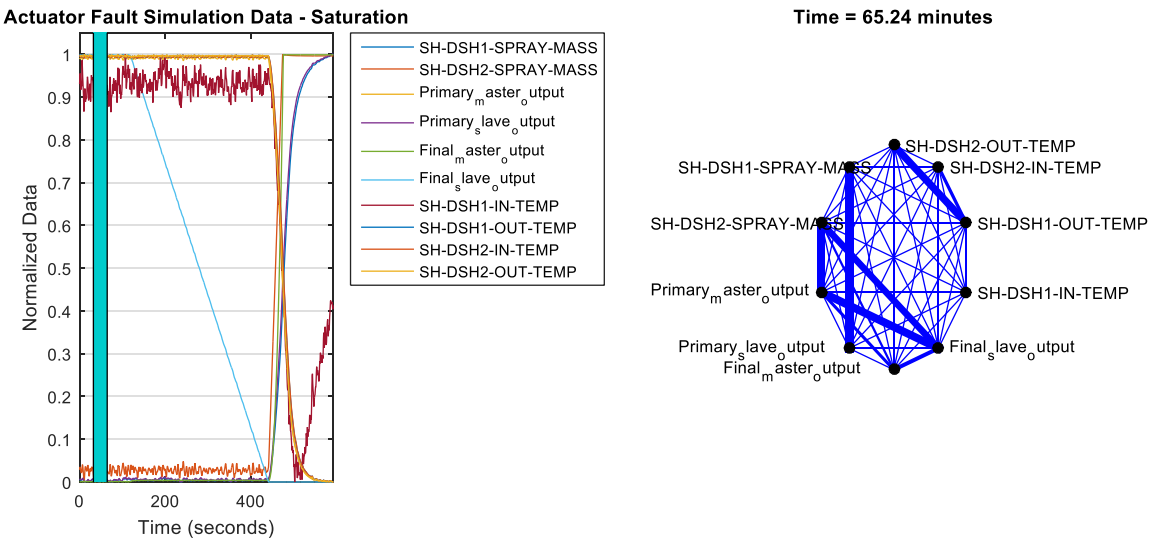


Figure 5.48: Information Connectivity Structure

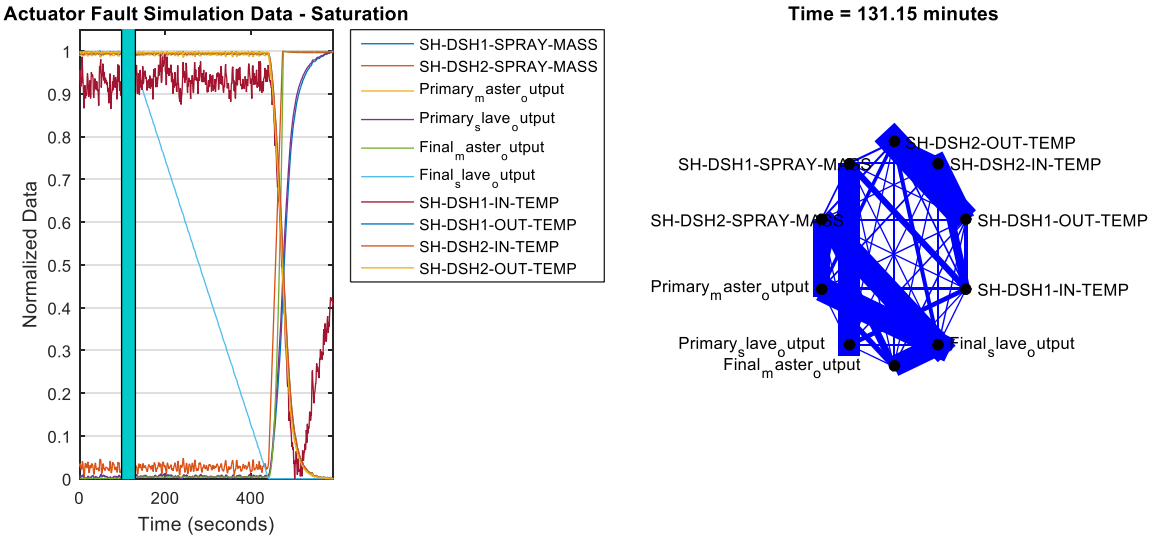


Figure 5.49: Information Connectivity Structure

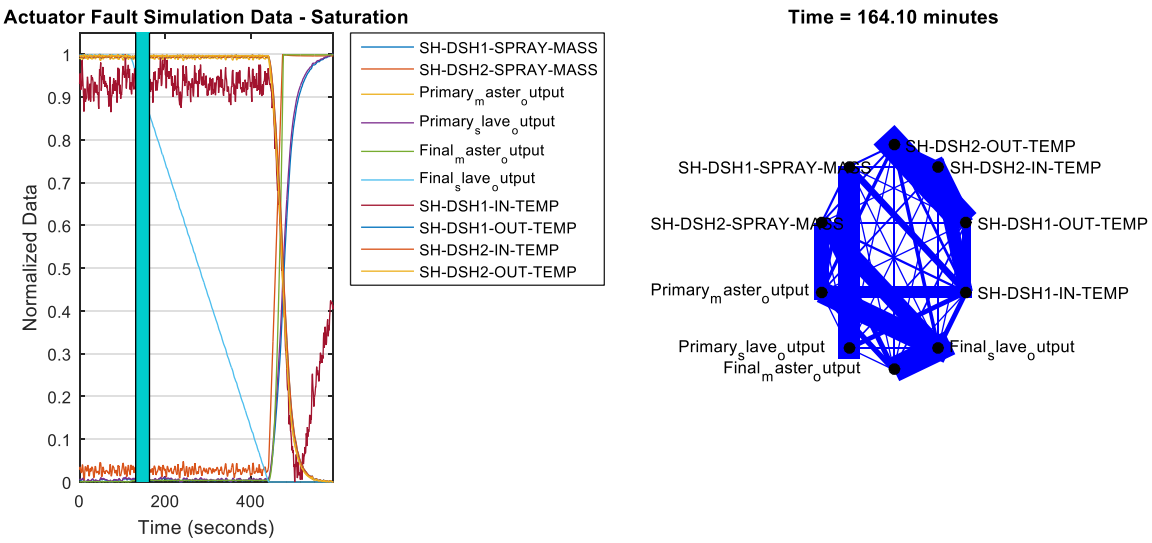


Figure 5.50: Information Connectivity Structure

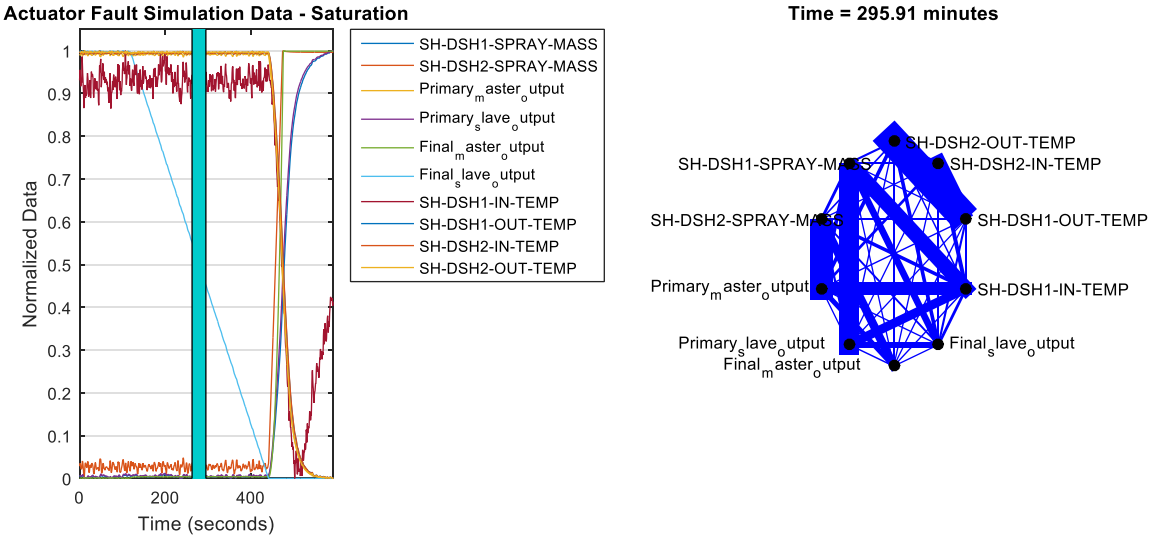


Figure 5.51: Information Connectivity Structure

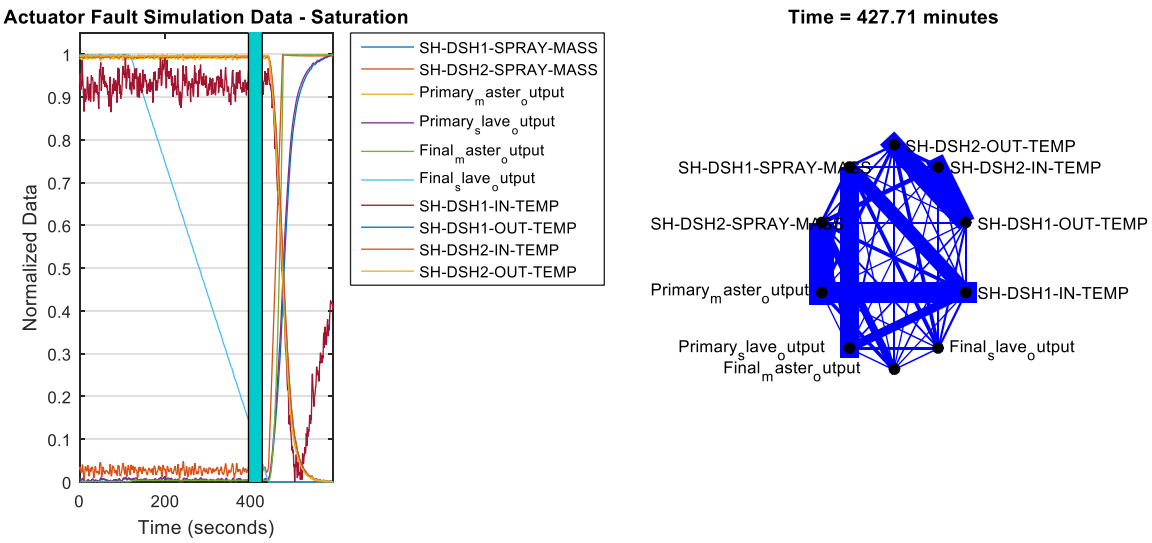


Figure 5.52: Information Connectivity Structure

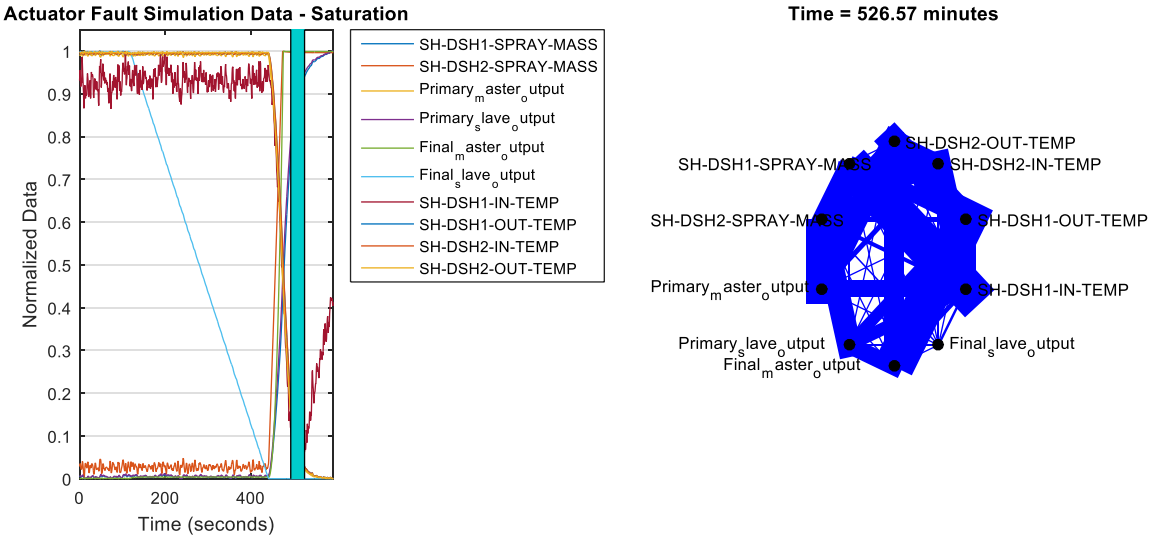


Figure 5.53: Information Connectivity Structure

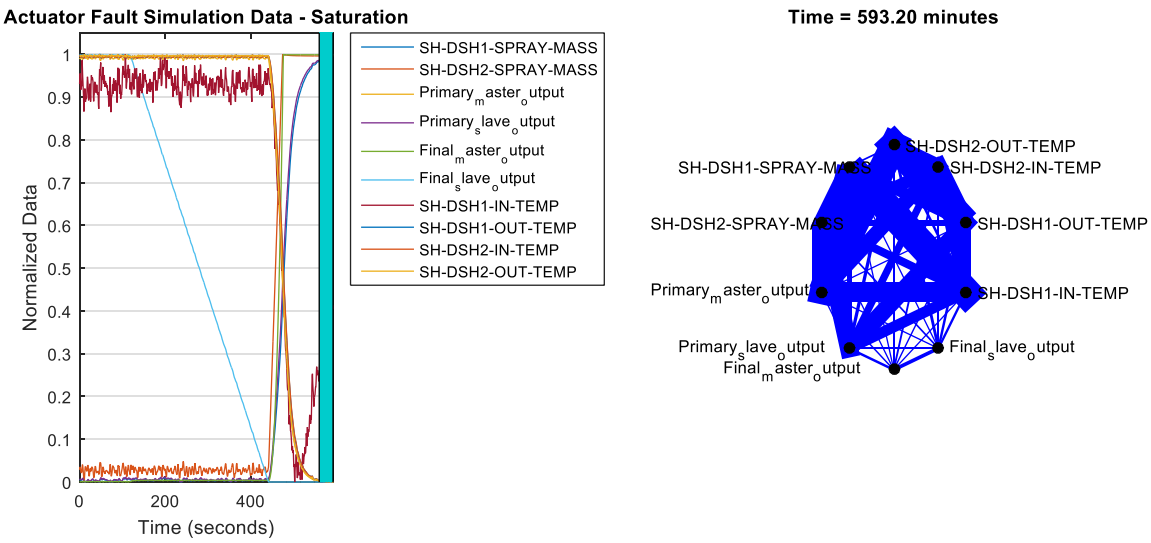


Figure 5.54: Information Connectivity Structure

Change Detection: The node similarity measure is used to identify the communication topologies to detect the presence of the fault as shown in Figure 5.55.

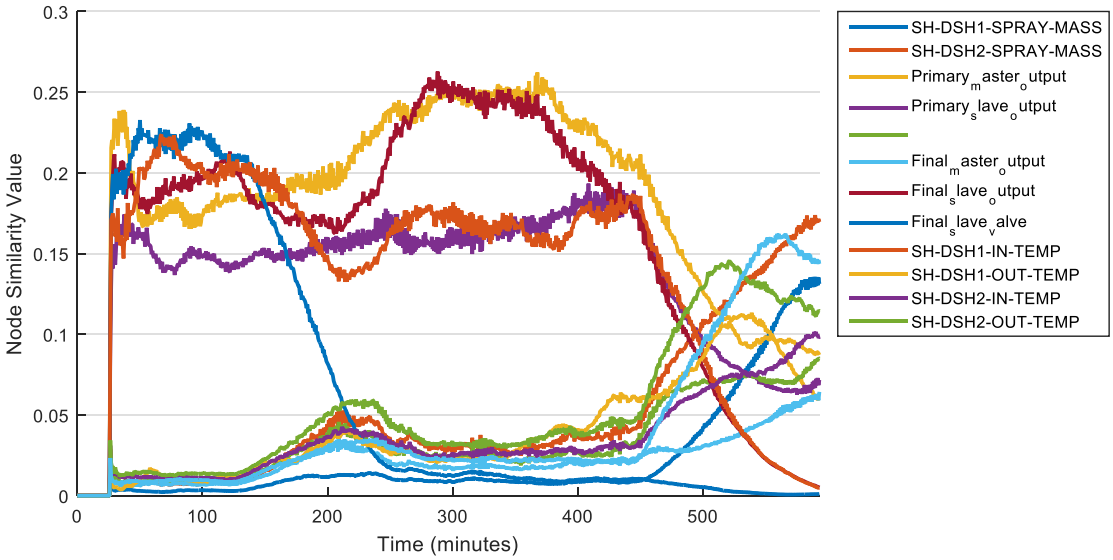


Figure 5.**55** . We can see that when the bias happens at time 120 minutes, the node similarity score for almost all of the nodes changes. After some time, these values stabilize and change again at time 480 minutes when the actuator has reached its lower bound. Therefore, we can detect the time when the fault has happened and when the actuator saturates. Note that by looking at original data, the start of the fault was not detectable. This shows a powerful aspect of our algorithm: it can detect changes in the system that can not otherwise be detected from the measures data directly.

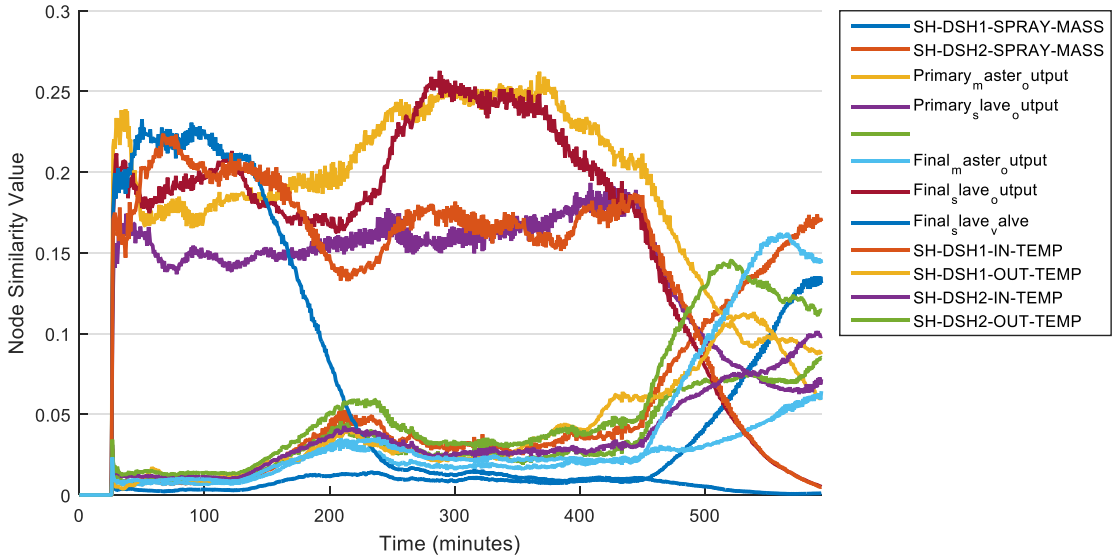


Figure 5.55: Actuator Fault Node Similarity Measure

Process Fault

In this fault, fouling occurs in the superheater, which causes a gradual change of the heat transfer coefficient. The heat transfer coefficient of all superheater heat exchangers starts a negative ram bias of 5% for around 6 hours, as listed in Table 5.18.

Table 5.18: Process Fault Scenario

Heat exchangers	Bias	Time when bias is applied	Ramp Duration
SH-Screen, SH-cavity, SH-Panel, SH-Platen, SH-Finish-IN, SH-Finish-OUT	negative 5% change for heat transfer coefficient of all the SH - heat exchangers	2 hours	6 hours

Process variables, controller outputs and setpoints for the outer loop of the main steam controllers in the boiler model are recorded every 5 seconds and are listed in Table 5.19.

Table 5.19: Process Fault Taglist

Taglist	Description
SH-DSH1-SPRAY-MASS	Primary desuperheater (SHDSH1) spray water mass flow
SH-DSH2-SPRAY-MASS	Final desuperheater (SHDSH2) spray water mass flow
Primary_master_setpoint	Master controller setpoint for primary desuperheater (SHDSH1)
Primary_master_output	Master controller output for primary desuperheater (SHDSH1)
Primary_slave_output	Slave controller output for primary desuperheater (SHDSH1)

Final_master_setpoint	Master controller setpoint for final desuperheater (SHDSH2)
Final_master_output	Master controller output for final desuperheater (SHDSH2)
Final_slave_output	Slave controller output for final desuperheater (SHDSH2)
SH-SCREEN-IN-TEMP	Superheater Screen inlet temperature
SH-SCREEN-OUT-TEMP	Superheater Screen outlet temperature
SH-CAVITY-IN-TEMP	Superheater Cavity inlet temperature
SH-CAVITY-OUT-TEMP	Superheater Cavity outlet temperature
SH-PANEL-IN-TEMP	Superheater Panel inlet temperature
SH-PANEL-OUT-TEMP	Superheater Panel outlet temperature
SH-DSH1-IN-TEMP	Primary desuperheater (SHDSH1) inlet temperature
SH-DSH1-OUT-TEMP	Primary desuperheater (SHDSH1) outlet temperature
SH-PLATE-IN-TEMP	Superheater Platen inlet temperature
SH-PLATE-OUT-TEMP	Superheater Platen outlet temperature
SH-DSH2-IN-TEMP	Final desuperheater (SHDSH2) inlet temperature
SH-DSH2-OUT-TEMP	Final desuperheater (SHDSH1) outlet temperature
SH-FININ-IN-TEMP	Superheater Finish-in inlet temperature
SH-FININ-OUT-TEMP	Superheater Finish-in outlet temperature
SH-FINOUT-OUT-TEMP	Superheater Finish-out outlet temperature
SH-SCREEN-IN-PRES	Superheater Screen inlet pressure
SH-SCREEN-OUT-PRES	Superheater Screen outlet pressure
SH-CAVITY-IN-PRES	Superheater Cavity inlet pressure
SH-CAVITY-OUT-PRES	Superheater Cavity outlet pressure
SH-PANEL-IN-PRES	Superheater Panel inlet pressure
SH-PANEL-OUT-PRES	Superheater Panel outlet pressure
SH-DSH1-IN-PRES	Primary desuperheater (SHDSH1) inlet pressure
SH-DSH1-OUT-PRES	Primary desuperheater (SHDSH1) outlet pressure
SH-PLATE-IN-PRES	Superheater Platen inlet pressure
SH-PLATE-OUT-PRES	Superheater Platen outlet pressure
SH-DSH2-IN-PRES	Final desuperheater (SHDSH2) inlet pressure
SH-DSH2-OUT-PRES	Final desuperheater (SHDSH1) outlet pressure
SH-FININ-IN-PRES	Superheater Finish-in inlet pressure
SH-FININ-OUT-PRES	Superheater Finish-in outlet pressure
SH-FINOUT-OUT-PRES	Superheater Finish-out outlet pressure

Process Fault Simulation Data: The normalized simulation data for all recorded variables is shown in Figure 5.**56**. We can see the fault starts at time 2 hours and lasts for a total of 6 hours. We also observe that after the bias is removed, the variables do not return to their original values.

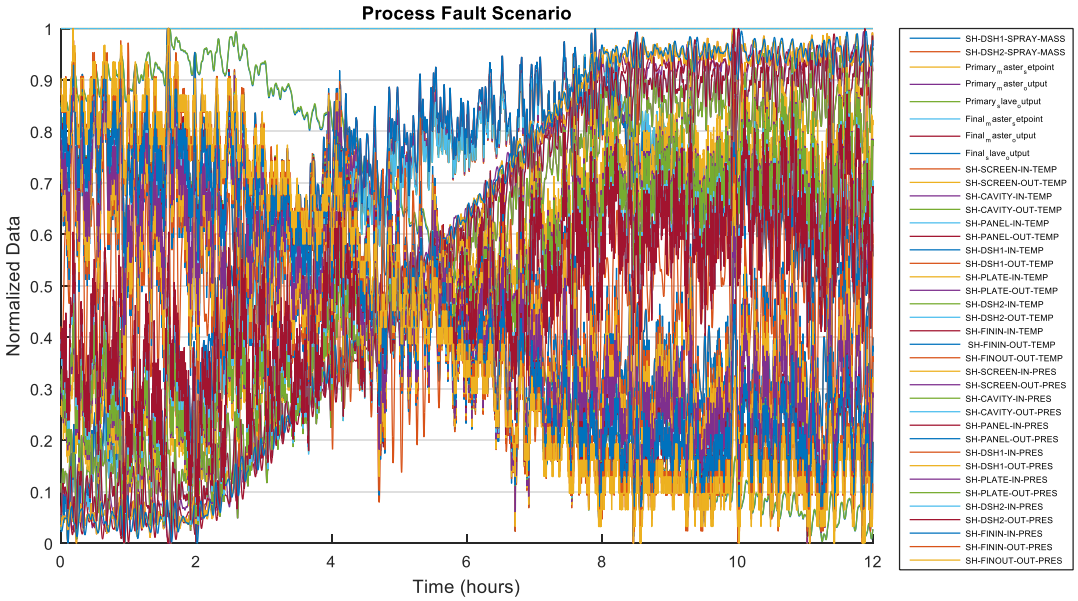


Figure 5.56 - Normalized Process Fault Simulation Data

Simulation Results: Similarly, we consider the same variables for simulation as sensor and actuator fault variables that are listed in Table 5.17.

Identified Communication Topology: The simulation specifications are similar to actuator fault. The identified intrinsic communication topologies for stable and transient states are shown in Figure 5.57 to

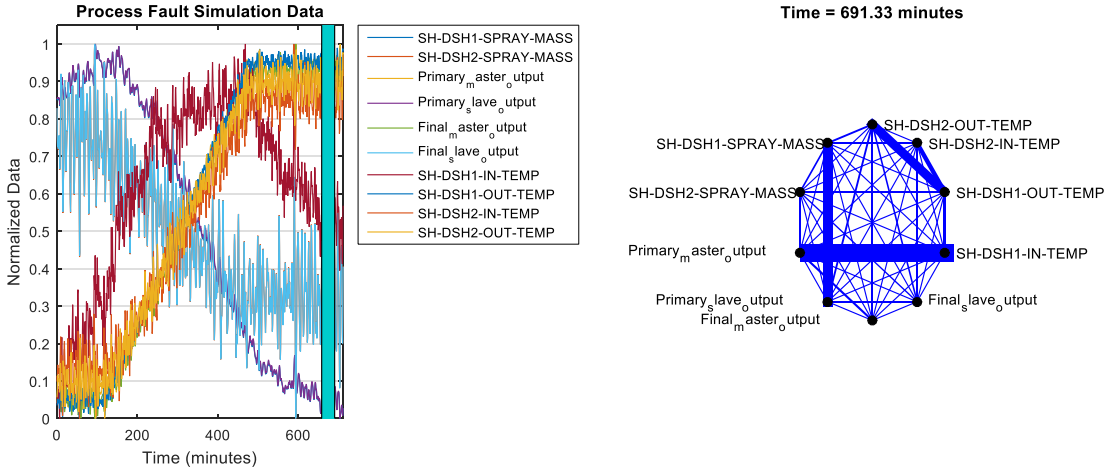


Figure 5.**62**.

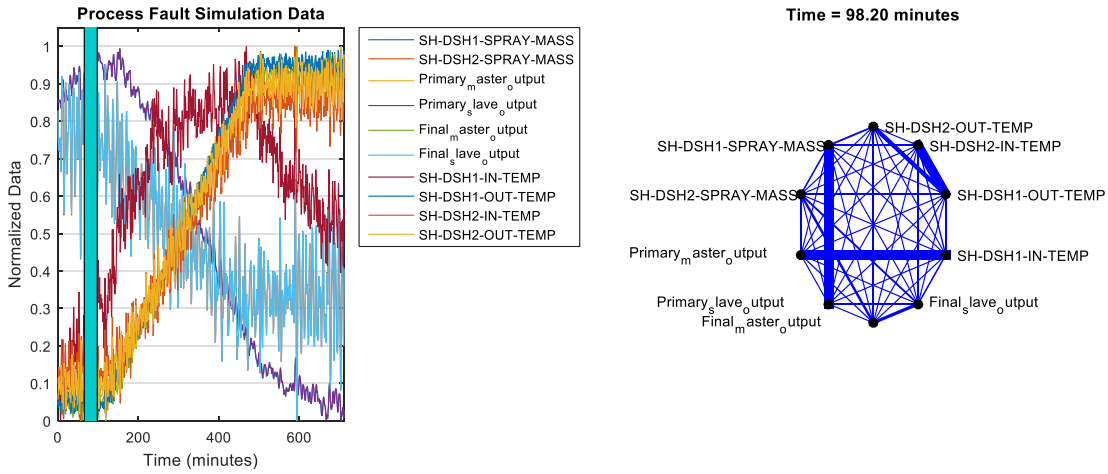


Figure 5.57: Information Connectivity Structure

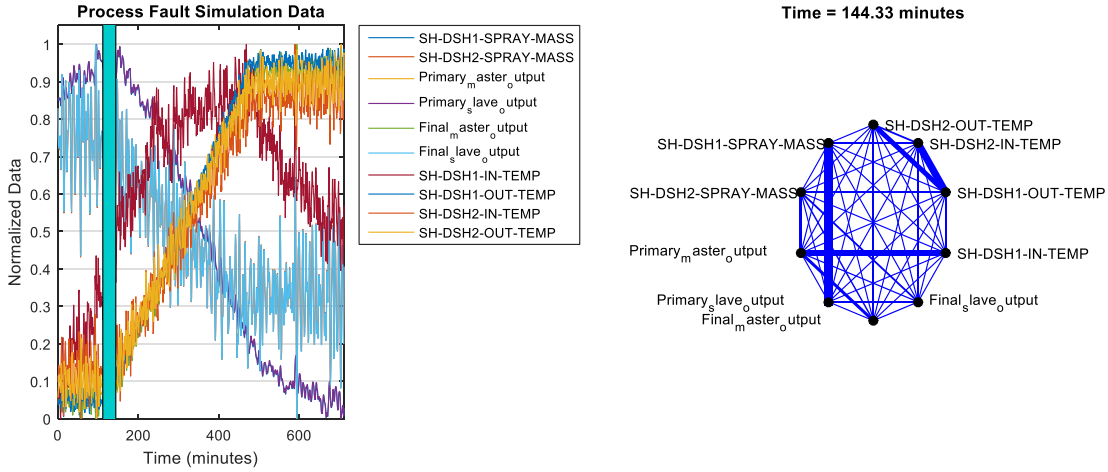


Figure 5.58: Information Connectivity Structure

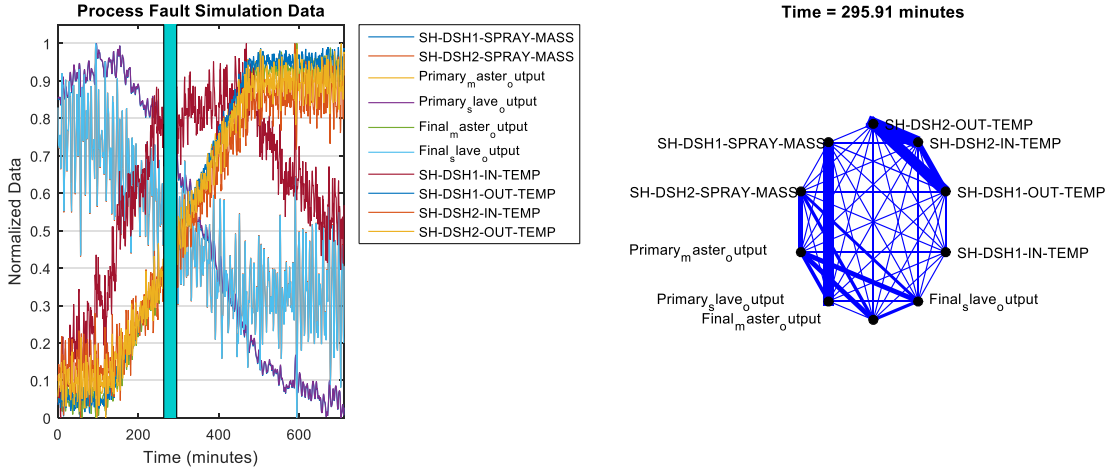


Figure 5.59: Information Connectivity Structure

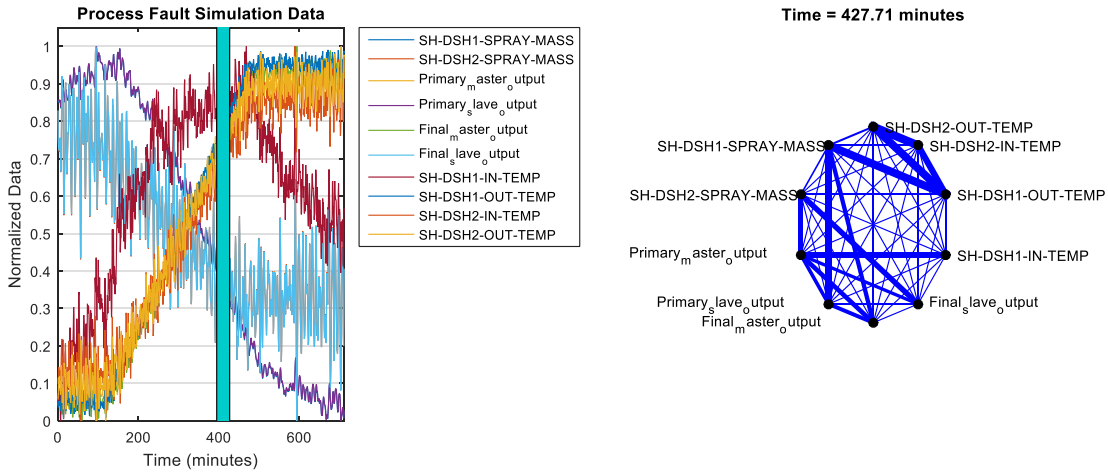


Figure 5.60: Information Connectivity Structure

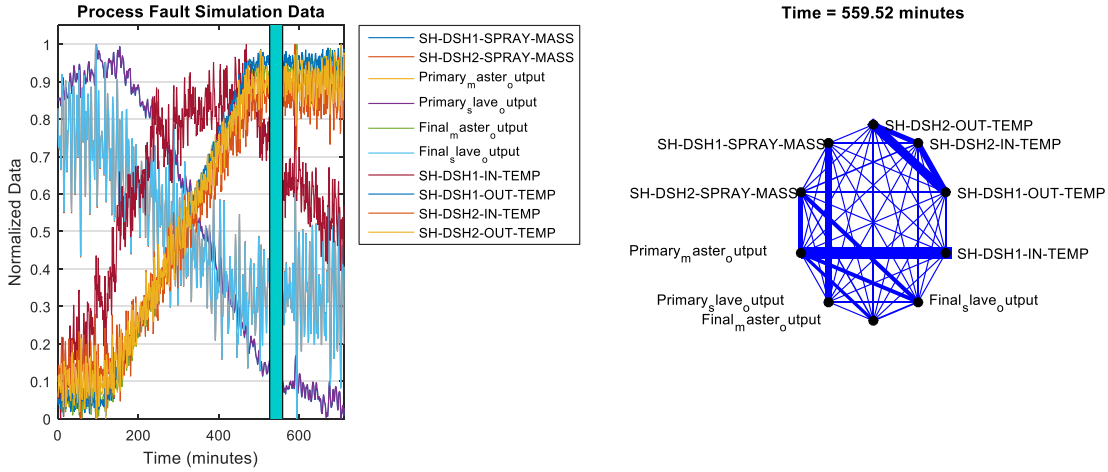


Figure 5.61: Information Connectivity Structure

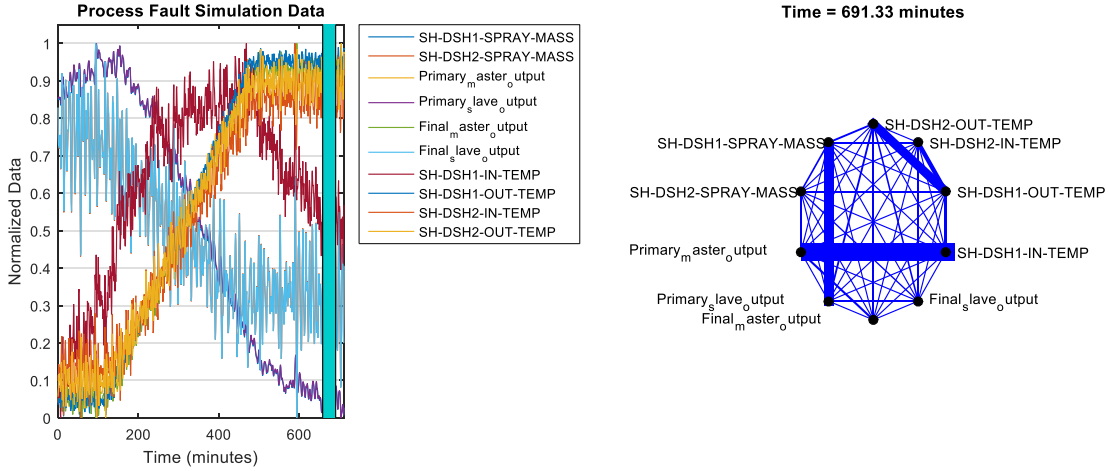


Figure 5.62: Information Connectivity Structure

Change Detection: The node similarity measure is used to identify changes in the communication topologies to detect the presence of the fault as shown in

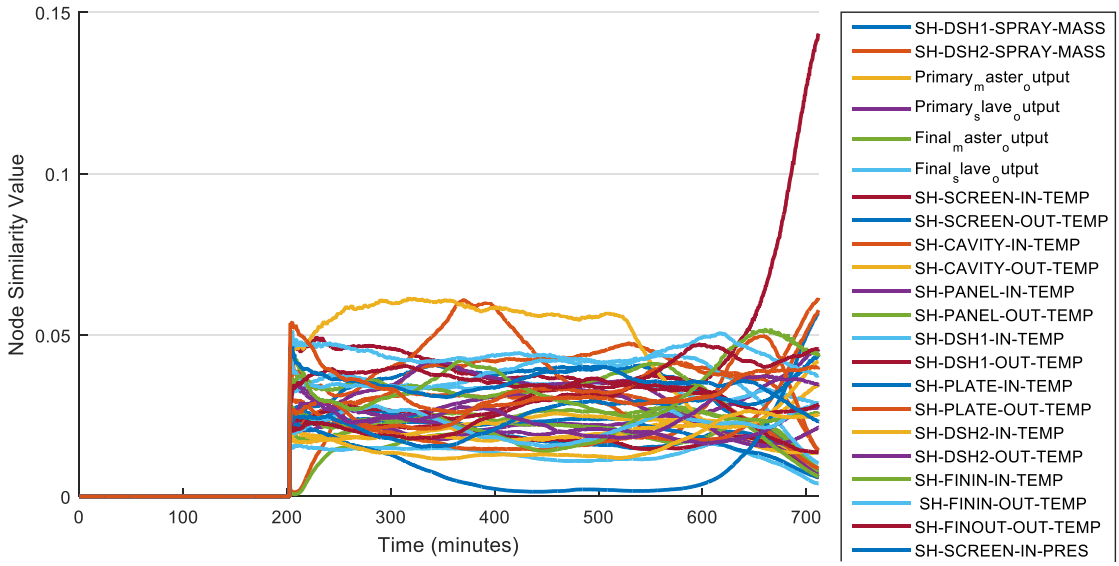


Figure 5.63.

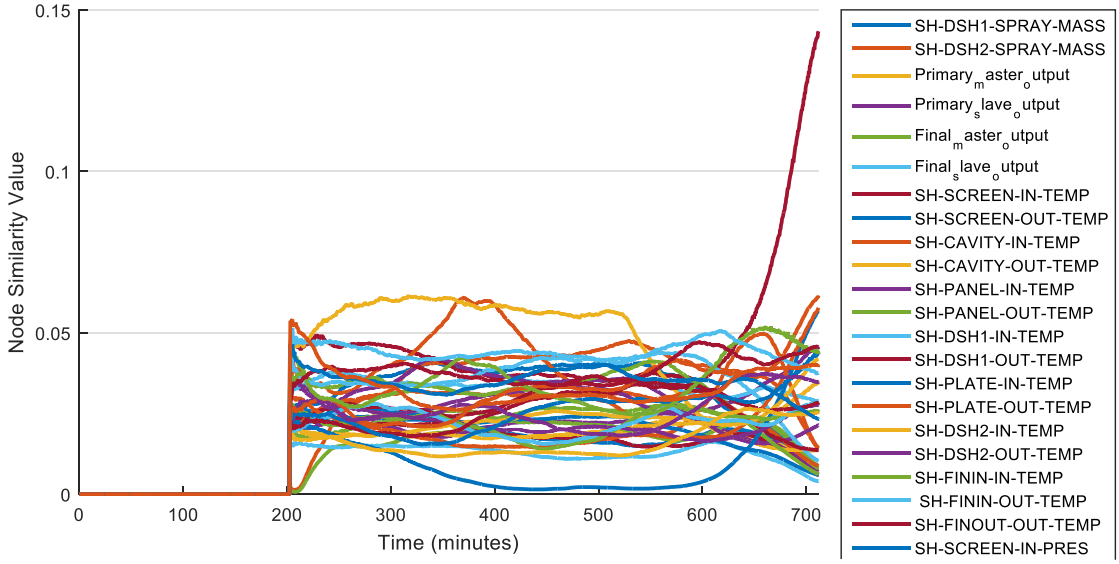


Figure 5.63: Process Fault Node Similarity Measure

Change Points: We employ our proposed change detection algorithm in detecting the change points in the system. The results for actuator, sensor and process fault simulation data are listed in Table 5.20 and we can see that our proposed algorithm can detect the changes in the system. It can detect when a change has happened and, also, when the system stabilizes after the change.

Table 5.20: Extracted Change Points

Fault Scenario	Actual Change Points	Extracted Change Points

Actuator	Step Positive		3600	1384 3954.2 6761.4
	Step Negative		3600	1660.8 4349.6 6801.5
	Ramp Positive		120 480	158.2 309.1 497.6
	Ramp Negative		120 480	166.1 355.2 494.3
	Reach Lower Bound		120 480	148.9 311.7 522.6
Sensor	**SH-DSH1**	Step Positive	30	33.1 76.2 108.4
		Step Negative	30	33.1 76.2 108.4
		Ramp Positive	120 480	165.5 313.8 541.3
		Ramp Negative	120 480	160.3 322.3 544.7
	SH-DSH2	Step Positive	30	36.1 62.2 94.6
		Step Negative	30	22.4 65.9 100.2
		Ramp Positive	120 480	168.4 383.1 504.6
Process	Process Fault		120 480	196.7 464.1 612.4

Multiple Faults Scenarios

In this part, we consider multiple faults in a single simulation data. Different fault scenarios are introduced to the plant simulation model to evaluate the performance of our algorithm in detecting the faults. Three type of faults are considered:

1. Sensor Fault
2. Actuator Fault
3. Process Fault

Sensor Fault: The temperature sensor of final desuperheater has the following faults:

1. A positive step bias of 5 degrees which starts from time 1 hour, takes around 30 minutes to reach its final value and ends at time 3 hour.
2. A positive ramp bias of 5 degrees which starts from time 5 hour, takes around an hour to reach its final value and ends at time 7 hour.

Figure 5.**64** shows the temperature sensor for the final desuperheater. We can see that it takes 30 minutes for the sensor value to get back to its original value after the step bias ends. For a ramp bias, this time is almost 1 hour. This will leave 1 hour for the network to settle back to its original state and the actuator fault starts at 9-hour time point. Note that the agents may not have sufficient time to process data and reconfigure the network to its original state. From the algorithm, the agents stay for 20 minutes at each node, collect data and then leave for the next node.

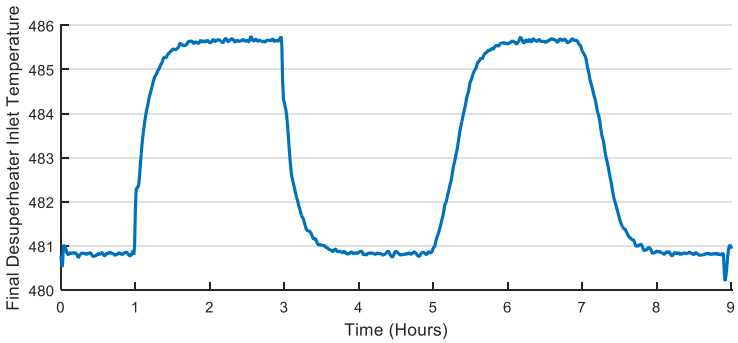


Figure 5.64: Sensor Fault

We run the simulation using two different pheromone evaporation coefficients of 0.3 and 0.6, considering 0.7 as the desired correlation coefficient value between windows of data and 40 minutes as the maximum age for the agents. The node similarity measure for the sensor fault is given in Figure 5.**65**. We can see that around the times where the fault is happening the similarity scores change and after some time they settle down, as shown in Figure 5.**66** with black and red lines representing the start and finish of a fault respectively. This is when the network is settled in a configuration related to the current state of the system. By comparing these configurations, one can identify the current state of the system.

Figure 5.**67** compares the transient and stable connectivity structures for the sensor fault scenario. In the transient structure there are temporary communication links between nodes that disappear after some time. There are also some differences between stable connectivity structures.

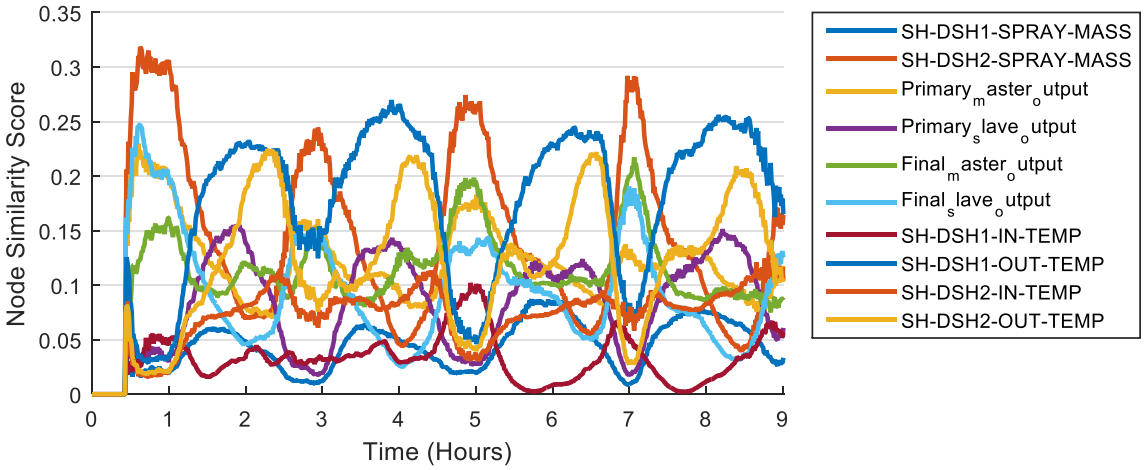


Figure 5.65: Node Similarity Measure

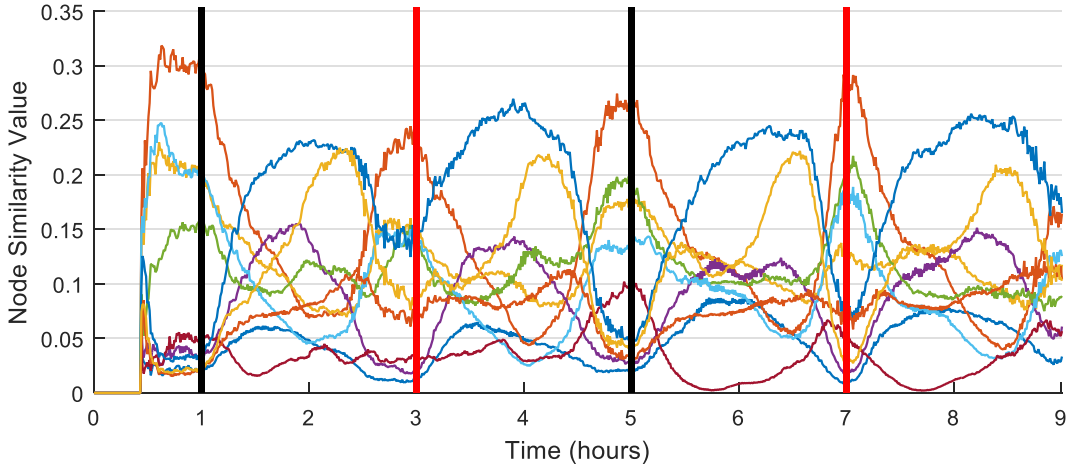
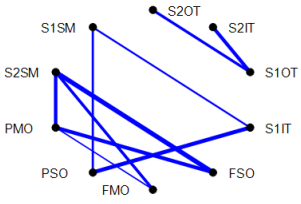
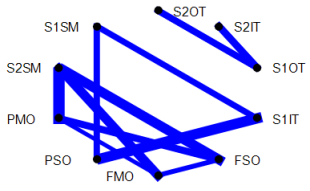


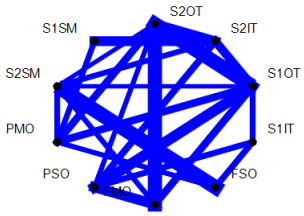
Figure 5.66:Node Similarity Measure with Sensor Fault Times



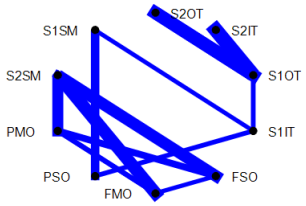
Time = 0.54 h
Initialization



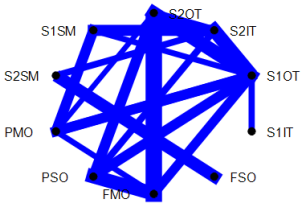
Time= 0.9 h
Stable Normal Structure



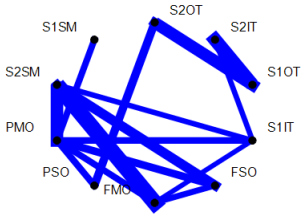
Time= 2.08 h
Transient After Step Fault Starts



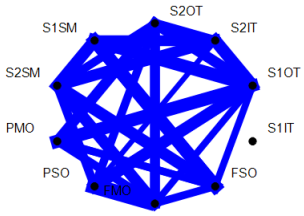
Time= 2.99 h
Stable Step Fault Structure



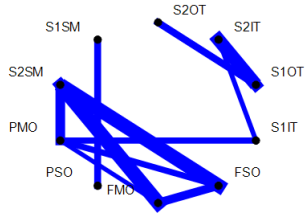
Time= 4.05 h
Transient After Step Fault Ends



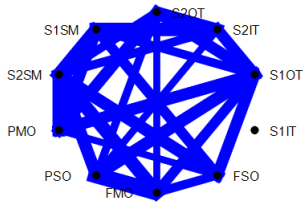
Time= 4.96 h
Stable Normal After Step Fault Structure



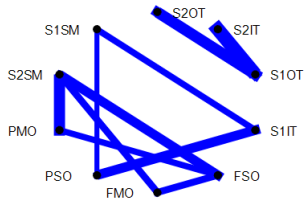
Time= 6.03 h
Transient After Ramp Fault Starts



Time= 6.99 h
Stable Ramp Fault Structure



Time= 7.68 h
Transient After Ramp Fault Ends



Time= 8.98 h
Stable Normal After Ramp Fault Structure

Figure 5.67: Sensor Fault Scenario Connectivity Structures

Actuator Fault: Figure 5.**68** shows the actuator for final desuperheater valve which has the following faults:

1. A negative step bias of 0.05 which starts from time 9 hour, takes around 30 minutes to reach its final value and ends at time 11 hour.

2. A negative ramp bias of 0.05 degrees which starts from time 13 hour, takes around an hour to reach its final value and ends at time 15 hour.

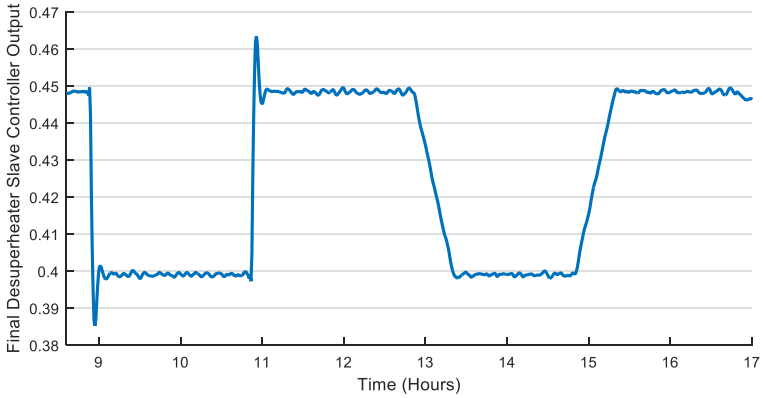


Figure 5.68: Actuator Fault

The simulation parameters are the same as sensor fault. Node similarity measure is shown in Figure 5.**69**. The lines representing the start and finish of faults are added to the plot in Figure 5.**70**.

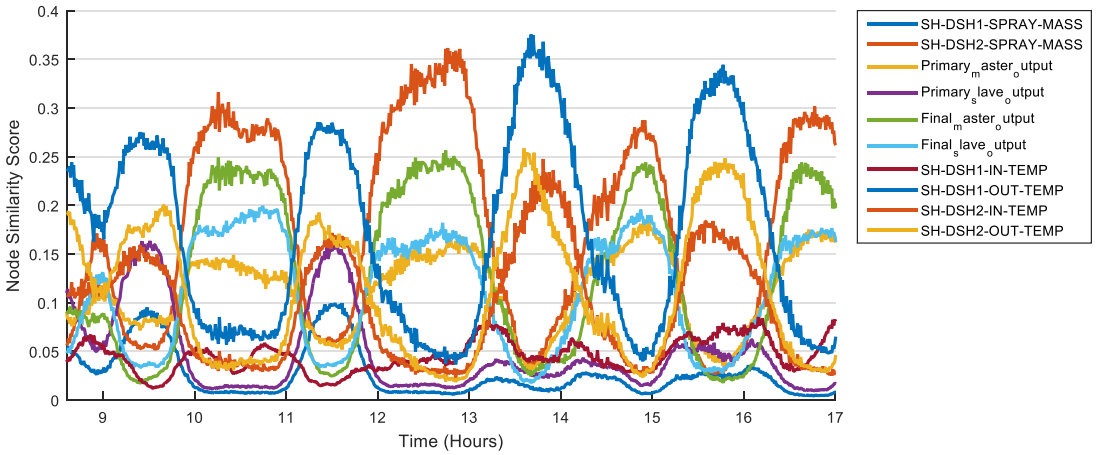


Figure 5.69: Node Similarity Measure

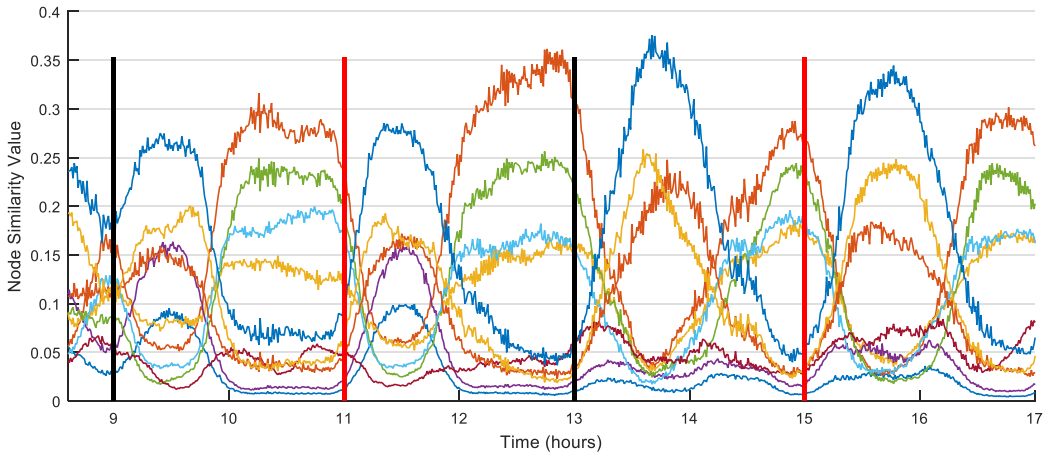
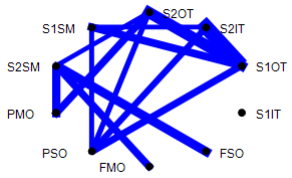


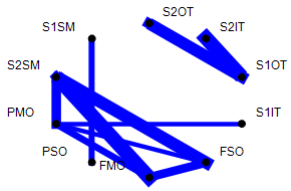
Figure 5.70: Node Similarity Measure with Actuator Fault Times

We can see that after each change, either start or finish of a fault, a similar pattern in the node similarity values is happening. There is a transient phase and after that the system has an internal change and then stabilizes.

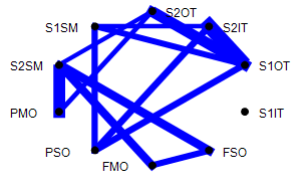
Figure 5.**71** shows the transient and stable connectivity structures. We can see that the stable connectivity structures are almost identical and have minor differences with sensor fault connectivity structures. We can use this structure in detecting the actuator fault in the system.



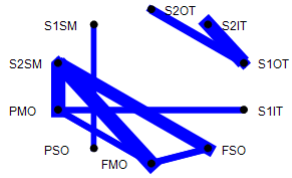
Time= 9.37 h
Transient After Step Fault Starts



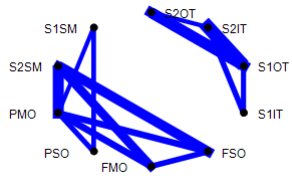
Time= 10.47 h
Stable Step Fault Structure



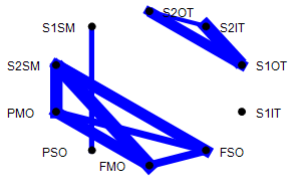
Time= 11.41 h
Transient After Step Fault Ends



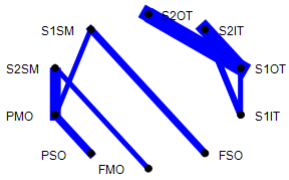
Time= 12.4 h
Stable Normal After Step Fault Structure



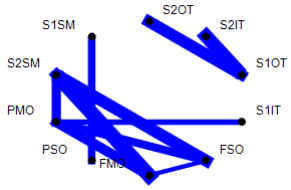
Time= 13.94 h
Transient After Ramp Fault Starts



Time= 14.49 h
Stable Ramp Fault Structure



Time= 15.37 h
Transient After Ramp Fault Ends



Time= 16.36 h
Stable Normal After Ramp Fault Structure

Figure 5.71: Actuator Fault Connectivity Structures

Process Fault: The heat transfer coefficient for all superheater heat exchangers have a negative 5% change starting at time 17 hour with a ramp duration of 30 minutes and an end time of 21 hour. Figure 5.**72** shows the normalized data for selected nodes. We can see the change in this data after the fault is imposed.

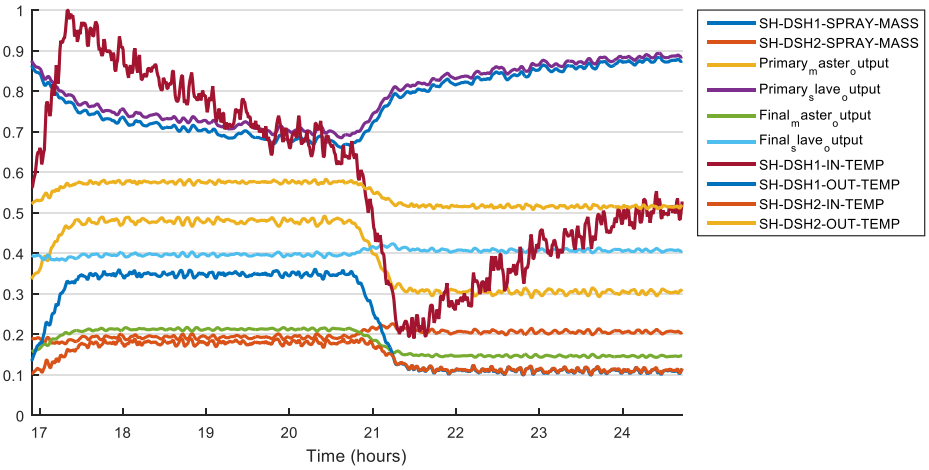


Figure 5.72: Process Fault

Node similarity measure and change lines are shown in Figure 5.**73** and Figure 5.**74**, respectively. The transient and stable patterns are visible from these figures. The transient and stable connectivity structures are listed in Figure 5.**75** and we can see that there are very minor differences between stable structures. Also note that these structures are different from the sensor and actuator stable structures, enabling us to not only detect the fault but to potentially identify the type of fault.

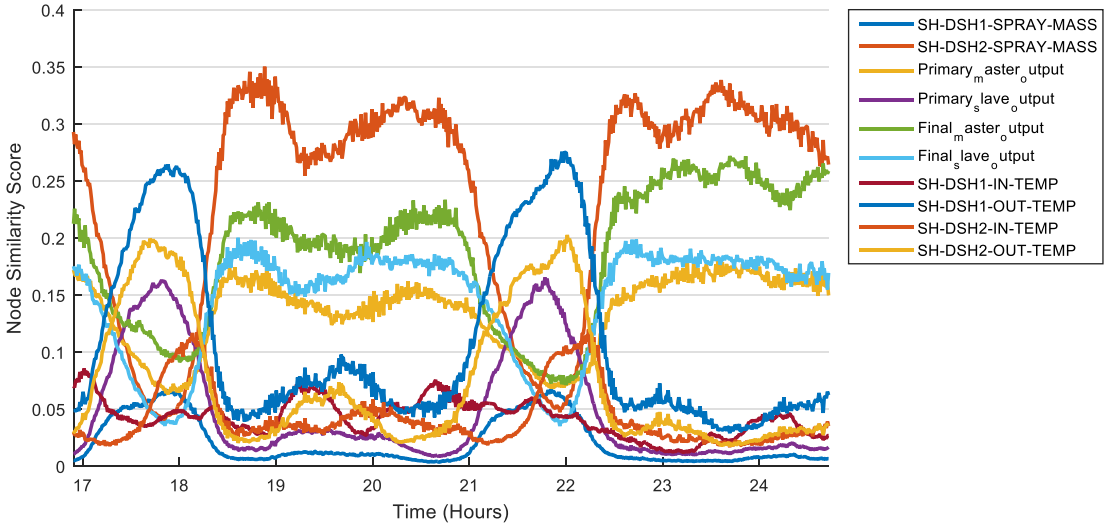


Figure 5.73- Node Similarity Measure

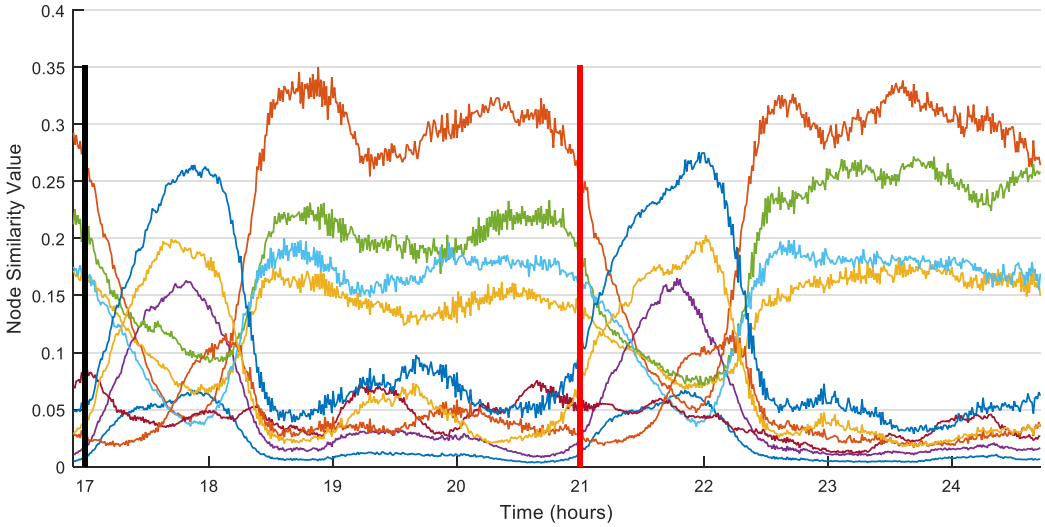
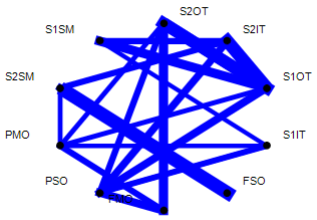
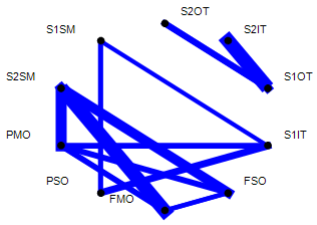


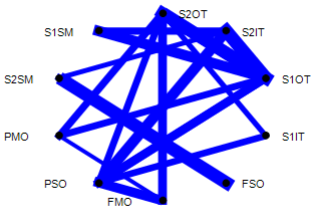
Figure 5.74- Node Similarity Measure with Process Fault Times



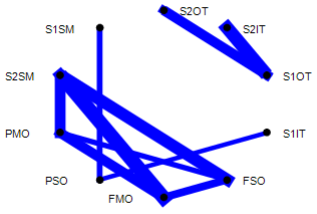
Time= 17.67 h
Transient After Process Fault Starts



Time= 19.87 h
Stable Process Fault Structure



Time= 21.52 h
Transient After Process Fault Ends



Time= 23.38 h
Stable Normal After Step Fault Structure

Figure 5.75: Process Fault Scenario Structures

Conclusions

In this section, we have used our algorithm for discovering the intrinsic communication topology for a system to detect, and possibly classify, single and multiple fault scenarios in a steam plant model. Our algorithm was applied to the faulty simulation data to detect the presence of the fault by identifying the intrinsic communication topology of the steam

plant model and observing the changes in the topology. When a fault occurs in the system the communication topology changes and using a change detection algorithm such as the node similarity measure, fault detection is performed by observing changes in the topology.

First, we considered single fault scenario: sensor, actuator or process faults. We applied the algorithm to the simulation data and identified the communication topology of the system. The communication topology of the system in the normal (baseline) operating mode is identified and we observed that after the fault is applied to the system, the identified communication topology changes. We used the node similarity measure to perform change point detection and could detect the points where the change occurred due to the fault. We compared the identified change points with the original fault location times and confirmed the performance of our algorithm in detecting the changes.

Finally, we considered multiple fault scenarios with sensor, actuator and process faults occurring in a single simulation run. We applied our algorithm to the simulation data and could observe the changes in the identified communication topology and, also, in the node similarity measure. We also observed that different faults had different connectivity topologies, so future development of this approach will look at extending the approach to combine fault detection with diagnosis.