Semantic networks for hybrid processes

by

Dhananjay Anand

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2013

Doctoral Committee:

Professor Dawn Tilbury, Co-Chair
Associate Research Scientist James Moyne, Co-Chair
Assistant Professor Ryan Eustice
Professor Ian Hiskens
Ya-Shian Li-Baboud, National Institute of Standards and Technology

For those who came before.

ACKNOWLEDGEMENTS

I would like to thank Rackham Graduate School, the NSF Engineering Research Center for Reconfigurable Manufacturing Systems, NSF grants EEC 95-92125 and CMS 05-28287 and NIST Award No. 60NANB11D183 for financial support at various points during my doctoral work. I would also like to thank Ya-Shian Li-Baboud, Prof. Ian Hiskens and Prof.Ryan Eustice for serving on my dissertation committee.

I am grateful for Professor Dawn Tilbury's guidance as my PhD advisor. Her patient charity towards my disorganization and tolerance of abused commas are only the start of her immeasurable contributions towards this goal. I feel proud calling myself her student because it implies that I might have imbibed some of the focus, diligence and clarity of thought that she applies to all her endeavors.

In Dr.James Moyne I am honored to have, an academic advisor with infectious passion for learning and a multitalented role model for life. Though I doubt I am the only victim of his zest and genuine affection towards those around him, I feel, (like the rest of us) that my relationship with him is both unique and priceless.

The RFT aisle will always hold a special place in my heart thanks to Lindsay Allen, William Harrison, Josh Langsfeld, Alex Sobolev, Jeff Fletcher, Deepak Sharma, Lucia Seno, Steve Vozar and many others. Dr.S. Krishnan, Debapriya Chatterjee, Devaky Kunneriath, Hilary Hill, Rupa Krishnan, Katie Laventall, Juil Yum and Julien Amelot; you are all walking standards for talent, persistence and kind heartedness and it is a continuing privilege being friends with you on Facebook.

Finally, I want to express my deep gratitude to my parents, grandparents and extended family. Your unwavering confidence may have rubbed off on me just a little bit. And on looking back over this journey I am thankful for its adhesive properties.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

ABSTRACT

Semantic networks for hybrid processes

by

Dhananjay Maroli Anand

Chair: Dawn Tilbury

Simulation models are often used in parallel with a physical system to facilitate control, diagnosis and monitoring. Model based methods for control, diagnosis and monitoring form the basis for the popular sobriquets 'intelligent', 'smart' or 'cyber-physical'. We refer to a configuration where a model and a physical system are run in parallel as a *hybrid process*. Discrepancies between the model and the process may be caused by a fault in the process or an error in the model. In this work we focus on correcting modeling errors and provide methods to correct or update the model when a discrepancy is observed between a model and process operating in parallel. We then show that some of the methods developed for model adaptation and diagnosis can be used for control systems design.

There are five main contributions.

The first contribution is an analysis of the practical considerations and limitations of a networked implementation of a hybrid process where all signals between the model and the process are communicated over a digital network. The analysis considers both the delay and jitter in a packet switching network as well as limits on the accuracy of clocks used to synchronize the model and process.

The second contribution is a semantic representation of models, and the network of interconnections between a model and the physical process as well as between model components. This semantic representation enables improvements to the accuracy and scope of algorithms used to update the model. By adding semantic information about the fidelity of the model, model uncertainty can be balanced against signal uncertainty originating from the communication network. By also including semantic information about the physical properties of model components, the structure of interconnections between model components may be automatically reconfigured if needed.

The third contribution is a diagnostic approach to isolate the key model components responsible for a discrepancy between model and process. Using a structure preserving realization of a system of ODEs, a Bayesian inference strategy is used to explore the interconnections between component states and reason about observed discrepancies. The method is demonstrated to work on an electrical circuit model with a determined causal direction.

The fourth contribution is an extension of the diagnostic strategy to include larger graphs with cycles, model uncertainty and measurement noise. The method uses graph theoretic tools to simplify the graph thereby making the problem more tractable. The method is applied to a semiconductor manufacturing line and shown to improve the computational feasibility, noise tolerance and accuracy of the diagnosed result.

The fifth contribution is a simulation of a distributed control system to illustrate the key contributions of this work. Using a coordinated network of electric vehicle charging stations as an example, a consensus based decentralized charging policy is implemented using the semantic modeling approach and declarative descriptions of the interconnection network. By reasoning about the model semantics, it is shown that the network structure can be automatically reconfigured to maximize the performance of the coordinated charging of electric vehicles.

CHAPTER I

Introduction

1.1 Motivation

As simulation models (and the computers that run them) become more powerful, they can be used to improve perception of an ongoing process. Simulation models can also be used to consider "what-if" analyses, i.e. to try out different scenarios both going into the future as well as in reverse e.g., to determine what possible faults in a system are consistent with the current observations.

Development work for high performance simulations is an area of research specific to a particular domain and/or application. The physics of combustion in engines, the dynamic response of electro-mechanical machines, the energy consumption in a manufacturing cell, and the performance of control software can all be simulated with continually improving fidelity. In many cases this improvement comes with the development of custom modeling methods for each application. The diversity in modeling strategy is also a result of the varied motivations for building simulations. Motivations may be academic impetus to better understand a process (e.g., biological systems), or the need to reduce the economic pressure of installing expensive infrastructure without accurate knowledge of expected behavior (e.g., manufacturing processes). Simulations also serve as proxies for future hardware in situations where diagnostic and runtime tests are hard to implement, as with electrical networks.

Mathematically modeling a physical process offers insight and understanding about its operation. This insight is still valuable after the physical process has been deployed or is operating in the field. Therefore the use of models to estimate the state of a parallel 'real' process is a natural extension to their use as virtual substitutes.

Recognizing this growing power of simulation models, research in the area of cyber–physical systems presents the idea of a **hybrid process** [68], in which a high-fidelity simulation model is run in conjunction with the actual process. A specific instance of a hybrid process is where a model of the full physical process is run in parallel with the process to estimate measurements that may not be available, in order to diagnose problems in the physical process or to predict the behavior of the physical process over time. We focus on this configuration of a cyber–physical system in this dissertation.

In an ideal world, the simulation model and its initial conditions exactly match the physical process, the environment model is also perfect and there are no unmodeled noises or disturbances. In this case the simulation model will exactly track the physical process, and can be run in parallel in open-loop, when initialized correctly. An example application is in the tele–operation of mobile robots where a model available locally may be used to estimate the state of a remote robot system and to provide feedback to an operator.

The use of models as observers or predictors is particularly useful for larger systems made up of several interconnected sub-systems that may each belong to a different physical domain. Interdependencies in these larger systems (and their models) are potentially convoluted, necessitating a formal representation for the interactions between components of a hybrid process. Prior work on the representation of networks of interdependent agents uses the notion of a semantic network [130], [139]. A **semantic network** is a graphical notation for representing knowledge in patterns of interconnected vertices and edges. Computer implementations of semantic

networks were first developed for artificial intelligence and machine translation, but earlier versions have long been used in philosophy, psychology, and linguistics. We adopt a semantic network as a directed graph to represent the dependencies between components in a hybrid process, where each vertex is a semantic representation of a component (or object) and each edge represents a dependence between vertices with the option to include domain specific ontology when required. Ontology renders a shared vocabulary and taxonomy to the model, including definitions of objects and their properties and relations.

Representing a network of dynamic systems as a semantic network of logical objects offers similar benefits in scaling and modularity to what Object Oriented Programming provides. The trend is reminiscent of the motivations behind using digital networks to connect dynamic systems: agility, reconfigurability, modularity and inter-operability. As will be described in more detail later, representing a network of dynamic systems in a semantic framework facilitates greater reconfigurability and scaling than current strategies.

The advantages of semantic modeling are highlighted in the research presented in this dissertation through analysis and innovation of methods used to address discrepancies between a model and a physical process for large systems with several interconnected components.

There will always be some level of mismatch between a model and the actual physical process. The environment can never be exactly modeled, and noise and disturbances cannot be eliminated. Even a fully semantic description is still subject to uncertainty in modeling or noise in the measurements. There is an opportunity, therefore, to study the interaction between a model and physical process when there is a discrepancy between the two. We do not consider the case of faults in the physical process here but focus, instead, on updating the model when a discrepancy is observed. Classical estimation theory for dynamic systems provides several tools

for updating models and managing signal uncertainty [17], [3] [100], [145].

However, when models of dynamic systems are expressed as semantic graphs, we explore methods to leverage both conventional estimation methods as well as graph theoretic concepts of belief propagation, logical consensus, and information entropy to improve the performance, scalability and scope of model adaptation in large, distributed, networked, heterogeneous hybrid processes.

1.2 Major contributions

The contributions of this work fall into two broad categories:

1.2.1 Model adaptation methods using semantic models

In a networked implementation of a hybrid process, signals between a model and the process are transmitted over a shared data network. The network introduces delays and jitter in the communication which must be addressed when an algorithm is designed to update a model.

We show, after evaluating the delay and jitter profiles of common networks, that the performance of model adaptation algorithms can be improved by using semantic representations of key performance parameters in the model. Improvements are demonstrated via three use cases.

In the first case, knowledge about the effect of communication delays (between the model and process) on the model-process discrepancy is provided. Using this knowledge, a state estimator is designed to adapt to changing delays and jitter in the network. This is presented in Chapter III.

In the second case, semantic knowledge about the relationship between communication constraints, measurement uncertainty and model order is provided. Using this information, the resolution or accuracy of the model is automatically adjusted to adapt to changing network conditions. In the third case, semantic assertions about

the physical nature of each component within the model are provided. The descriptions facilitate automatic reconfiguration of the model components when the structure of the model needs to be updated. Using the semantic description, every proposed reconfiguration is a physically feasible solution and can be scored based on its likelihood of satisfactorily resolving a discrepancy. Cases two and three are presented in Chapter IV.

1.2.2 Model adaptation methods using declarative network descriptions

In some cases a model is made up of several heterogenous model components connected together. For very large compositional models, it is advantageous to isolate and adapt as small a subset of model components as possible.

Assuming that a formal description of the interconnections between the components is provided, we propose a graph theoretic methodology to systematically explore the topology of interconnections and identify components that are the root cause of a discrepancy. The methodology is applied to practical topologies for electrical networks and manufacturing lines. Several specific contributions are made in order to apply the methodology to very large graphs, systems with limited observability, processes with measurement uncertainty and models comprised of components from different physical domains. These contributions are presented in Chapters V and VI.

1.3 Application areas

Semantic networks have been used in many applications where it is necessary for computers to infer meaning from a data set. Common application domains include:

- Natural language processing: sentence semantics are commonly used in programs used to detect plagiarism in natural language text.

- Robotics and automation: semantic models of maps and cause-effect relation-

ships facilitate high-level interactions between a robot and a human.

- Protein kinetics: databases of protein molecules with added semantics about their mutual interactions allows experimenters to query the database with desired chemical behaviors instead of manually searching through names.

- Google and Facebook graph search: semantic libraries for commonly searched information are designed to give meaningful answers to natural language queries rather than a list of links.

We will focus on two application areas in this dissertation:

Manufacturing processes: Manufacturing processes, e.g. for automobiles, semiconductors and chemical products, are becoming increasingly high value, high throughput and highly automated. The need for improving quality, speed and efficiency (reducing defects and power consumption) have led to the use of digital networks and model based control methods. Manufacturing processes are well suited for semantic representation since each manufacturing step can be modeled to include semantic propositions such as energy cost, causes of defects and throughput constraints. The network of interconnections between process steps is usually well defined and a clear ontology exists for the dependency between process steps (some steps have to precede some others, some act on the same feature of the product while some are independent).

We present a case study of an industrial DC motor drive circuit equipped with semantic propositions in Chapter IV. In Chapter VI we present a diagnosis methodology for semiconductor manufacturing lines by using a formal definition of the topology of dependencies between manufacturing steps.

Electrical power networks: Distribution networks for electrical power also are well suited for representation as a semantic network. Electrical machinery used

in a distribution network are usually drawn from a finite set of machine types and their behaviors can be represented in semantic form. A declarative description of the interconnection network is available in the form of a circuit diagram including a vocabulary (voltage, current, phase) for the information shared over the network.

We use semantic propositions to automatically adapt a model for an electrical transmission line in response to changes in measurement uncertainty in Chapter IV. We use declarative description of an electrical circuit to target specific parameter in need of adaptation in Chapter V and show that semantic models in conjunction with declarative topology descriptions may be used to efficiently coordinate between a network of electrical vehicles charging stations in Chapter VII.

1.4 Dissertation overview

The objective of this dissertation is to develop tools for constructing, investigating, diagnosing, and adapting semantic models when used in a hybrid process configuration.

1.4.1 Background

In Chapter II, the concepts and tools used throughout this document are introduced and defined. Prior work in the area of networked model adaptation and semantic modeling is presented.

1.4.2 Performance of networks, synchronization and model adaptation

Since one of the motivations for this work is to improve model adaptation for systems where information is shared over a digital communication network, we present

a summary of results showing the performance of commonly used digital networks in Chapter III. Protocols used in the manufacturing domain and in the monitoring and control of electrical substations are evaluated. The performance of a hybrid process also depends on the level of synchronization between the model and the process which in turn depends on the accuracy of the internal clocks used to ensure that the model operates in lock-step with reality. We also present an evaluation of Ethernet based clock synchronization algorithms used in the application areas of interest. Finally, a model adaptation method using precise clocks and inbuilt compensation for network effects is presented and the limitations discussed.

1.4.3 Model adaptation methods using model semantics and declarative network interconnections

In Chapter IV we present two specific cases where semantic information about a model is used to improve the performance of model adaptation for a networked implementation of a hybrid process.

In the first case, information about the relationship between the number of Markov parameters in a model to the accuracy of the model output, and information about the relationship between clock accuracy and measurement uncertainty in a physical process is considered. A semantic representation of the system is used to establish an associative relationship between the choice of model order and the measurement uncertainty, and used to optimize the choice of model order for a given clock accuracy.

In the second case, a model is assumed to be constructed from several model components. Assuming that information about the physical nature and relative compatibility of individual model components is provided, a framework for automated reconfiguration of model components is presented. Additionally, a semantic reward function is designed to score every proposed reconfiguration so that only 'meaningful' topologies are generated after each combinatorial reconfiguration.

1.4.4 Targeted model adaptation

In Chapter V we consider a large networked system with tight integration between physical components and their models which is typical of manufacturing systems and electrical networks. We address a scenario where it is necessary to adapt the model while satisfying network constraints. Specifically, the adaptation strategy must efficiently utilize the limited access to measurements over the communication network while meeting the scale of complexity expected in a large networked system such as a regional electrical power distribution network.

Assuming that the model (like the physical process) is made up of several interconnected components where the topology of interconnections is explicitly known, a topology exploring diagnostic procedure is proposed in order to identify and isolate specific model components in need of adaptation. By restricting adaptation to the smallest set of components possible, we show that the computational complexity of the adaptation problem is significantly reduced.

The work in this chapter also addresses technical challenges in transforming a conventional state-space model for a system into a topology that is compatible with the topology exploration procedure by using a topology preserving realization for the state-space model.

We use a 5-bus power system as a test case. The test case shows how the methods presented might be used in a wide area network implementation where component level measurements are expensive to obtain.

1.4.5 Modifying the model structure to improve targeted adaptation

In Chapter VI we apply the diagnostic procedure outlined in Chapter V to a manufacturing process. Here the diagnostic process is used to isolate key manufacturing process steps affecting the end-of-line quality of a product. The Bayesian inference algorithms used for diagnosis incur a heavy computational penalty when applied to

large topologies with repetitions, un-modeled interactions and re-entrants, all of which are commonly found in manufacturing workflows. We are able to improve the performance of the diagnostic inference process by 'folding' and 'clustering' vertices, where possible, to simplify the analysis. We also include the effect of un-modeled interactions by using a memory efficient auxiliary relaxation. Finally, we present a method to strategically reverse the simplifications, when necessary, so that the accuracy of the diagnosed result is not compromised.

The analysis process is shown to be effective at identifying the root cause of global yield discrepancies in a semiconductor manufacturing workflow. It is also shown that auxiliary relaxation improves the convergence properties of the inference algorithm and reduces the net uncertainty in the final result.

1.4.6 Distributed control using semantic networks

In Chapter VII we present a decentralized approach to regulate the cumulative electric load in a distribution circuit by coordinating between a network of electric vehicle charging stations. The coordination strategy utilizes semantic models for the charging stations as well as a formal representation of the network of interconnections between charging stations to automatically synthesize a decentralized control policy.

A consensus based implementation of the control policy is used to coordinate the charging load between a peer group of charging stations to ensure that connected electric vehicles are charged as desired while abiding by the power constraints of the local distribution circuit. A semantic representation of each consumer's objectives is used in conjunction with a declarative description of power constraints in the distribution circuit to produce an optimal control law for each charging station.

The proposed method operates in accordance with a resource allocation schedule provided by the power utility. However, by applying reasoning to the semantic network of constraints and objectives, a topology generation mechanism organizes sub-

sets of charging stations into "peer groups". The peer groups mitigate disturbances in load regulation and follow the utility schedule using a peer-to-peer arbitration algorithm. By eliminating the need for a central controller to compensate for load disturbances, the proposed solution reduces the computational requirements of the resource allocation program while improving the robustness and the response time of a coordinated peer group of chargers.

CHAPTER II

Background and Related work

2.1 Model adaptation for networked dynamic systems

Automated methods to adapt mathematical models of dynamic systems have been an area of focus since the advent of modern control theory and our contribution can be put in the context of the previous literature in the area. Notable early work by [151], [15], [32] and [100] offer algorithmic approaches to updating model parameters and estimating the internal state. The focus of much of the early work in the area was on uncertainty of the model, noise in the measurements and intermittence of observations. All of these concerns are still valid today, and despite the decades of advancements the fundamental concerns facing a model adaptation algorithm remain unchanged.

Figure 2.1 shows a generic configuration of a hybrid process with an automated model adaptation algorithm. Noisy measurements are used to update a model when there is a difference between the model output $\hat{y}$ and the physical measurement y. Three properties of the model may be updated in response to the observed difference; the internal state of the model ($\hat{x}$), model parameters ($\hat{\lambda}$) and the structure or topology of the model ($\hat{\Gamma}$). The prior work in updating each of these three properties have evolved somewhat independently. We will present some perspective on the state of the art in each of the three research domains below.

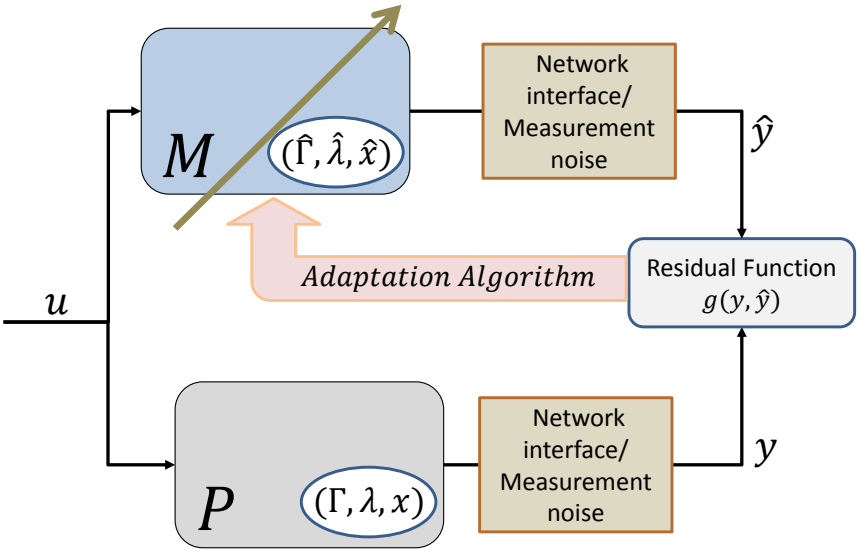

Figure 2.1: The model $\mathcal{M}$ and the physical process $\mathcal{P}$ operate in parallel and receive the same input y. Differences between the model output $\hat{y}$ and the physical measurement y are used to adapt the model using an adaptation algorithm. Both outputs are subject to disturbances such as sensor noise and network delays. Three properties of the model may be updated: the state vector $\hat{x}$, the set of model parameters $\hat{\lambda}$ and the structure of the model $\hat{\Gamma}$.

2.1.1 State estimation for networked systems

A state estimator for the physical process $\mathcal{P}$ is comprised of the model $\mathcal{M}$ and a closed loop adaptation algorithm designed to update the internal state $\hat{x}$ of $\mathcal{M}$. $\hat{x}$ is an estimate of the true state x of $\mathcal{P}$ and is updated using potentially noisy measurements y. The most ubiquitous recursive estimation technique in control is the discrete-time Kalman filter [135] – modeling the value of a measurement as a signal convolved with a random process whose parameters are related to the characteristics of a sensor or the communication channel.

The standard assumption in classical control theory is that data transmission required by a control algorithm can be performed with infinite precision. However due to the advent of digital communication technology, it is becoming more common to employ finite capacity networks for the exchange of information between components. Examples include complex dynamical processes in advanced aircraft, spacecraft, auto-

motive, industrial and power systems. Bandwidth constraints on the communication channel are often major obstacles to control system design using classical theory.

The use of networks has created a new chapter of control theory that deals with networked systems and combines together the control and communication issues, taking into account all the limitations on communication between sensors, controllers, and actuators. Recently there has been a good deal of research activity in this field. The focus has been on modeling and analyzing limited capacity channels in terms of quantization effects, channel errors, dropouts and time delays. Many of the major results in this domain treat a network as a noisy channel with known stochastic properties in order to derive an analytical closed form solution for a state estimator. In [66] and [157], the authors proposed to place a Kalman filter on the sensor side of a communication link in order to iteratively generate a linear statistical model for quantization and data loss in the channel. [159] considered Bernoulli packet losses (and delays) between the plant and the model and posed the estimator design as an H_∞ optimization problem. [137] considered a suboptimal but computationally efficient estimator that can be applied when the arrival process for measurements over a network is modeled as a Markov chain, which is more general than the assumption of a Bernoulli process. Another direction in the research proposes a systems level approach where networked control systems (network, plant, model, sensor, controller and actuator) are modeled as Markovian jump linear systems (MJLSs) [58].

Other non-analytical or simulation based approaches include [116] who present an LQG optimal estimator with non-parametric but bounded delays between sensors and the model, and between the model and the actuator. Signal conditioning methods are also widely researched as more computational resources are becoming cheaply available. For example, [117] and [96] use a combination of model based delay compensation and dropout interpolation for each measurement communicated over a network.

There is also an extensive literature, inspired by Shannon's results on the maximum bit-rate that a channel with noise can reliably carry, whose goal is to determine the minimum bit-rate that is needed to stabilize an estimator through feedback [76], [93].

Our contribution to networked state estimation, presented in Chapter III, is an analysis of fundamental limits on state estimation accuracy based on empirical observations of network delays in manufacturing systems and power networks. The accuracy of distributed clocks and the time reference as a limiting factor on accuracy are presented in detail. We also consider the accuracy of sensors, models and the network in the design of an estimator and present a design tradeoff to maximize system level performance.

2.1.2 Parameter estimation for large models

If the model $\mathcal{M}$ contains a set of parameters λ, then a parameter estimation procedure may be used to find a probability distribution for each element in λ that best fits the measurements from $\mathcal{P}$. The principle of maximum likelihood estimation (MLE) [48] states that the desired probability distribution for λ is the one that makes the observed data Y most likely. The MLE for λ is sought by searching the multi-dimensional parameter space and the algebraic complexity of the search/convergence equations is exponential in the number of parameters. Algorithms for MLE can be found in the active and well established fields of system identification [99] and adaptive systems [27].

Recent improvements to MLE algorithms address the exponential scaling concerns in order to consider larger models. The convergence improvements sacrifice some optimal properties of the MLE such as the efficiency and sufficiency condition [127]. The tradeoff of performance versus optimality differs depending on the domain. In [78] a formulation suited to financial data is presented where a very large number of

data samples are available. With very large data sets the authors are able to improve the lowest-possible variance of each parameter. In [142] non-parametric distributions are considered for the very large models used for predicting the evolutionary dynamics of genetic markers. Models for gene expression tend to be fairly uncertain justifying an improvement in asymptotic consistency of the parameters against an increase in variance of each parameter.

MLEs have widespread application in the modeling of power systems [57], [133]. The MLE is commonly used to estimate the static state of an electrical network. With the recent interest in dynamic state estimation for power systems, faster and more numerically efficient MLEs have begun to appear in literature. The authors in [124] discuss a dynamic parameter tuning system, based on an MLE, for large power networks. Similar MLE based techniques have also been proposed for modeling consumers or loads [110]. Both power system parameter estimation methods use heuristics in the form of a bounded probability distribution on the parameter space or manually inserted constraints on some parameters to improve the convergence rate.

Our approach, presented in Chapter V, addresses the challenge of scale for large parameter estimation problems by decomposing the model and isolating a subset of parameters that warrant adaptation. Our method is compatible with standard MLE algorithms and suited for systems made up of multiple components such as electrical networks and manufacturing lines, where restricting an MLE to the smallest set of parameters possible offers significant computational savings and accuracy improvements.

2.1.3 Reconfiguring the structure of a model in a distributed system

Most model adaptation methods assume that the structure of the system is given a priori. However, there is also well established research on identifying the structure of a system based on its input-output properties. The domain covers the area of

fuzzy logic [94], neural networks [12, 97] and non-linear system identification [67]. The end result of structure/topology identification is an automatically synthesized interconnected structure of model elements. For example, using an un-supervised neural framework, a model made up of a network of neurons with interconnections and weights may be obtained. Apart from physical similarities to biological neural networks [162], the neural structure rarely exhibits physical similarity to the system being identified.

Structure identification of power system models [118] is also widely researched. In the surveyed cases, power systems are modeled as networks of interconnected linear dynamic systems [132]. The structure identification methodology is reduced to a MIMO case of Markov parameter identification. The decomposition of a non-linear process into cascaded linear blocks is another commonly used method for identifying multi-periodic models for gear chatter and harmonic distortion [94]. Theoretical decomposition techniques also exist for cascaded or parallel Hammerstein-Weiner processes [67].

For more non-linear, multi-physics systems such as protein structures [83] or manufacturing processes [126], a supervised neural network is a common approach. Like the other methods mentioned here, a neural model is capable of producing good input-output correspondence with the process. However, the lack of physical insight in many of the methods presented here is a significant limitation.

We use several declarative forms to represent the structure of a model in our research. In all cases we retain the physical analogue to the real physical system as far as possible. In some cases we show that semantic information about component compatibility in the physical system or a reward function based on physical feasibility can be used to improve structural adaptation methods.

2.2 Semantic modeling

As defined in [139], a semantic network is a notation for representing knowledge in patterns of interconnected vertices or edges. Semantic networks are commonly used to represent formal logic in philosophy and linguistics, but can also be used to represent knowledge for any system with a formal set of definitions (vertices) and assertions (edges). Once a semantic network is constructed, it can also be used to support an automated system for reasoning about knowledge [140].

The two types of semantic elements we will use to build a semantic network are "Definitional models" that form the vertices and "Assertional descriptions" of the network of interconnections that form the edges.

A definitional model represents a component by specifying the *genus* or general type and the *differentiae* that distinguish particular instances of the genus. Say for example that a new motor is designed and features a thermal cut-out feature. The definition the motor model is greatly enhanced with the knowledge that most of the motor is made up of components from the 'motor genus' but this instance of the motor features one special differentiator in the form of the thermal cutout. The definitional representation can be used to include semantic information about the performance of a model as well. Continuing our example, consider that the motor model also includes a voltage sensor with a specified noise profile. The voltage sensor may be defined as an instance of a general genus of noisy sensors. Using the definitional hierarchy for every instance of a sensor, a computer program can be written to aggregate all noisy sensors in a large interconnected system with several motors and assess the cumulative uncertainty in a given measurement.

Clearly, we also require a formal description of the network of interconnections between the definitional models to build a semantic graph. We use an assertional network to explicitly declare an entity-relationship map between definitional models. The assertional network expresses the relationships between entities using assertions

such as *Motor1*↔**electrically connected to**↔*Sensor1*, where '*Sensor1*' and '*Motor1*' are instances of definitional models and '**electrically connected to**' is an assertion. Other assertions that are commonly used include A↔**occurs after**↔B or A↔**causally follows**↔B↔**causally follows**↔C.

Semantic representations of large electrical networks consist of definitional models of components such as transformers and switches. Definitional models for manufacturing lines include drilling operations and welding operations. Assertional declarations are made to describe the network of electrical interconnections between various electrical components or the sequence in which various machining operations are performed.

A combination of definitional models and topological assertions allow us to equip a model with tools and information required for improved model adaptation, diagnosis and control.

We use the language *Modelica*® [51] for semantic modeling. Modelica is an object-oriented, equation-based, multi-domain modeling language primarily aimed at physical systems. The model behavior is based on ordinary and differential algebraic equation systems combined with discrete events. Modelica allows the semantic model to be constructed acausally so that causal constraints (or assertions) may be added when needed. In addition, Modelica provides a pervasive definitional hierarchy for every model component enabling a large model made up of many components to be constructed compositionally.

CHAPTER III

Networked state estimation

The work presented in this chapter appears in proceedings of the 2009 and 2010 *IEEE International Conference on Automation Science and Engineering* and the 2009 *International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication* ([9], [8], [10]).

3.1 Introduction

In process control, the quantities one is interested in cannot always be measured directly. State estimation is a technique that reconstructs the state vector of a physical process using a model in combination with available measurements from the process. When measurements are reported over a communication network, they may be lost or delayed due to finite communication bandwidth and communication interference. In order to effectively design a state estimator which is capable of reconstructing the process state over a network, it is necessary to first characterize the nature of delays and transmission loss in practical networks. We present a comprehensive record of network delays and loss for commonly used Ethernet based communication networks under several operating conditions in our prior work [9]. This data can be used to propose some likely delay distributions for network components including in network installations for electrical substations and industrial plants. In this chapter we will

present a summary of our experimental results characterizing delay and jitter for digital communication networks.

In studying estimator/observer based methods to mitigate the effects of network delays, we will focus on another implementation problem common in networked control, that of clock synchronization. Most state estimation algorithms used for networked systems require precise knowledge of time across the network. Clocks that are synchronized over an Ethernet network are also affected by network delay and jitter and have limited precision. We evaluate the performance of two clock synchronization algorithms operating on industrial and substation networks in [10] and then subsequently are able to include the effect of finite precision clocks and empirically derived network delay profiles to design a state estimator for control networks [8].

3.2 Network performance characterization

Network performance considerations for control applications vary as widely as the context in which the term "networked control" is used. In a process control application in a chemical plant, for example, process parameters may be sampled by sensors once every few minutes all the way down to once every few milliseconds [103]. While both of these may be low data bandwidth applications, the fast sampling sensor will need to access the transmission medium at a higher frequency, placing much higher demands on the Medium Access Control (MAC) layer than the slow sensor. Similarly, a network designed to support supervisory work flow control and a network carrying signals from the safety system may both feature high network utilization, but the time criticality of the transmissions from the safety system demands prioritization over other transmissions in case of medium access contention. Also, the size of the data frames communicated over Networked Control Systems (NCSs) vary greatly from several bytes in a low level sensor/actuator interface to several thousand bytes in a high level Human Machine Interface (HMI).

Table 3.1: Performance specifications for industrial process control (automotive manufacturing). Performance requirements are classified by the maximum acceptable delay for a message to be received.

	Poll interval	Message size	# of nodes	Range
High Speed I/O	10ms	$\sim$ 64 Bytes	50	10m
Medium Speed I/O	100ms to 1s	$\sim$ 64 Kilobytes	50	30-60m
Low Speed I/O	1hour	$>$ 1 Megabyte	100+	100+ m

In the following section we will present a summary of our survey to identify network performance specifications for industrial control networks and control networks in electrical substations.

3.2.1 Performance specifications

Industrial process control networks are divided into three operational classes: High speed I/O for applications such as automated carriers on electrified monorail systems, medium speed I/O for applications such as mobile PLC test stands and low speed I/O for applications such as remotely monitored utility meters. The requirements for each of these classes from a control system designer's perspective are tabulated in Table 3.1 [128].

In the case of substation monitoring and control, the IEEE Power Engineering Society has defined communication performance requirements for substation telecommunications in IEEE Standard 1646 [74]. The standard broadly classifies the communication delivery time requirements for substation automation into four classes presented in Table 3.2.

Tables 3.1 and 3.2 provide a general sense for the rate at which messages are expected to be transmitted over the network. The message rate is also an indication of the maximum update frequency that can be expected when designing a state estimator.

Table 3.2: Performance specifications for electrical substations

	Poll interval	Message size	# of nodes	Range
Configuration updates	>100ms	$\geq$ 10 KBytes	100+	100 Km
Non-critical measurement data	10ms to 100ms	$\sim$ 10 Bytes	50+	30-200m
Event notification messages	2ms to 10ms	$\sim$ 4 Bytes	50+	100m
Real time sampled measurements	$<$ 2ms	$\sim$ 4 Bytes	10-100	100m

3.2.2 Performance evaluation of Ethernet networks

The communication delays in a network are rarely deterministic, the possibility of large outliers and jitter are not captured in the performance specifications in Tables 3.1 and 3.2.

We conducted experiments to more accurately measure the time taken by an Ethernet network to transmit data packets between two nodes. The Ethernet link was restricted to a point to point link and the data packets were generated by a software emulator for network protocols used for substation automation (IEC-61850) [90] and industrial automation (EthernetIP/OPC). Figure 3.1 shows a sequence of round trip delays for a single hop test. Note that the average round trip delay is about 0.5 ms. The distribution of delay values shown in Figure 3.2 is non-parametric and shows large outliers that significantly limit the accuracy of networked state estimators and networked controllers.

Further, as shown in Figure 3.3, the distribution and magnitude of delays for a industrial grade wireless network is significantly worse.

3.3 Distributed Clocks

In a widely distributed system such as the power grid, it is critical that each device be aware of the global time at which an operation is performed or measurement taken so that, for example, the device may be able to coordinate its operation with other components to collectively deliver uninterrupted power. Precise clocks also

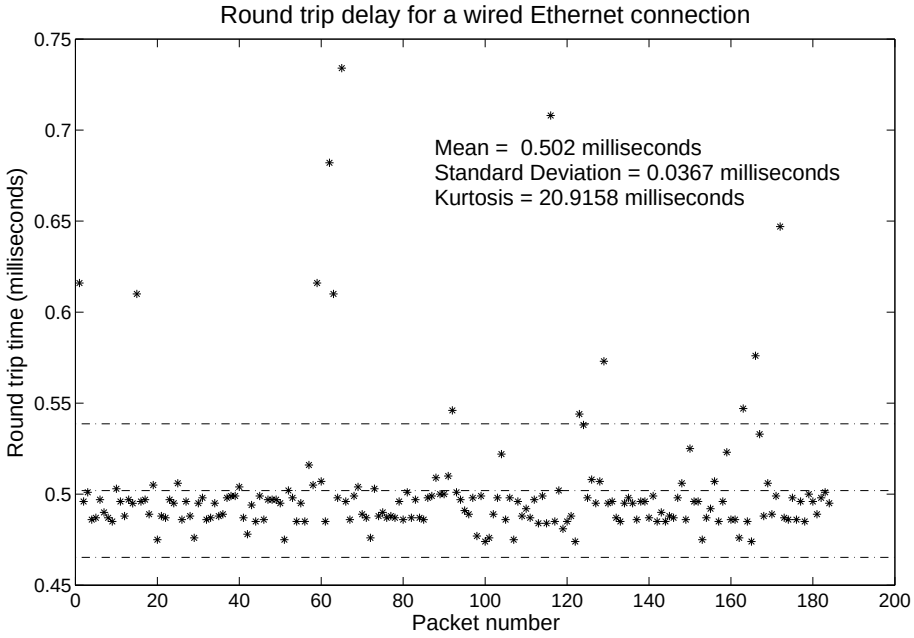

Figure 3.1: The round trip time for 60byte packets over a wired Ethernet connection between two nodes. The data is collected under nominal network cross-traffic (about 40% of the maximum bandwidth). The mean value is 0.502 ms, and the maximum delay is 46% larger than the mean.

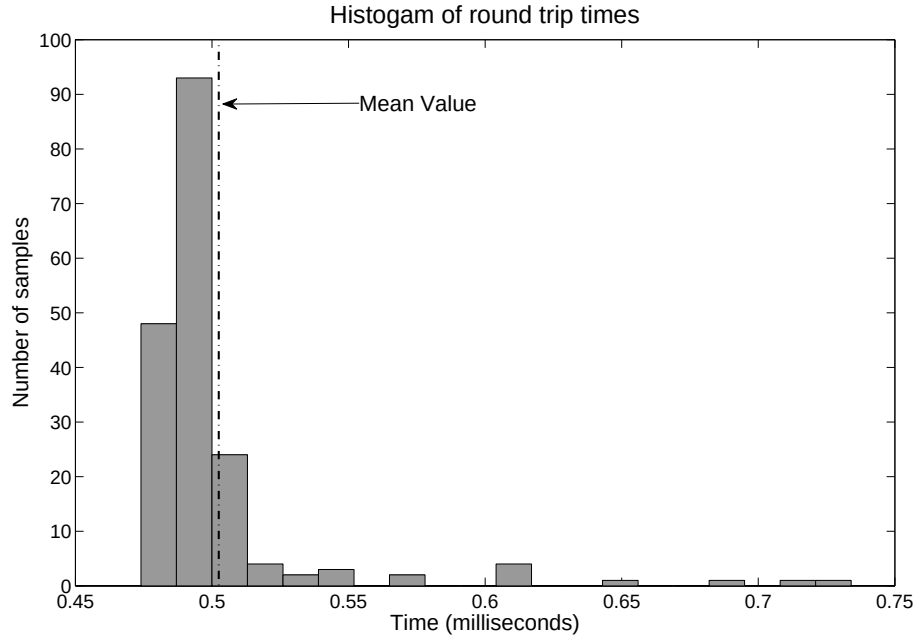

Figure 3.2: Histogram of the the delay values in Figure 3.1. The figure shows most of the delay values clustered about the mean and a small number of outliers clustered around 0.7 ms.

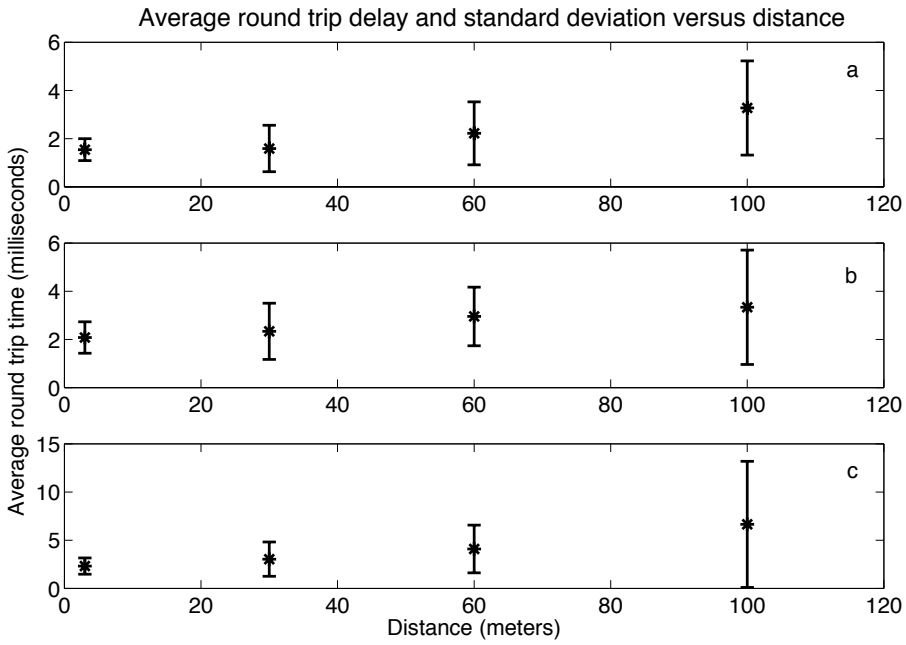

Figure 3.3: The average (and 1σ spread) of message delays for a point to point wireless network. The plots **a** and **b** correspond to industrial networks operating indoors and outdoors respectively. Both plots (**a** and **b**) show the delays for a packet switched wireless network reporting sampled values. Plot **c** shows delays for an IEC-61850 datagram used in an electrical substation.

enable simulation models to interact with physical sensors to create as accurate of a representation of the grid as possible [84] and [22].

In order to perform coordinated tasks, a device's local clock needs to be synchronized to a global time reference. Time stamps based upon this synchronized time are applied to measurements and provide a reference to the time at which the measurement occurred. The measurements can then be ordered by any user to provide a time-line of events or a snapshot at a particular instant of time. The authors in [19] present a discussion on the importance of clock synchronization for power grid devices.

A device which is playing an increasingly important role in grid health monitoring is the PMU (phasor measurement unit). PMUs are used in wide area monitoring and protection schemes to compare voltage and current measurements collected throughout the grid at specific instants in time. The PMUs collect this information and transmit it with an associated timestamp to a PDC (phasor data concentrator), where data from multiple PMUs collected at the same point in time are combined to provide a snapshot of the grid condition.

The accuracy of the data packets, also known as synchrophasors, is determined by the TVE (total vector error), which is required to be less than 1 percent [104]. The error arises from inaccuracies in the initial measurement, internal processing time, and errors with the timestamp. To illustrate the importance of clock synchronization on PMU performance, a timestamp error of $26\mu s$ will lead to a 1 percent TVE. In order to maintain an acceptable TVE in the presence of the other errors mentioned above, PMU clocks should be synchronized to within $1\mu s$ of UTC. More information can be found in [154] and [104].

An important and often overlooked factor of device performance is the local oscillator within devices. These oscillators are responsible for providing the basic unit of time for the device, as well as the basis for the local approximation of global time.

While oscillators have varying accuracies depending on the material and housing, each individual oscillator will display unique drift and offsets. If devices are to be used in coordination with one another effectively, this variation in oscillator offset and drifts must be minimized. Clock synchronization techniques address this issue by periodically correcting the local estimated time to reflect the accepted global time. Current technology can theoretically synchronize standard quartz clocks to within 100 ns of the global reference within a local area network [19]. These methods will be discussed further in Section 3.3.1.

With this accurate knowledge of global time, devices can be coordinated to perform actions or take measurements at a specific instant in time. However, the accuracy of this performance is bounded by the accuracy of the device's local time estimation. For example, if a device were to apply a 1 ms accurate timestamp as a measurement is taken, the measurement could have occurred at any point within the 1 ms window. Therefore, increasing the performance of device clock synchronization can only improve measurement capabilities.

3.3.1 Performance objectives for networked clock synchronization

Advanced network protocols such as the IEEE 1588 Precision Time Protocol (PTP) provide clock synchronization within 1 μs across an Ethernet network [42]. Additional communication mediums such as fiber-optics may provide a platform to further increase synchronization accuracy. Model fidelity can also be improved as more accurate clock synchronization will lead to lower noise introduced by clock variations. The remainder of the section will detail the types of networked synchronization algorithms commonly found in industrial and power networks.

PMUs are currently synchronized to UTC (coordinated universal time) via GPS. This system is capable of providing clock synchronization within 100 ns of UTC depending on the wiring of the antenna and the availability of the GPS signal [19].

Table 3.3: Performance specifications for clock synchronization algorithms

	Communication medium	Time accuracy	Network type
GPS	Satellite	$< 100\ ns$	Planet wide
PTP	Ethernet	$< 1\ \mu s$	30-200m
IRIG-B	Dedicated network	$< 100\ ns$	Kilometers
NTP	Ethernet	$1 - 100\ ms$	$\sim$1000 Kilometers

GPS synchronization is ideal for wide area networks since no wiring is needed between locations, but does not provide economical synchronization within a local network. As the number of devices on a network requiring synchronization increases, additional antennas must be installed and the cost of the network quickly escalates. IRIG-B and PPS are often used to extend GPS synchronization to devices [19].

A promising solution is the introduction of PTP into the synchronization path. PTP is an Ethernet specific clock synchronization protocol that is capable of providing synchronization of all clocks on a local network to within 1 μs. This level of synchronization can be used to extend the time accuracy of a GPS clock received through one antenna to all devices on the same Ethernet network. Table 3.3 compares important performance characteristics of various synchronization protocols. Additional information on PTP can be found in [42] and [92].

The precision and accuracy of the PTP, like other networked clock synchronization algorithms, is compromised by asymmetric and variable packet transmission delays and processing delays in network protocol stack [115]. We performed extensive experimentation to profile and understand the practical limits on clock precision for industrial and power networks [92], [10], [5], [4]. A summary of our findings is presented in Section 3.3.2.

3.3.2 Performance evaluation of clock synchronization algorithms

We developed a testbed to establish methods for measuring and testing the accuracy and reliability of clock synchronization algorithms, as well as to characterize fac-

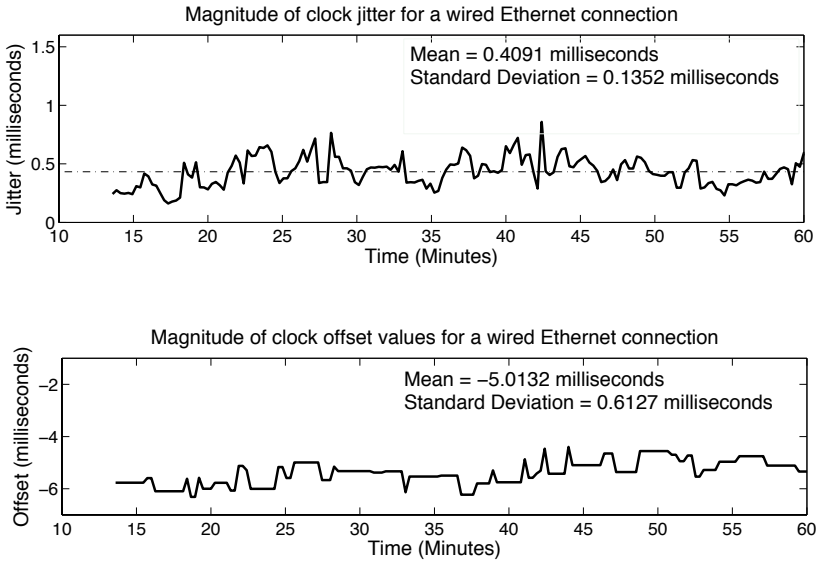

Figure 3.4: The clock offset and frequency jitter observed between two clocks synchronized using the Network Time Protocol (NTP) over a wired Ethernet connection. The network configuration for this experiment emulates a typical industrial implementation. The plot shows that the accuracy of NTP is insufficient for real-time sampling and high speed I/O requirements.

tors impacting synchronization performance using commercially available products. We conducted tests on hardware and software developed by commercial vendors. Grand Masters, Boundary/Transparent Clocks and Ordinary Clocks were tested for interoperability between manufacturers, and compliance with the objective in Table 3.3. Scenarios were designed to test several PTP parameters (such as different sync rates), or the type of topology used (star, ring, high availability seamless ring). The resulting conclusions are presented in [5], [92] and [10].

Figures 3.4, 3.5, 3.6 and 3.7 show a few highlighted results from our experiments. In the next section we will demonstrate a state estimation method that incorporates measurements of clock offset and to compensate for variable communication delay.

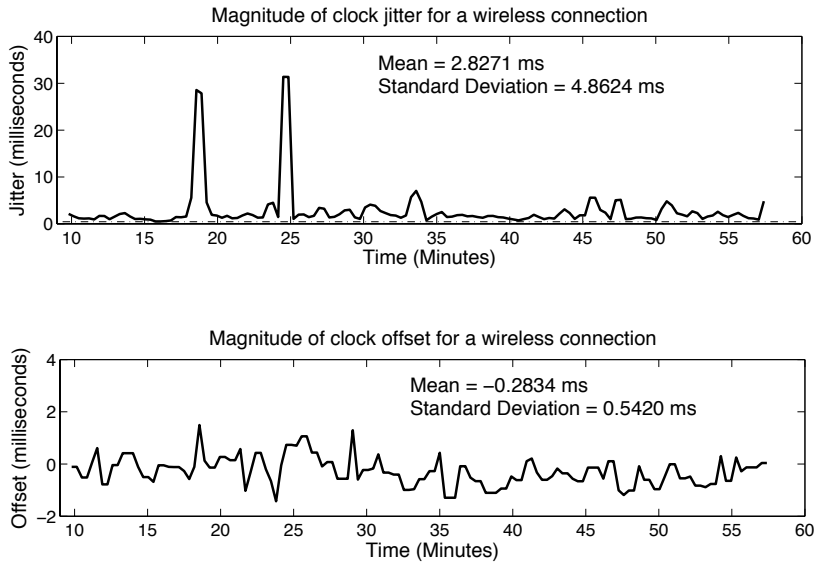

Figure 3.5: The clock offset and frequency jitter observed between two clocks synchronized using the Network Time Protocol (NTP) over an IEEE 802.11g wireless link. The network configuration for this experiment emulates a typical industrial implementation. The plot shows that the accuracy of NTP is insufficient for most applications in substation and industrial automation.

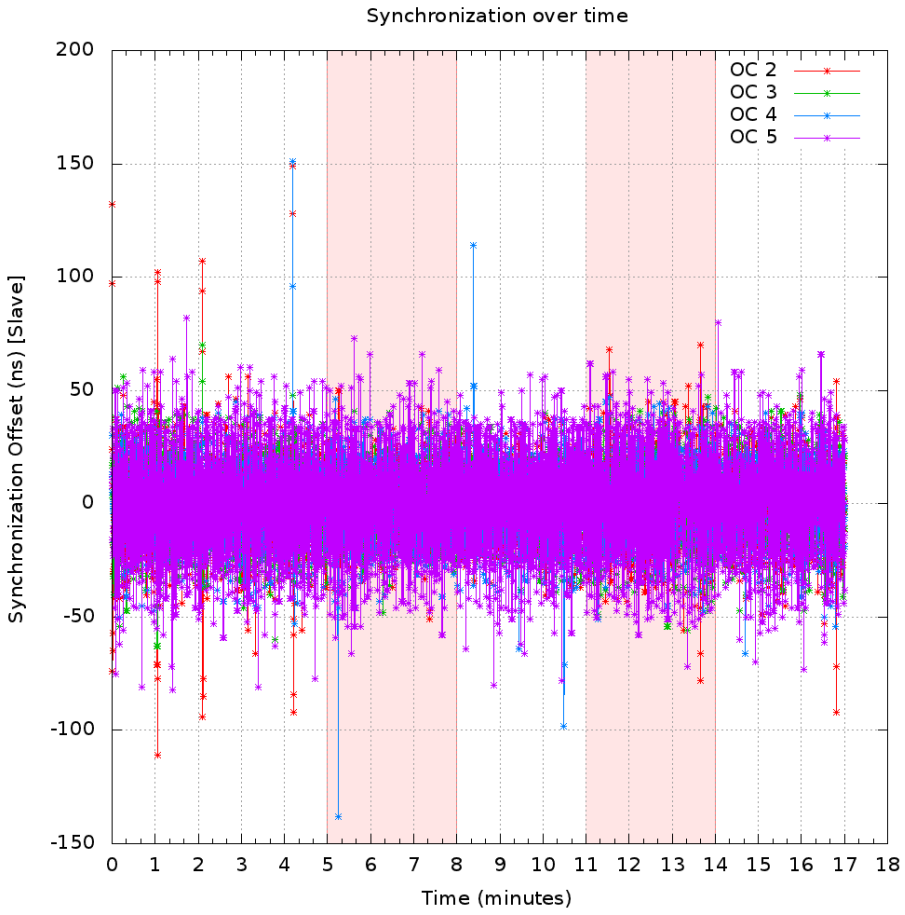

Figure 3.6: Clock offsets for four networked clocks synchronized using the Precision Time Protocol (PTP). The network configuration for this experiment emulates an electrical substation. The graph shows that under normal operating conditions PTP is able to deliver sufficient clock accuracy for even the most critical functions specified in Tables 3.1 and 3.2.

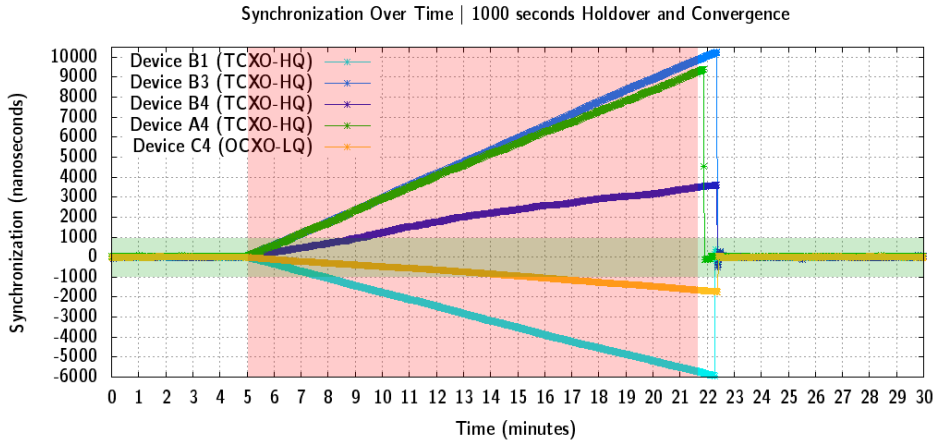

Figure 3.7: The drift of four local clocks upon loss of the synchronization signal (highlighted in green) and resynchronization after a synchronization update is received. The plot shows that the required clock accuracy for real time control and data acquisition is violated when PTP communication is lost for > 60 seconds. In 1000 seconds, the system is no longer able to support most control functions. The network configuration for this experiment emulates an electrical substation.

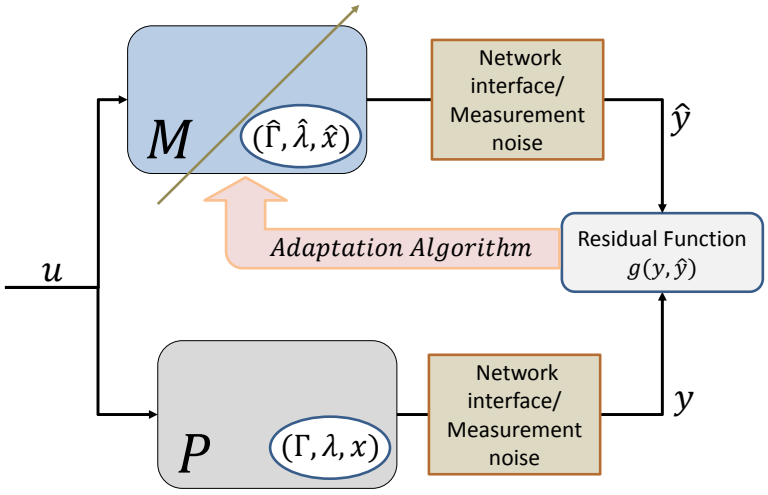

Figure 3.8: A schematic of a 2 element hybrid process. $\hat{x}$ is a model based estimate of x. The adaptation algorithm updates $\hat{x}$ based on differences between y and $\hat{y}$.

3.4 State estimation over Ethernet networks

Performance for an estimator is most often evaluated as an inverse function of the error between the estimated outputs $\hat{y}$ and the real measurements y. An adaptation law familiar to control system designers is to introduce the output error $y - \hat{y}$ into the model $\mathcal{M}$ as an additional input. Assuming that there is no network and that $\mathcal{P}$ and $\mathcal{M}$ are linear systems, the closed loop equations for this estimator design are shown in Equation 3.1.

$$
\begin{aligned}
\mathcal{P} &: \begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) \end{cases} \\
\mathcal{M} &: \begin{cases} \hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L(y(k) - \hat{y}(k)) \\ \hat{y}(k) = C\hat{x}(k) \end{cases} \\
\text{Error dynamics} &: \begin{cases} e(k) = x(k) - \hat{x}(k) \\ e(k+1) = [A - LC]e(k) \end{cases}
\end{aligned}
\tag{3.1}
$$

In this closed loop formulation, the error dynamics $[A - LC]$ are independent of the output y. As a result it can be shown that an estimator can be designed such that the magnitude of the estimation error $(e(k))$ always converges to zero given accurate measurements y and identical system matrices (A, B, C) for $\mathcal{M}$ and $\mathcal{P}$. Faster convergence can be achieved by placing the poles of $[A - LC]$.

However, faster convergence comes at the cost of increased sensitivity to noise in the measurement channel y. For a networked system, noise in the measurement channel can be caused by **variable network delay** and **imprecise clock synchronization**.

3.4.1 Review of existing methods for networked state estimation

Modeling the network as uncorrelated additive noise is a very common approach and is the basis for a majority of networked control research [98]. Many models for the 'network noise' exist in literature commonly cited ones include the Gilbert-Elliot model, the Rayleigh fading model and the Fritchman model.

If the statistical structure of the network noise is known then one can apply classical estimation techniques to networked systems. The Kalman optimal estimator is one such technique. The authors in [135] investigate Kalman filtering for systems with network noise and identify a threshold condition for the packet loss rate (due to delay or failure) to guarantee stability of the estimator. Further, the authors in [3] show that the mean-square error of a state estimate remains bounded if the average packet loss rate is bounded.

Extending the results on acceptable packet loss rate, the authors in [161] present a tradeoff between the rate at which updates are transmitted, network bandwidth requirements and the desired error bound for estimation error. In their state update strategy, a copy of $\mathcal{M}$ is implemented at the network interface module at the output of $\mathcal{P}$. This 'local copy' of the model is used to gauge the performance of the remote estimator. The network interface module compares the output of $\mathcal{P}$ and the output of the local copy of $\mathcal{M}$ and when the error exceeds a given limit it transmits the current value of y to the adaptation algorithm, hence triggering an update.

The state update mechanism in [161] as well as the results in [3] regarding the accuracy of state estimates require that all models and processes across the network be perfectly synchronized in time. In reality, clocks used by the models and processes in the network have to be synchronized over the network as well, and therefore have limited precision. In Section 3.5, we present a brief description of a proposed state estimator that is designed to counteract the presence of clock offsets and jitter between nodes.

3.5 State estimation using synchronized clocks

Clock synchronization and time stamping of measurements is useful for state estimation especially when the data is aggregated from distributed sources. As long as the time stamp resolution is less than half the measurement interval, nodes can reorder measurements when they are received out of order based upon their time stamp. In addition, measurements recorded at one point on the network are valid globally as all clocks on the network are synchronized to a single reference. State estimation algorithms can also determine the absolute time at which the measurements were obtained, instead of estimating this time based on when measurements were received. Diagnostic information about the clock offset and relative clock drift between distributed sources can be used to modify parameters in the state estimator. The effect of network jitter (variation in communication delay) can also be compensated in a real-time fashion thereby greatly improving the performance of state estimation and control algorithms operating in a cluttered, delay prone network.

We propose a networked state estimator which incorporates the IEEE 1588 Precision Time Protocol (PTP) [73] for accurate time synchronization and time stamping. The proposed state estimator is well suited for situations where the sampling interval is non-deterministic for each measurement sample and opens up avenues for truly modular control design. The design also alleviates the need for deterministic performance guarantees on the communication network.

Our control methodology is based on model generated estimates, and state and signal estimates are only as good as the local model. Since the system is designed with no a-priori model distribution step, the fundamental bounded model uncertainty can be derived from the timing uncertainty. Feeding this uncertainty through the dynamics of the model provides (assuming linear superposition) a bounded error magnitude of the estimates. Studies of stability and control performance of model based control techniques in the sub class of systems we deal with in this paper are

presented in [108].

3.5.1 Control problem formulation

Our control problem was selected to practically illustrate potential improvements in control performance with precise clock synchronization and low-level data time-stamping. The primary control goal is to track a reference phase command $r(k)$ from a remote source as shown in Figure 3.9. Additionally, slave system (Ψ) must attempt to reduce error between its output and that of the master system (Φ). Both motors have local controllers (MC_Φ) and (MC_Ψ), implemented as state feedback gains. The controllers have full state measurements and receive reference commands and communicate relative phase error over an Ethernet network. The input to the plant is voltage and the output is motor shaft position. Phase synchronization in these plants is a DC motor position control problem. Each system has an inherent pole at the origin, making it a Type 1 system. The dynamics of a Type 1 system integrate control inputs, thereby increasing the effects of jitter and measurement delay. The signal flow schematic in Figure 3.9 shows all of the delays ($\mathrm{d}_\Phi, \mathrm{d}_\Psi$ and d_Ω) considered.

The master–slave pairing presented here represents a sub–problem in a more generic class of distributed and networked systems, such as in [125]. In a peer to peer system with distributed agents this master may be dynamically selected using some form of overarching distributed consensus algorithm, similar to that discussed in [30]. In order to ensure good collective performance of the master and a cluster of neighboring slaves, it is necessary to explore methods for improving control communication to the group and within it.

The scheme presented in this section examines a strategy to mitigate network effects in a manner which is transparent to the control algorithm. Figure 3.10 shows the control algorithm using state feedback to place the poles of system Φ and Ψ to track the reference command $r(k)$.

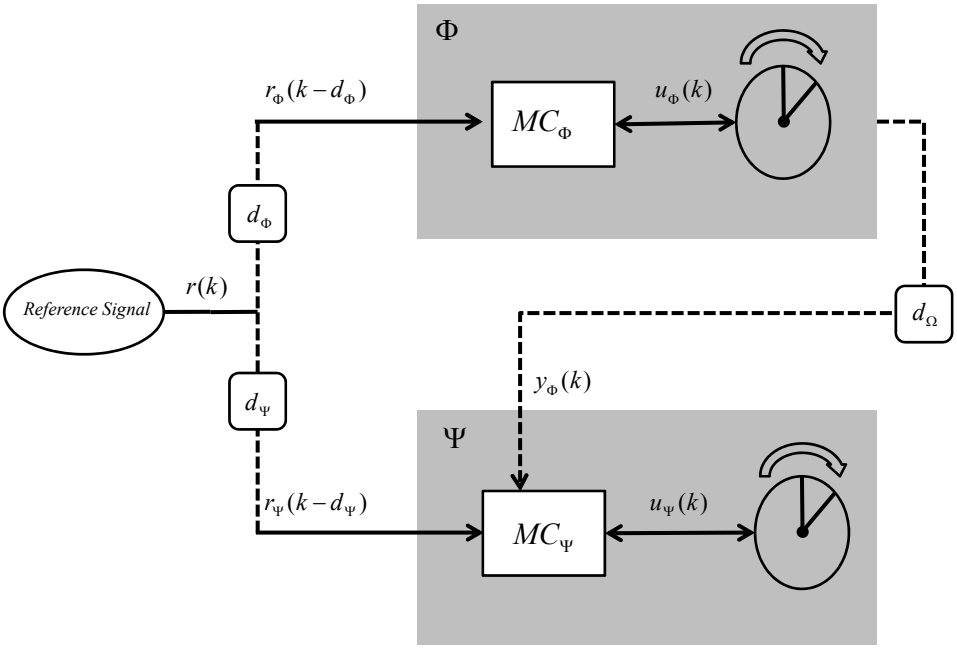

Figure 3.9: A schematic showing the control layout for the networked master–slave pair. The blocks d_Φ, d_Ψ, and d_Ω symbolize network delays between the respective components. MC_Φ and MC_Ψ are the motor controllers. $y_\Phi(k)$ is the trajectory of Φ and $u_\Psi(k)$ and $u_\Phi(k)$ are the control signals sent to the motors. Φ and Ψ denote the two individual dynamic systems.

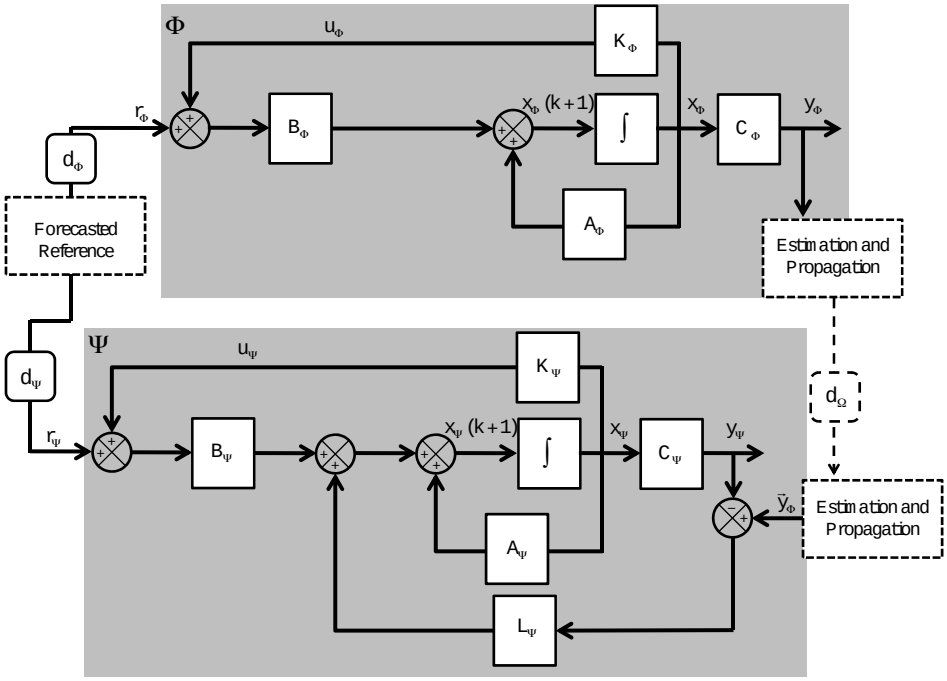

Figure 3.10: A block diagram of the control algorithm on both modules. Both controllers use state feedback gains to track the reference, the slaved system is additionally regulated through gain L_Ψ to minimize the error residual $\hat{y} - y$

It is necessary to consider the physical limits of the hardware before using state feedback to arbitrarily place poles. We tested a hardwired setup to establish the performance limits of our electronic hardware. The resulting system was used as the archetype for the state feedback controller. Equation 3.2 shows the recursive equations for the controlled dynamics in systems Φ and Ψ. Clearly, when running in isolation with zero measurement noise, no network lag, and perfectly modeled dynamics, the two closed–loop systems Φ and Ψ with feedback gains (K_Φ) and (K_Ψ) can be manipulated to have identical dynamics, represented by (A_{cl}). The output matrix C was chosen to be identical in both systems since the output position is a direct measurement of one of the state variables.

$$\begin{aligned} x_\Phi(k+1) &= (A_\Phi + B_\Phi K_\Phi)x_\Phi(k) + B_\Phi r(k) \\ x_\Psi(k+1) &= (A_\Psi + B_\Psi K_\Psi)x_\Psi(k) + B_\Psi r(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3.2}$$

Once Φ and Ψ are connected as shown in Figure 3.9 the dynamics of Ψ are designed to ensure that Ψ tracks the output of Φ. Using a state observer (L) as shown in Figure 3.10 we are able to inject additional control effort into Ψ at every sampling interval, which is a function of the error residual $y_\Phi - y_\Psi$ or correspondingly $C(x_\Phi - x_\Psi)$. The dynamics driving the evolution of the error residual, e(k), are shown in Equation 3.3. As shown in the equation, the separation principle still holds and the poles of the system driving the error dynamics can be placed independent of the closed loop poles. The observer dynamics presented here, similar to the control design, do not consider the impacts of d_Φ, d_Ψ, and d_Ω, or their stochastic nature.

$$
\begin{aligned}
e(k) &= x_\Phi(k) - x_\Psi(k) \\
e(k+1) &= x_\Phi(k+1) - x_\Psi(k+1) \\
e(k+1) &= [(A_\Phi + B_\Phi K_\Phi)x_\Phi(k) + B_\Phi r(k)] \\
&-[(A_\Psi + B_\Psi K_\Psi - LC)x_\Psi(k) + B_\Psi r(k) + LCx_\Phi(k)] \\
e(k+1) &= (A_{cl} - LC)x_\Phi(k) - (A_{cl} - LC)x_\Psi(k) \\
e(k+1) &= (A_{cl} - LC)e(k)
\end{aligned}
\tag{3.3}
$$

In our testbed we combine two strategies to mitigate the impact of network delays; one is a method of forecasted waypoints to improve the quality of the reference command, and the other is a strategy of inline estimation and propagation used for the observer. A schematic diagram of the enhanced system is presented in Figure 3.11. We use the same control system for the new setup to study the improvement in performance. We assume that the sampling process $[k, k+1, ...]$ is synchronous over all distributed elements. This simplifies the presentation of equations and analysis, but is not a necessary assumption. With time stamping there are several solutions for re–sampling, ranging from linear interpolation to model based smoothing [99].

3.5.2 Strategy 1: Reference forecasting

We assume for the purposes of this testbed that the reference input is a tabulated a-priori set of points which the controllers must track. This assumption comes from a view that topological hierarchy is necessary in large distributed systems to ensure proper scaling. This hierarchy is established from a performance perspective as well, in that often times a supervisory controller manages several sub-controllers, and has sampling time or a time constant that is significantly larger than that of it's sub-

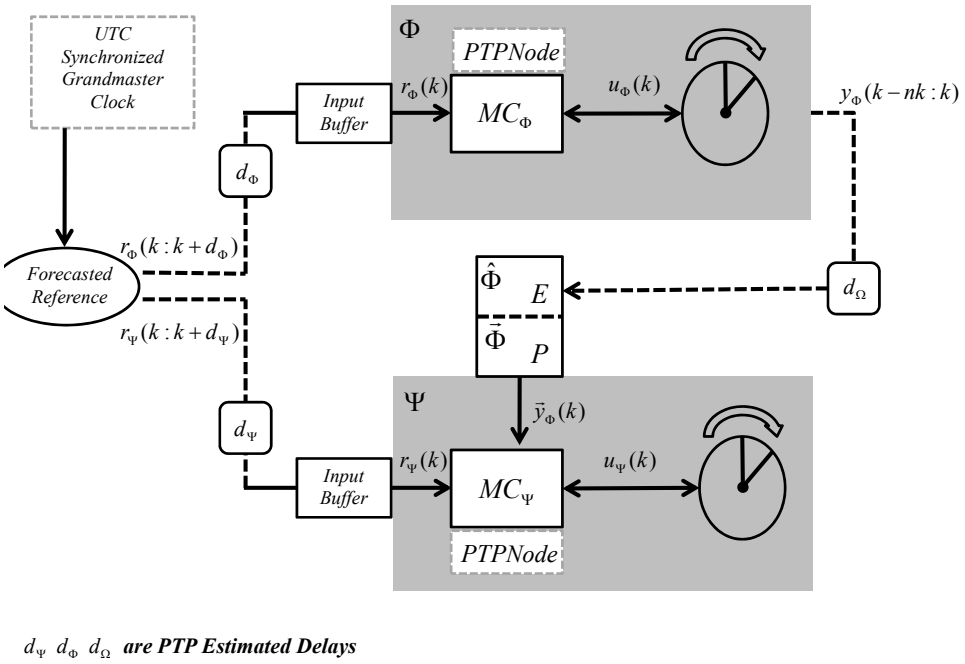

Figure 3.11: A schematic of the same control layout shown in Figure 3.9. Modules E and P denote the Estimation and Propagation blocks added to compensate for delays d_Φ, d_Ψ, *and* d_Ω.

processes. This simplifies the global control design problem into modular components where the assumption is that the dynamics are decoupled between tiers (sub-processes are at steady state before supervisory action is taken). If there were a need for the supervisor to regulate the dynamic state of the sub-controllers, then the use of Model Predictive Control is warranted [98].

In the current case, the problem of random delays in the reference signal is compensated for by using an input buffer on the individual control modules and aggregated data transmissions to systems Φ and Ψ of the form $[r(k)...r(k+d_\Phi)]$ and $[r(k)...r(k+d_\Psi)]$ with corresponding time-stamp vectors $[t|_k...t|_{k+n})]$. Our discretization strategy is shown in Equation 3.4.

$$r|_{t=UTC\ at\ poll}(k) = \begin{cases} r(t \pm \delta t) & \text{if measurement is present} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

While this is a conceptually simple solution, it is difficult to implement physically due to the unavailability of estimates for d_Φ and d_Ψ. With the PTP implementation we are able to draw estimates of the network delay between any two network components by comparing the delay values computed by the PTP algorithm. The PTP algorithm uses a time calibrated peer to peer probe message to estimate the network delay. The algorithm uses the delay estimates to compute the relative clock offsets between node clocks. We intercept the delay estimation messages from the PTP network modeling service and use them to estimate reference delays. To accommodate differences in state since the last PTP update cycle we use additional points in the reference table to get $[r(k)...r(k+d+nTs)]$, where Ts is the sampling time. n is automatically updated based on network conditions. Judging from delay distribution presented in [9], $n = 5$ was found to be the average margin for the testbed network. The input buffer module records these vector tables and presents a valid $r(k)$ on being polled by the control recursion.

3.5.3 Strategy 2: Estimation and propagation

We adopt a two step process to introduce synchronous estimates for y_Φ into Ψ. At the first step we generate a time stamped vector table of input and output values from Φ, $[r(k-nk)...r(k)]$ and $[y_\Phi(k-nk)...y_\Phi(k)]$ where $y_\Phi(k)$ is the current output measurement and $y_\Phi(k-nk)$ is an output measurement from n samples ago. n is picked to ensure a non-singular regressor matrix [99] for the system identification process. In our system, this value is manually fixed to be equal to the clock drift sampling interval ($\sim$ 200 ms). Sample selection for system identification is explored in significantly more detail in [99]. As this vector table is transmitted to the Estimation and Propagation module (which is physically co-located with the slave system), it incurs a delay d_Ω. We use this time stamped vector table for parametric system identification with an assumed linear 2^{nd} order structure. An Auto-Regressive Moving Average (ARMA)

model [99] is used to identify the system parameters and generate an identified model $((A_\Phi \widehat{+ B_\Phi} K_\Phi)\,, \hat{B_\Phi})$. We do not consider the case where individual samples are lost since they are packaged in vector tables. For a case where individual samples might be lost, the authors in [14] present a strategy for managing lost samples during system identification over unreliable networks. To populate the regressor matrix we use a similar policy as in the input buffer mentioned in Section 3.5.2. A comparison of the output of the identified system using the tabulated vector time table against a one with streaming data from Φ is shown in Figure 3.13. Once the identification model is updated, we have a local model of Φ at Ψ validated against a measured dataset up to $\hat{y}_\Phi(k - d_\Omega)$. We use this estimation model $(A_\Phi \widehat{+ B_\Phi} K_\Phi)$ to propagate y_Φ forward to $\vec{y}_\Phi(k)$ using the recursion shown in Equation 3.5. The structure of the output matrix C can be easily converted using similarity transformations to coincide with our assumptions in Equation 3.3. The inputs $[r(k - d_\Omega)...r(k)]$ required for the propagation module are pulled from the input buffer for system Ψ.

$$\begin{aligned} \vec{x}_\Phi(k+1) &= (A_\Phi \widehat{+ B_\Phi} K_\Phi)\vec{x}_\Phi(k) + \hat{B}_\Phi r(k) \\ \vec{y}_\Phi(k) &= C\vec{x}(k) \end{aligned} \tag{3.5}$$

The evolution of output signals as they move through the Estimation and Propagation modules are summarized in Equations 3.6 and 3.7 respectively. The resulting closed–loop dynamic equation for the slave system Ψ is presented in Equation 3.8.

$$y_\Phi(k - d_\Omega - nk : k - d_\Omega) \xrightarrow{E} \hat{y}_\Phi(k - d_\Omega) \tag{3.6}$$

$$\hat{y}_\Phi(k - d_\Omega) \xrightarrow{P} \vec{y}_\Phi(k) \tag{3.7}$$

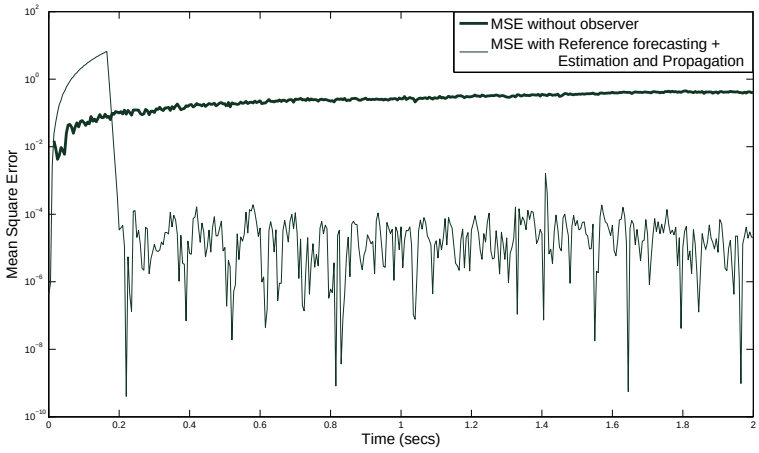

Figure 3.12: A simulated result showing the Mean Squared Error between Master and Slave systems with and without an observer.

$$x_\Psi(k+1) = (A_\Psi + B_\Psi K_\Psi)x_\Psi(k) + B_\Psi r(k) + L(\vec{y}_\Phi(k) - y_\Psi(k))$$
$$x_\Psi(k+1) = (A_\Psi + B_\Psi K_\Psi - LC)x_\Psi(k) + B_\Psi r(k) + L\vec{y}_\Phi(k) \tag{3.8}$$

3.5.4 Results

Figure 3.12 shows a simulated comparison of the Squared Tracking Error between Φ and Ψ when they are running in isolation and when the proposed estimator is introduced. The plot shows significantly reduced error in the latter case (close to two orders of magnitude lower) after the initial spin–up phase of the estimator. The large error during spin–up is attributed in part to our aggressive placement of observer poles and the initial clock offset between Φ and Ψ. Figure 3.13 shows the output of Ψ using a state estimator with and without time stamped measurements. The plot shows that time stamps improve estimator performance.

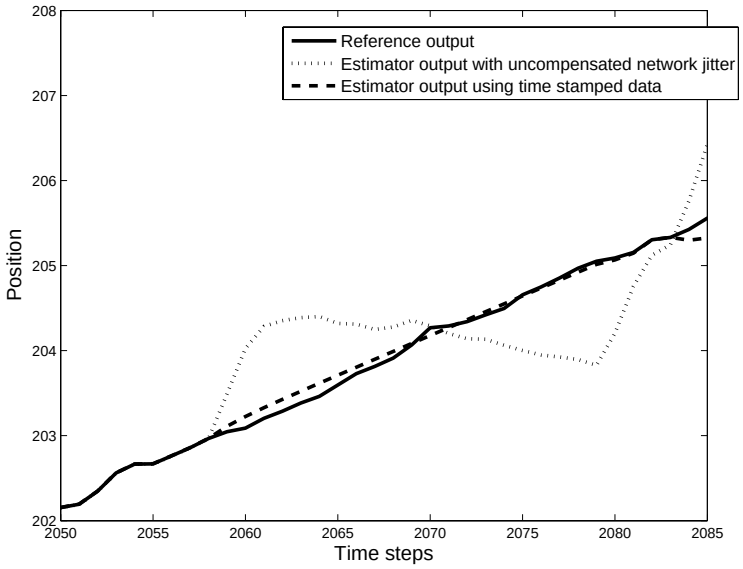

Figure 3.13: A simulated result showing the improvement in estimator performance when the input data (data used for system ID) is time stamped.

3.6 Summary

In this chapter we discussed the limitations of networked state estimation and outlined two contributing factors, variable network delay and inaccurate clocks. We presented a summary of our experiments to profile network delays and jitter for typical industrial networks and substation networks. We then presented empirical studies of clock synchronization algorithms. Using the results from our experiments we proposed a modified networked state estimator which not only considered the additive noise due to network jitter but also compensated for delay on each network packet by estimating the clock offsets between networked nodes.

Our experiment shows that, as long as precise time stamps and clocks are available, the results of [3] are valid for large substation networks and industrial networks. That is, the stability of a linear state estimator only depends on the gross update frequency. As long as the analytically computed update frequency is met, network jitter, clock drift and clock offsets may be compensated individually to cumulatively improve the accuracy of the state estimate.

The work presented in this chapter demonstrates the use of a simple semantic definition relating the clock offset between nodes in a network to design parameters of the state estimator. Ordinarily, clock synchronization and networked state estimation are addressed independently and are not designed to interact. We show that providing limited access to certain variables used by the clock synchronization algorithm (clock offset estimates, syntonization heartbeat counters and peer to peer delay estimates) greatly improve the performance of a state estimator and open up new design options for compensating network delay.

CHAPTER IV

Model adaptation methods using model semantics

The work presented in this chapter appears in proceedings of the 2010 *IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication* and the proceedings of the *2011 ASME Dynamic Systems and Control Conference* ([7], [6]).

4.1 Introduction

In this chapter we present two case studies where semantic assertions are used for model adaptation. In particular, assertions connecting the performance of a model to key structural attributes are used to design algorithms to adapt the structure of a model.

In the first case, system level assertions are made relating the order of the model to the system level performance of a hybrid process. Factors such as the accuracy of sensors, the accuracy of clock synchronization and cost of network utilization are used to optimize the choice of model order.

In the second case, the problem of reconfiguring the structure of a model is considered when the model made up of discrete model components. The structure of the model $\mathcal{M}$ is updated so that its output matches the output of the corresponding physical process $\mathcal{P}$. The combinatorial scale of the reconfiguration problem is reduced

by including syntactic constraints on each model component. Further, semantic assertions are used to evaluate and rank each feasible reconfiguration so as to find the 'best' reconfiguration that resolves any difference between the outputs of $\mathcal{M}$ and $\mathcal{P}$.

Both cases are presented using illustrative examples of hybrid processes in the electrical power network.

4.2 Modeling and control for electrical networks

The present and future operations of power networks rely on critical advances in wide area monitoring and control. Advances in communication and network topologies have made it possible to aggregate information from many nodes separated by vast distances in a timely manner. Improvements in computational power and speed have increased the complexity of simulation models that can be used for control. The convergence of these two factors means that a distributed system such as the power grid can be controlled over a network with unprecedented accuracy.

A key enabler in this transition towards advanced timing requirements is the phasor measurement unit (PMU). The following is a brief list of PMU specifications, more information can be found in [154] and [104]:

- The PMU is a high fidelity sensor capable of sampling voltage and current waveforms at rates up to $10,000$ Hz.
- The synchrophasor is a vector measurement that is reported at a rate of up to 60 Hz.
- These compiled reports are time-stamped, accurate to within 1 μs of coordinated universal time (UTC).
- Synchrophasors from multiple PMUs and their corresponding time stamps together provide a snapshot of grid conditions.

Mathematical modeling of the energy grid is another key enabler that, when used in conjunction with accurate sensing and network communication, is vital for improved analysis and control. An example of models and PMUs working in unison is model based estimation [84]. The Smart Grid Interoperability Standards Roadmap published by the National Institute of Standards and Technology [44] lists several challenges in controlling widely distributed electrical distribution assets and lists several requirements for the next generation grid control infrastructure. One of the highlighted aspects in the roadmap is the need for **accurate real-time models** and **standardized model design**.

4.2.1 Accurate real-time models

Since an electrical network can contain hundreds to tens of thousands of components depending on the application [22], it is impractical to solve full-model equations in reasonable time for real-time model integration. Commonly, a reduced model is used for real-time control and estimation functions. The reduced model is simple enough to meet the speed of simulation required for control, but also reflects enough of the characteristics of the system to be useful. Automated or assisted model order reduction/deduction is a powerful tool to reach this optimal choice for model order. The transmission line example described in Section 4.3.2 demonstrates a system where the starting point is a simple model that can be scaled up in model order automatically until a performance bound is reached.

4.2.2 Standardized model design

The Smart Grid Interoperability Roadmap specifically addresses the need for a standardized modeling approach for the diverse set of resources connected to an electrical network. As a specific case, the roadmap discusses the advent of small distribution circuits with self contained demand response and generation capabilities.

Such circuits can be found in university campuses, large manufacturing plants and some residential neighborhoods. They are commonly called a **microgrid** [125, 89]. A network of microgrids may be connected to a larger utility grid which is an electrical network managed by a regional power transmission and distribution service. The microgrid exports or imports power from this larger network, but is also able to disconnect from it and run independently. Keeping with the philosophy of microgrid autonomy, control design within the microgrid is not standardized by the utility. The interface between the microgrid and the utility grid however needs to be tightly regulated. In order to strike this balance, the utility must be able to estimate the internal state of a microgrid when it is connected to the grid. Real-time measurements from the microgrid are at a premium since the network infrastructure cannot support a wide area implementation where all the data from every microgrid resource is published out into the wide area network. The utility therefore uses a model for the microgrid to estimate the internal state. If the model of the microgrid is made up of standard model components, the utility is better equipped to accurately estimate its internal state. The structural adaptation method in Section 4.4.4 demonstrates how the model of the microgrid (made up of standard components) may be automatically updated when changes are made to the physical structure of the microgrid circuit.

4.3 Using clock accuracy to guide model synthesis in distributed systems

4.3.1 Model Order Deduction

In order to pose model synthesis of a power transmission line as a problem for automated modeling, we will first define the order deduction process in spirit. A power transmission line can be expressed as a spatially distributed vector field of voltages and currents. These voltages and currents are subject to transformations

through a system of differential algebraic equations.

A common modeling simplification used for transmission lines is to use a lumped parameter formulation, where the transmission line is expressed as a finite set of interconnected "capacitive, inductive and dissipative" elements as shown in Figure 4.1. Networks of these elements can be assembled into discrete units representing other power system components such as busses, switches, etc. A further simplification is to say these functional blocks are linear and the parameters are time-invariant. These assumptions are common in electrical system modeling as seen in [57]. The authors in [47] discuss a few example cases of model reduction for non-linear circuits. The assumption of lumped linear models does not detract from the motivation or the algorithmic approach, but allows us to use the order of coupled ordinary differential equations to mathematically express "complexity".

Model order deduction is an automated approach to finding a 'proper' model. The basic strategy is to start with a simple model for a dynamic system and then to iteratively add complexities (model order in linear systems) until a stop condition is reached. Several exit conditions and details about the basic algorithm are presented in [80], [153] and [144]. For a linear model of the transmission line, states are added to the state space model in the form of generalized inductive and capacitive elements at each node and in the interconnects until a proper model order is reached.

We use the frequency domain model order deduction algorithm (FD-MODA) presented in [153] to deduce a proper model. We first determine a frequency range of interest (FROI). The authors in [80] specify that the model should be accurate at frequencies 2-6 times the maximum input frequency. The maximum input frequency ω_{in} in our case is the upper limit of the envelope of frequencies input to the transmission line. For this paper we will set this at $3000Hz$ based on the work presented in [77] and on the fundamental limits of the sampling process in a PMU. The upper limit for the FROI is $\omega_{max} = 15kHz$. According to the FD-MODA algorithm, a proper

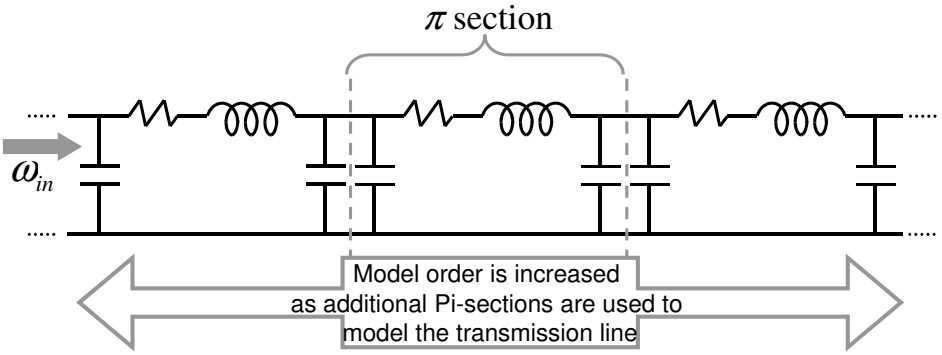

Figure 4.1: Schematic representation of the π-model for a single phase power transmission line. The center section represents a single π section, and many π-sections in series model the transmission line.

order r is reached when model order $r+1$ does not appreciably change the frequency response, G, of the system within the FROI. As the lumped parameter models are divided into finer and finer submodels, a threshold is reached when the frequency response and continuum model of the system are close enough. Equation 4.1 gives the stop condition for the FD-MODA algorithm.

$$\delta G_n^r = \max_{\omega \in \text{FROI}} \left| \frac{G^{r+1}(j\omega)}{G^r(j\omega)} - 1 \right| < TOL_m \tag{4.1}$$

where, $\omega \in FROI = [0, \omega_{max})$ and the frequency response convergence tolerance TOL_m is picked as necessary. In our case $TOL_m = 0.01$, or 1% error in the model frequency response. Further, if the model of the system is made up of m connected components with ranks $[r_1, r_2 ... r_m]$ FD-MODA increases the rank of the system model iteratively while minimizing $r = \sum r_i$, i.e. increasing the rank of the most sensitive component before the others. n is the highest possible model order (or rank) of G.

4.3.2 Use Case

The linear system presented here is that of a power transmission line modeled as a π-model. Figure 4.1 shows the scalable π-model of the transmission line. The number of sections (highlighted in the schematic) can be arbitrarily scaled up using

the FD-MODA algorithm. This is an ideal candidate for an example since the model properties approach continuum behavior as it is discretized into finer and finer π-sections. The system parameters (inductance, capacitance and impedance) for each π-section also change as the order is increased. We use the MATLAB SimPowerSystems simulation package to develop our models. Since we are only modeling one homogenous element, we can directly attempt to meet the condition in Equation 4.1 without having to run through the iteration step to find the most sensitive sub-model. If there were other components in the system such as capacitor banks, filters, loads and generators, then the general class FD-MODA algorithm discussed in [153] may be used.

An example use case highlighting the need for a FD-MODA optimized π-model is in the estimation of the state of a feeder line between shared transmission assets where the PMUs are only available at the terminal nodes. Consider an application as shown in Figure 4.2 where a 10 kilometer long 26.5KV transmission line connects two substations both with captive spinning reserves (small generators designed to compensate for transient load fluctuations). The closed loop control of these generators is achieved through measurements from the local PMU and estimates of transmission line state and the state of the remote substation. The switching station located somewhere along the transmission line switches in the remote generator when required. This switching event sets off a transient state on the feeder line which is measured by both PMUs at the local and remote end. The standing practice to prevent a parasitic oscillation resulting from this perturbation is to estimate the effect of this event on the transmission line, which requires an accurate representation of the transient phenomenon across the transmission line. Since it is infeasible to string phase sensors over the entire length of the feeder line to build this estimate, a mathematical model of transmission line is used instead in conjunction with available sensor data to build a model-in-the-loop estimator. In such a case, the model of the transmission line

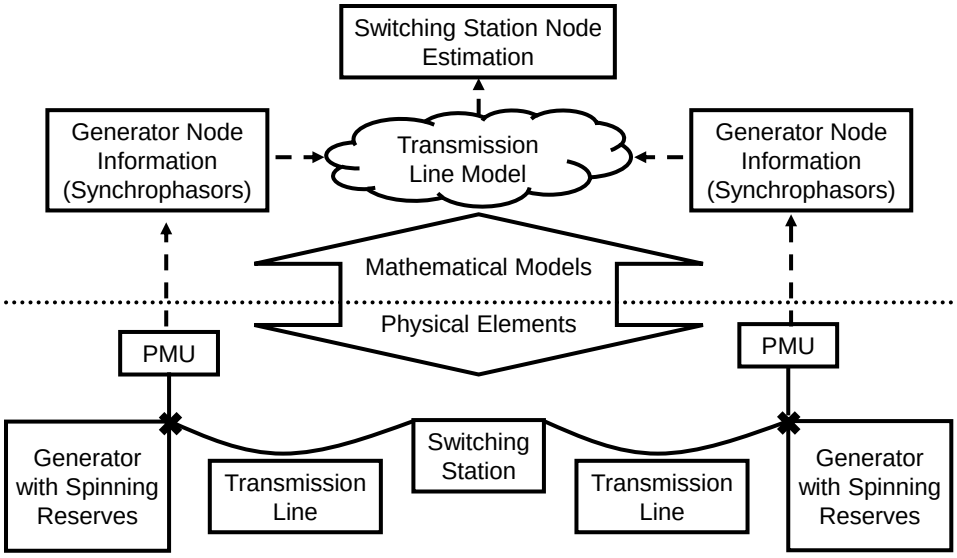

Figure 4.2: Schematic representation of a hybrid process with coupled interaction between physical elements and mathematical elements. PMUs provide physical measurements at the terminal ends of a transmission line connecting two substations. A model of the transmission line is used in conjunction with physical measurements to provide an estimate of the state of the two generating substations.

becomes critical in monitoring the state of the transmission line. These models must account for the electro-magnetic properties of the transmission line and the methods for sampling the line. Since the performance of this model is subject to the same data quality and reporting constraints as the sensors in the network, clock performance begins to play a role in the model performance as well. For example, any noise (clock error) in time stamping process at the switching station would manifest as errors in the model generated estimate of the state of the remote node. Similarly, the loss of clock precision on the PMU sampler would corrupt the correlation between the real PMU measurements and the modeled signal. Ideally this information must be incorporated into the model in order to optimize the hybrid performance of the collective system.

4.3.3 Building models with timing constraints

With the pervasive use of switched IP networks for the current implementation of grid control, there is growing interest in using IEEE 1588 Precision Time Protocol (PTP) within power system networks to synchronize clocks. PTP could potentially provide the necessary accuracy for synchrophasor measurements from PMUs as discussed in Chapter III.

In order to integrate mathematical models into a network architecture with network clock synchronization, we first need to be able to express the dynamic clock uncertainty in the network as a form of process noise in the mathematical model for the integrated system. Some strategies for introducing sampling effects and communication corruption into the modeling of distributed systems are presented in [160] and [82]. A form of frequency bounded stochastic truncation can be used to identify process uncertainty close to the sampling frequency. While this is a useful tool for controller design, we still have to understand the stochastic properties of clock error and translate that to uncertainties in the model.

Since we are using the FD-MODA algorithm for model synthesis and we are only interested in the maximum difference in the frequency response, we will attempt to treat the timing inaccuracy as a form of "noise" injected at the output of the model. The authors in [160] show that superposition holds true for linear systems with random time delays; we can therefore introduce a time corrupted form of the input signal to an identical system model to yield the 'noisy' output y_n^{r*} and then claim that $\|y_n^{r*} - y_n^r\|$ is a suitable measure of output side sensitivity to clock uncertainty. To stay true to the FD-MODA algorithm we will use frequency response over the range of interest as the sensitivity metric for $\|y_n^{r*} - y_n^r\|$. To achieve this we use the exit condition shown in Equation 4.2. TOL_c is the allowed tolerance for change in frequency response of the system due to inaccurate data reconstruction with given time stamp accuracy. In our case $TOL_c = 0.0025$ or one quarter of the tolerated

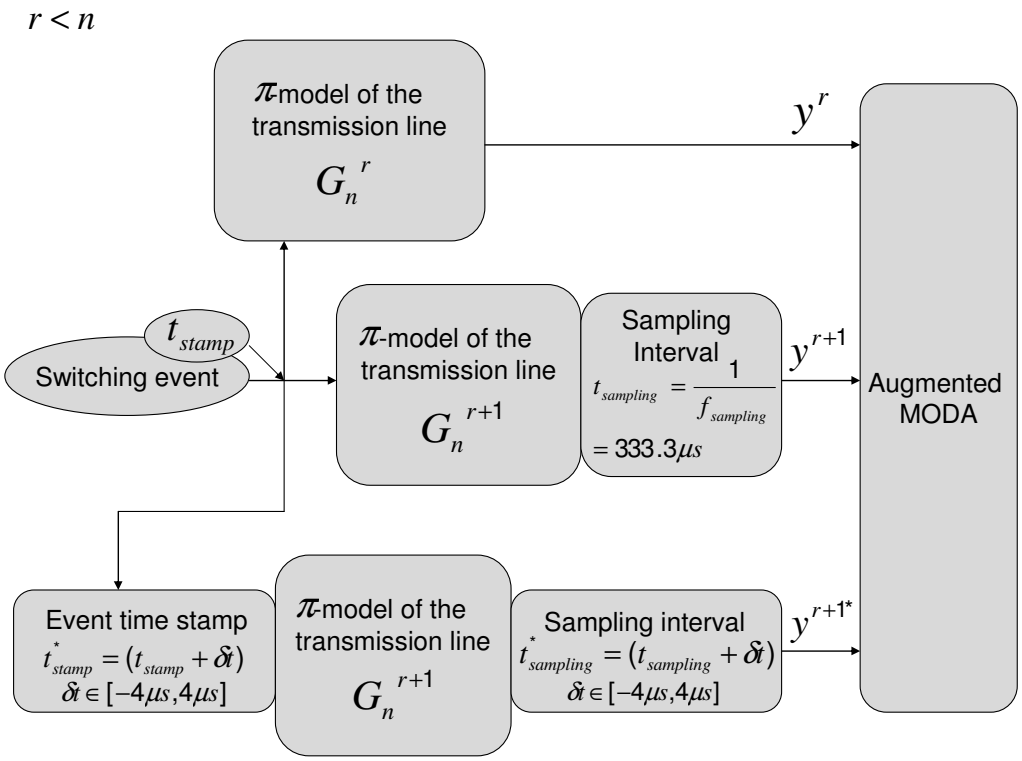

Figure 4.3: Schematic representation of the decision inputs when clock accuracy is added to the MODA system. The current model order is r, such that $r < n$ where n is the maximum model order available.

model error. The choice for TOL_c is a design parameter and picked in this case based on practical experiments involving model based estimation on clock synchronized controllers discussed in [8].

$$\delta G_n^{r*} = \sqrt{\frac{(G^{(r+1)*}(j\omega) - G^{r+1}(j\omega))^2}{(G^{r+1}(j\omega))^2}} > TOL_c \tag{4.2}$$

4.3.4 Results

We use the FD-MODA algorithm on a π-section model of a power transmission line introduced in Section 4.3.2. The π model converges to a high fidelity representation of the phase and frequency dynamics of a real life conductor carrying AC waveforms (within TOL_m). To decide on a satisfactory level for this granularity of division, we use an augmented form of the FD-MODA technique [80]. We perform the increased order sensitivity test to find if increasing the model order makes a significant contribution to the frequency response characteristics of the system. Additionally, we perform another test to check if the added model order ends up reducing the effective performance of the system because of loss in frequency response precision as the system approaches the limits of clock accuracy. This clock accuracy is a function of synchronization method used as discussed in Chapter III. Figure 4.3 outlines our simulation approach used to introduce clock inaccuracies in the model integration process where we attribute a random delay in the interval $[-4\mu s, 4\mu s]$ to both the switching event, which excites the dynamics of the transmission line, and the sampling interval within the PMU to simulate drift in the PMU sampling clock. Details are discussed in the subsequent sub-sections.

4.3.4.1 Model response with changing model order

The transmission line model described in Section 4.3.2 was subjected to a 265KV 60Hz AC input waveform. Figures 4.4-A and 4.4-B show the input output characteris-

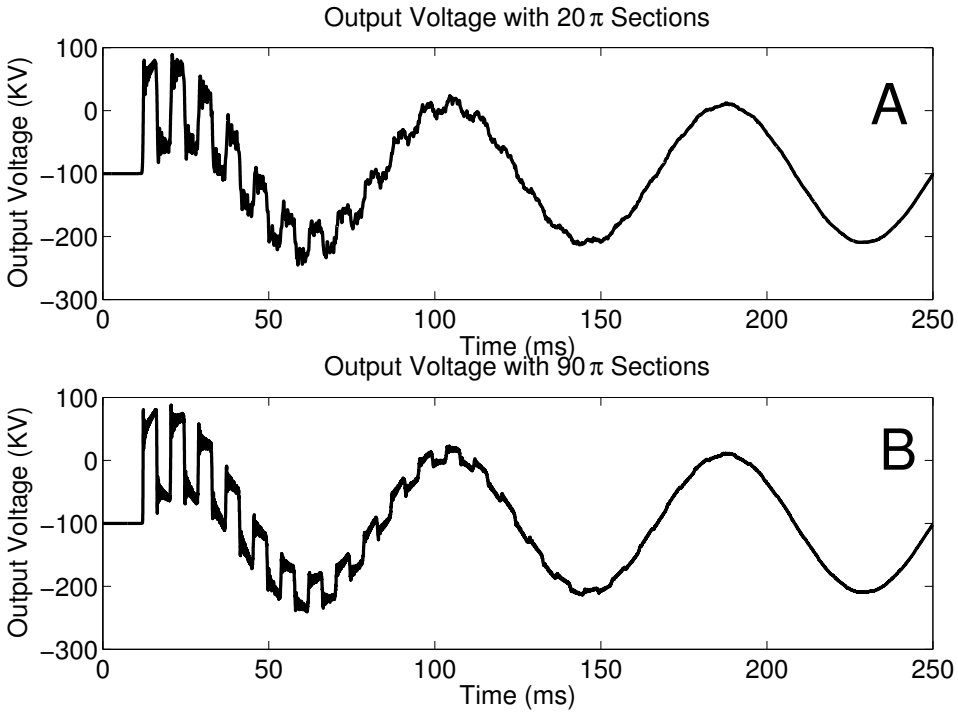

Figure 4.4: 265 kV, 60 Hz transmission line voltage waveforms under circuit breaker closure condition. The output response presented in plots A and B correspond to models using 20 and 90 π-sections respectively.

tics of the transmission line model with 20 and 90 π-sections respectively. Both plots show the response of the model to a circuit breaker "close" event at 0.002 seconds. Figures 4.5-A and 4.5-B show the magnitude spectrum for the frequency response of the 20 and 90 π-section models respectively. As the model order is increased, the response to higher frequencies is significantly increased, which is especially visible on Figure 4.5 close to 100KHz. The FD-MODA algorithm was applied to the transmission line model with FROI=15 KHz in order to meet the desired tolerance TOL_m. The model order was iteratively increased by the algorithm until the error norm of the frequency response between order r and $r+1$ was less than 1%, which occurred on the addition of the 86th π-section. Figure 4.6 shows that there is also a significant improvement in model performance within the FROI with increasing order.

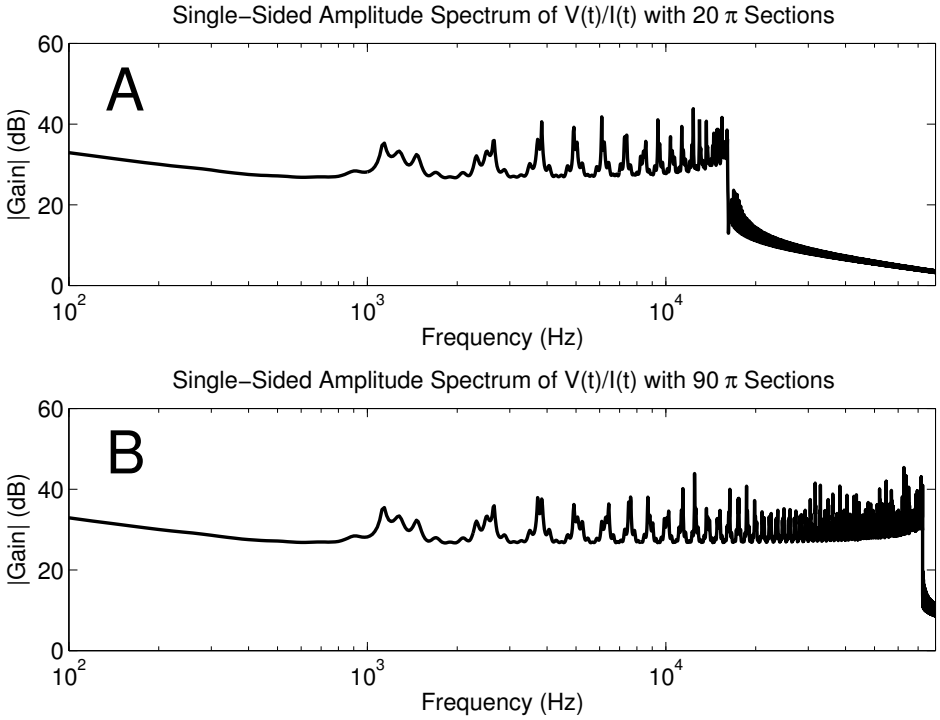

Figure 4.5: Frequency response of models using 20 π-sections (Plot-A) and 90 π-sections (Plot-B). Plot-B shows that the model with 90 π-sections has a higher gain at higher frequencies.

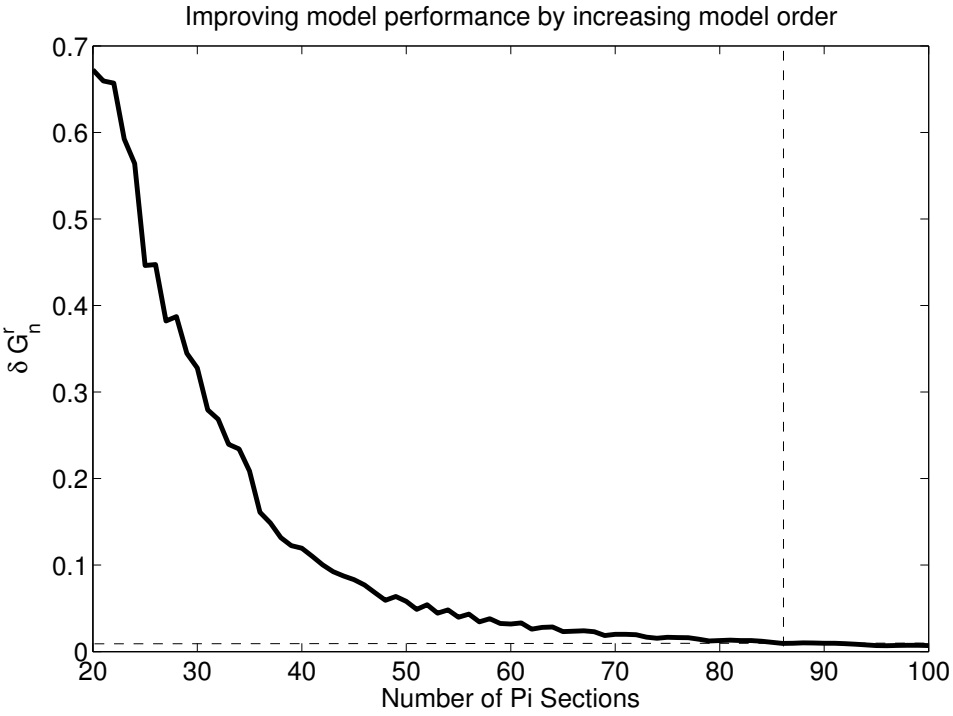

Figure 4.6: Max norm of MODA algorithm applied to the transmission line model. The norm drops below a tolerance of 0.01 at 86 π-sections

4.3.4.2 Model response with timing uncertainty

The simulations discussed in Section 4.3.4.1 do not account for the possibility of time corruption in the process of reporting and sampling data. Uncertainty in the exact time at which the switching event occurred and uncertainty in the trigger signal for the sampling process were added to the simulation. The assumption is that as the model order is increased the model exhibits increased sensitivity to clock corruption. This phenomenon is confirmed in Figure 4.7 where we see a growing error norm due to clock uncertainty as the model order is increased. Intuitively, the higher order models have a larger spectral radius and therefore are more sensitive to time uncertainty. The error norm in the figure is calculated assuming a time-stamp error between the limits $[-4\mu s, ..., 4\mu s]$. This range was chosen based on the nominal performance of the precision time protocol (PTP) presented in Chapter III and based on observations made on wide area implementations of PTP presented in

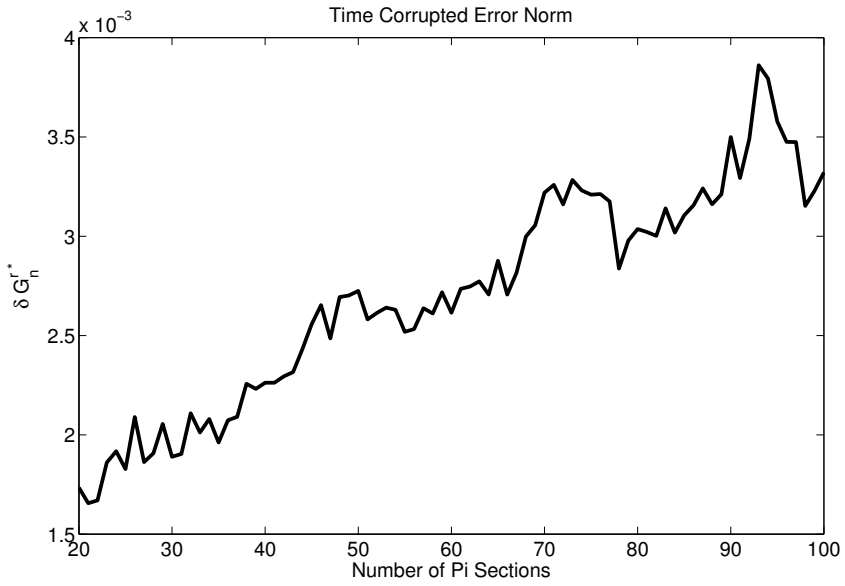

Figure 4.7: Error norm due to clock uncertainty in sampling and time-stamping. The clock errors are assumed to be normally distributed with $\sigma = 4\mu s$ The norm exceeds 0.025 at 48 π-sections.

[107]. The authors experimented with PTP over an undersea network connecting ocean observatories and off shore sensors spanning an area of tens of kilometers. The results in [107] show that at steady state the system offsets have a zero mean with frequent offset corrections of up to $\pm 4\mu s$.

We then invoked the FD-MODA algorithm with an additional exit condition represented by Equation 4.2 to ensure TOL_c is not violated. The resulting deduced model had 48 π-sections, showing that it is not possible to meet both given performance parameters TOL_m and TOL_c. The designer is now faced with the choice of either improving the timing accuracy within the network or to compromise on the desired model fidelity. In this use case, a model with 48 π-sections satisfies TOL_c but has a model error of about 1 percent, which is ten times the desired value for TOL_m. Applications which might require higher fidelity models also mandate much tighter synchronization to support the model complexity.

From the perspective of optimal design it is convenient to integrate δG_n^r and δG_n^{r*}

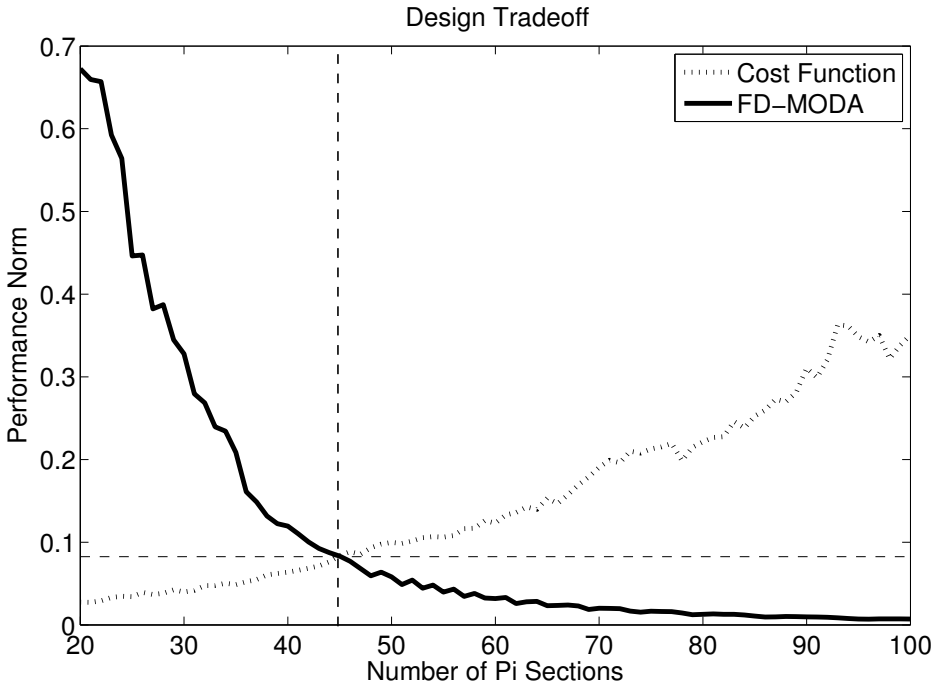

Figure 4.8: Optimal model choice the number of π-sections for the model of the transmission line is a tradeoff between the FD-MODA algorithm (δG_n^r) and the cost function related to timing uncertainty ($\delta G_n^{r*} \times T_{sim}$).

into a single performance metric. This is a very application dependent choice, but in the case of our example we decided to penalize increasing model fidelity against the product of the growing error norm due to clock uncertainty and increasing computing cost expressed as simulation time in seconds T_{sim}. The optimal choice is the intersection of the trajectories of δG_n^r and $(\delta G_n^{r*} \times T_{sim})$. Figure 4.8 shows this design tradeoff, indicating that the optimal design for our use case is a model with 45 π-sections.

4.4 Semantic models facilitate updating model structure

For large scale modeling, especially in the case of the electrical network, models are built out of assemblies of sub-models. A model for a voltage transformer for example is made up of an assembly of an ideal transformer, several inductances, resistances and non-linear hysteresis models. Knowledge about the exact arrangement of these components in the transformer model allows us to identify the value of the internal

parameters and to update the correct state when needed.

The use of commercial models makes it difficult to be sure about the internal structure of the model. It is often required that we first identify the internal model structure based on its input-output properties. There is well established research in the domain of model structure identification, the domain covers the area of fuzzy logic [94], neural networks [12, 97] and other model based reasoning techniques. Most structure identification techniques use a specific structural syntax. For example, using an un-supervised neural network to identify the blackbox input-output model we obtain a model made up of a network of neurons with interconnections and weights.

Since the neural structure has no physical meaning, the parameters identified by the parameter update algorithm for the neural model as well as the internal state of the neurons have no physical similarity to the plant. While neural networks are often capable of producing good input-output correspondence with the plant, the lack of physical insight about the internal structure of a neural network is a significant limitation.

In the upcoming sections we will make an argument for a semantic modeling scheme to satisfy the need for physical value to the measurements from the model while retaining the ability to automatically adapt the structure. We will show that the semantic form is compatible with all three update mechanisms discussed in this section and in addition will simplify implementation of the structural adaptation process.

The semantic approach for model description draws from the area of knowledge based engineering (KBE) [140]. The sentiment of KBE is to design a computer model with the aim of realizing problem-solving capabilities comparable to a domain expert. In the form we use KBE, semantic modeling implies that we represent the model for an electrical machine as an assembly of subcomponents or building blocks, such as transmission lines, ideal switches, relays and transformers, many of which recur

multiple times in an electrical network. While the models for specific subcomponents may be considered to be proprietary or 'black-box', the component interfaces to the outside world are standard and it is possible to construct a larger macro model by explicitly specifying the relationships between individual subcomponents [64]. An analogy for the design approach is to compare the structure of the macro model to the structure of a sentence in the English language. The semantic constraints we apply to the components within the model are similar to the grammatical constraints placed on the assembly of words into a sentence.

4.4.1 Modelica description

For the analysis presented here we will use the Modelica modeling platform [61, 43] to describe the semantic relationships between subcomponents. Modelica is an object-oriented equation-based modeling language primarily aimed at physical systems. The model behavior is based on ordinary and differential algebraic equations combined with discrete events; it uses acausal modeling and so makes it easier to reuse subcomponents since equations do not specify a signal flow direction. In addition to reuse of components it facilitates automated or guided evolution of models and has multidomain modeling capability. A model definition of a simple DC motor is shown in Figure 4.9 to highlight the salient features of the language. The macro model 'dcmotor' is made up of several subcomponents, 'Resistor', 'Inductor' etc. These individual subcomponents are invoked by the motor model, but no information is known about the internal properties of these subcomponents. The 'equation' section of the definition shows the semantic structure of the model. In the case of the simple motor example the structure is straightforward showing a serial relationship between a voltage source, a resistor, an inductor and so on. The semantic definition also contains additional information about the compatibility of the subcomponents. In the motor, the model includes knowledge that rotational components, such as the inertia

```
model dcmotor
  Modelica.Electrical.Analog.Basic.Resistor r1;
  Modelica.Electrical.Analog.Basic.Inductor i1;
  Modelica.Electrical.Analog.Basic.EMF emf1;
  Modelica.Mechanics.Rotational.InertiaLoad l1;
  Modelica.Mechanics.Rotational.ViscousFriction b1;
  Modelica.Electrical.Analog.Basic.Ground g;
  Modelica.Electrical.Analog.Sources.ConstantVoltage v;
equation
  connect(v.p,r1.p);
  connect(v.n,g.p);
  connect(r1.n,i1.p);
  connect(i1.n,emf1.p);
  connect(emf1.n,g.p);
  connect(emf1.flange_b, load.flange_a);
end dcmotor;
```

Figure 4.9: A model of a PMDC motor described in the Modelica language.

and the viscous friction bearing, can be connected together whereas a voltage source cannot be connected to a load inertia. One can imagine that for larger models the connection between subcomponents can become fairly nebulous; however, there are several techniques specially suited to making sense of large semantic graphs [139].

With the representation shown in Figure 4.9 we have sufficient information to understand the structure of the model and its components. If the component definitions used are global then the components can be reused by several macro models greatly reducing the memory and complexity of a large model based control system such as an electrical grid.

4.4.2 Adaptation using semantic rules

With the semantic architecture just presented we can revisit the problem of structural adaptation discussed in Section 4.4. Two immediate benefits emerge from the new definition.

First, the model structure has a real physical foundation. The subcomponents are selected from a vocabulary of physically meaningful elements and the connections between them can be understood by a user with knowledge about the working of the real physical system.

Second, the description carries with it knowledge about component compatibility and the physical units of the data exchanged between components. This knowledge allows an automated adaptation algorithm to only apply plausible modifications to the model rather than randomly iterating through all possible arrangements of elements. With more and more semantic meta-data added to the model, the search space of plausible models can be greatly reduced to the point where it is possible to deterministically guarantee the convergence of the search algorithm to a unique 'good' model [56].

To understand the application of the semantic adaptation method to the electrical grid consider the following use case: A regional utility company uses models to estimate the state of all the distributed generators connected to a particular substation. One of the models shows consistently bad performance. Attempting to address the estimation errors, the frequency of measurement updates is increased at cost of network bandwidth and an effort is made to reassess the model parameters, however neither of these strategies yield satisfactory results. A field technician is able to use engineering intuition to recognize that the model errors may be due to a structural error in the model.

With the semantic description for the model we have the tools we need to automatically test a structural hypothesis and verify the response. Let us say he proposes a general hypothesis that the generator in question probably has compliant shaft resulting in an un-modeled oscillatory response to step loads. As step 1, we are able to invoke a spring component into the macro model of the generator and using the 'equation' definition we have the ability to introduce the component anywhere in the model structure. In a model with n subcomponents we have $\mathcal{O}((n+1)!)$ possible candidates with the addition of 1 spring. If we apply the similar adaptation policy to the motor model shown in Figure 4.9 we find that there are $\mathcal{O}(10^4)$ candidates evaluated by brute force trial.

Since the semantic description is already equipped with knowledge about the physical significance of each component, as step 2 we can use the semantic knowledge to reduce the search space to only the physically meaningful candidates. The proposed compliant shaft for example can be connected only to other rotary mechanical components with compatible physical units (angular velocity, phase and torque). The application of this simple constraint to the model in Figure 4.9 reduces the number of candidates to $\mathcal{O}(10^2)$.

One other aspect must be considered before empirically testing all the candidate models. For a very large model with multiple outputs not all the subcomponents within the model have significant impact on the output measurement of interest. If the model performance is being evaluated based on an output measurement $\hat{z}$ which is a subset of the set of all outputs $\hat{y}$, it is advantageous for the adaptation algorithm to only address the subcomponents directly related to the output value $\hat{z}$ while minimally impacting unrelated components. We call this step 3 in the candidate selection process, *sensitivity based decomposition*.

4.4.3 Sensitivity based decomposition

The principle behind the sensitivity decomposition process is to evaluate the likelihood of one model candidate over another based on the sensitivity of the model output $\hat{z}$ to the addition of the proposed subcomponent. We propose that it is only justified to evaluate a candidate model when the addition of a new component significantly affects the model's output. By sorting candidates based on sensitivity we are limiting the introduction of 'passive' components in the adaptation process. For example if we used a recursive adaptation algorithm where several hypotheses are iteratively implemented, then sorting the hypotheses in decreasing order of sensitivity automatically applies the hypothesis with most impact first, greatly improving the insight awarded by the process and the speed of model convergence. Sensitivity based

decomposition can be applied to conventional state space models based on the impact that adding the subcomponent has on the dominant eigenvalues of the system [152]. In the case of the semantic model, the model output $\hat{z} = f(\hat{z}_1, ..., \hat{z}_n, x_1, ..., x_p)$ is a function of the outputs of n subcomponents $\hat{z}_1, ..., \hat{z}_n$ and p model inputs $x_1, ..., x_p$. The sensitivity of the output to the subcomponents and the inputs is given by the gradient vector $\nabla f = [\frac{\partial f}{\partial \hat{z}_1} ... \frac{\partial f}{\partial x_p}]$. The adjoint function ∇f is expensive to compute with a large number of terms.

The technique we use to evaluate sensitivity in semantic models is a variation of Algorithmic Differentiation [63]. In particular, we adapt the reverse gradient evaluation method for our purposes. Briefly, the approach is as follows: Consider the schematic representation of the motor model in Figure 4.10. The output of the model is the output of the n^{th} subcomponent. The gradient vector for the n^{th} subcomponent is given by ∇z_n. Exploiting the knowledge that there is only one other semantic connection to component n, we see that $\nabla z_n = \frac{\partial z_n}{\partial z_{n-1}}$. Then, by chain rule we have $\frac{\partial z_n}{\partial z_{n-2}} = \nabla z_n . \nabla z_{n-1}$. Propagating the gradient computation backwards from the output to the inputs x_1 and x_2, we get the adjoint function ∇f. The algorithmic complexity of the process is $\mathcal{O}(n)$, as opposed to executing the algorithm without a-priori semantic knowledge where the complexity is close to $\mathcal{O}(n!)$. It is now feasible to compute ∇f^* for every candidate model from step 2 and order them in decreasing order of the sensitivity metric $\frac{f^*}{\partial z^*}$ where z^* is the output of the proposed component.

Using the 'best few' from the ordered set of candidates a test and verification program can be implemented where the candidates are run alongside the original model to evaluate the model with the best fidelity to the real measurements z. The 'best few' candidates also provide physical insight to support further refinement of the model structure.

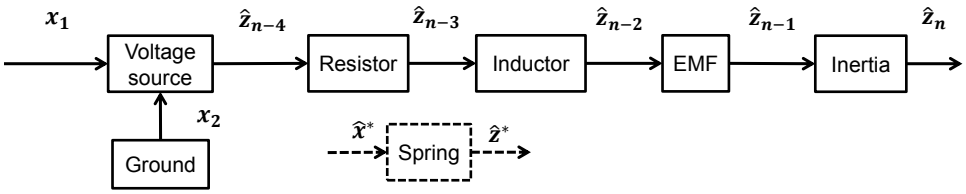

Figure 4.10: A schematic view showing the individual components and connections in the Modelica motor model.

4.4.4 Application

We implemented the presented approach in a simulation using a Matlab model of a Maxon-RE25, 10 Watt, Brushed, PMDC motor. For the sake of simplicity and ease of analysis we consider the Matlab model of the motor (a 3^{rd}-order linear state space model) to be our plant. The motor shaft drives a rotary inertia $J_{load} = 5 * J_{motor}$ so that the total rotary inertia in the system is $Jm = J_{motor} + J_{load}$. A shaft encoder is mounted on the load which provides measurements of the absolute angle θ. The input to the motor is a variable terminal voltage v_{ref}. The output of the state space model is a measurement $z = \theta$.

The semantic model we used to study adaptation was a model for an ideal PMDC motor written in the Modelica language, similar to the model in Figure 4.9. All the subcomponent parameters were set to be identical to the plant parameters. The performance of the semantic model was evaluated based on the error in the estimates of the output angle. The model error is given by $|(z - \hat{z})/z|$. A closed loop state update mechanism was implemented to ensure that the model error was less than 1% for a sine input $v_{ref} = sin(2\pi t) + \eta(t)$ where $\eta(t)$ is a zero mean normally distributed random disturbance $\aleph(0, 0.01 * max(v_{ref}))$.

We introduced a structural change to the plant model by connecting the load rotor J_{load} to the motor through a compliant shaft so the rigid shaft assumption $Jm = J_{motor} + J_{load}$ no longer holds. The addition of the rotary spring adds two additional states to the state space model of the system and results in very different

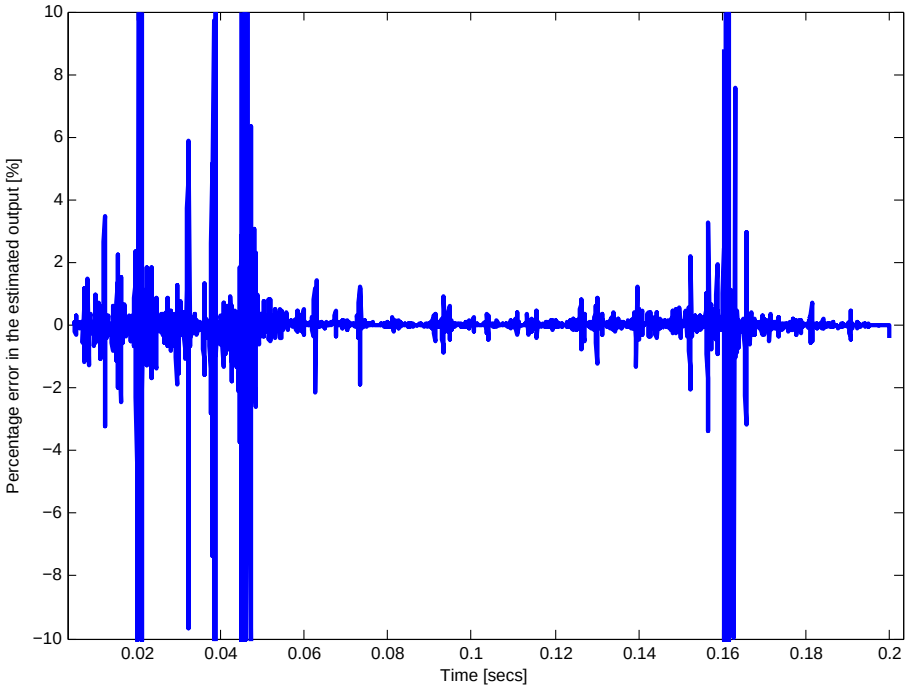

Figure 4.11: Percentage error between the estimated output $\hat{z}$ and the physical output z with a complaint motor shaft.

system behavior. The new state vector for the plant is $[i, \dot{\theta}_{motor}, \theta_{motor}, \dot{\theta}_{load}, \theta_{load}]^T$, as opposed to the state vector $[i, \dot{\theta}, \theta]^T$ for the ideal motor model. The semantic estimator model is retained with an ideal motor structure.

Figure 4.11 shows a trace of the percentage error between the new plant model and the semantic estimator model with a closed loop state update mechanism (discussed in Chapter III) in place. The plot shows that the model errors are frequently above 1% and that the state update mechanism is unable to compensate for the errors introduced due to structural differences between the plant model and the estimator model.

We followed the structural adaptation process to identify the structurally altered plant model. To start, we seeded the system with a hypothesis that any combination of one, two or three components could potentially be added to the model to improve the performance. The components provided were: a load rotor with inertia J_{load}, an ideal gear-box with gear ratio $G = 1 : 2$, and a torsional spring with stiffness K.

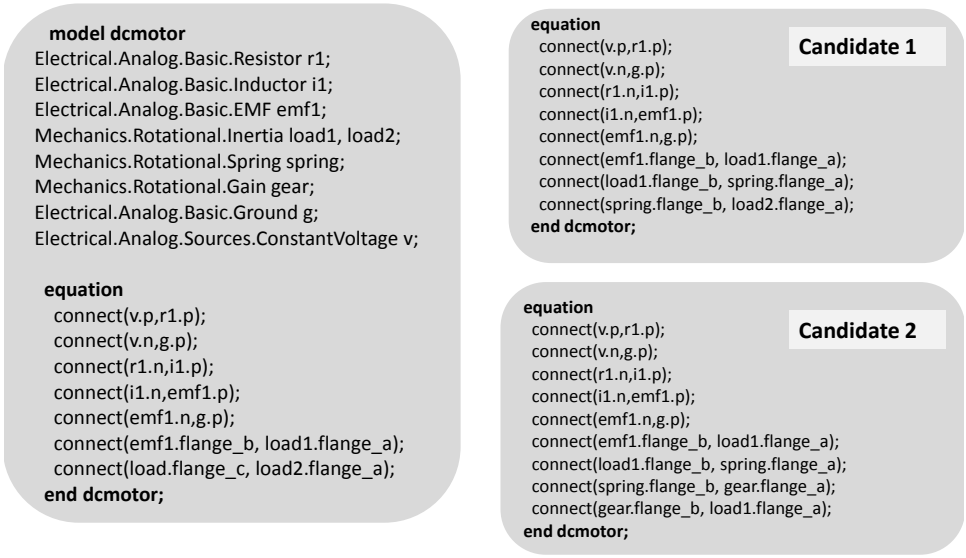

Figure 4.12: The left pane shows the Modelica model of the original system with augmented components. The two right panes show the Modelica equations for the two most likely structural candidates.

The left pane in Figure 4.12 shows the three new components added to the Modelica vocabulary while the *equation* definition of the model is still that of an ideal motor. Evaluating candidates by brute force we find that there are $10 * 9! + 6 * 8! + 3 * 7! = 3.8 \times 10^6$ candidates. Refining the search space with the semantic knowledge about compatibility we get $10 * 4! + 6 * 3! + 3 * 2! = 282$ candidates.

We can now apply the sensitivity analysis as discussed in Section 4.4.3, taking special care to reject models where candidates have passive components with respect to both output velocity and torque. An example of few candidates with passive components are shown in Figure 4.13 with the passive elements shown with dotted lines. The results of the sensitivity analysis are easy to intuitively interpret; Case 1 shows that three rigidly coupled rotors have no dynamics between them and behave as one rotor, Case 2 shows that a spring with no load torque is essentially a mass-less shaft and Case 3 shows that the entire motor model is in effect passive when driving an ideal gear train with no torque feedback.

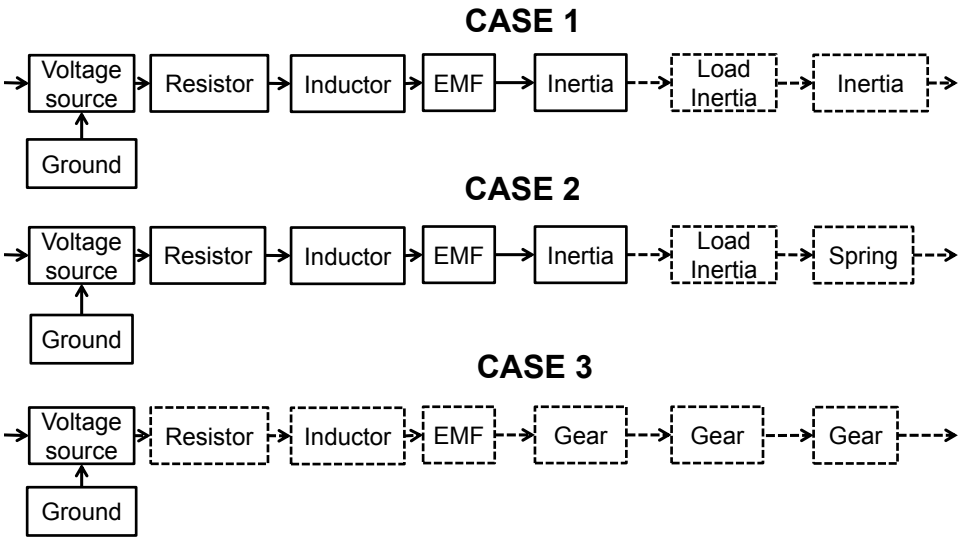

Figure 4.13: Figure showing three cases where a sensitivity analysis reveals the presence of passive components.

4.4.4.1 Results

At the end of the sensitivity analysis we manually selected the 'best few' candidates since we don't yet have a robust automated algorithm to rank candidates based on sensitivity when multiple components are added to the system. We selected 4 candidates shown in Figure 4.14 to test and verify against our plant model. For the verification process each model was run over a 5 second operation sequence in parallel with the plant and the 2 models with the lowest average error were selected as proposed structural updates.

The Modelica models for the two most likely candidates as selected by the verification process are shown in the right panes of Figure 4.12 and for illustration purposes the corresponding schematic diagrams are highlighted in Figure 4.14.

An interesting physical aspect is highlighted by the selection of Candidate 3 where two instances of the inertia element J_{Motor} are selected. We know that $J_{Load} = 5 * J_{Motor}$, therefore with the gear ratio $G = 1 : 2$ the effective load inertia at the output end of the spring is $\frac{J_{Motor}}{G^2} = 4 * J_{Motor} \approx J_{Load}$. Hence the dynamic response of Candidate 3 is very close to Candidate 1. This case is another instance where using

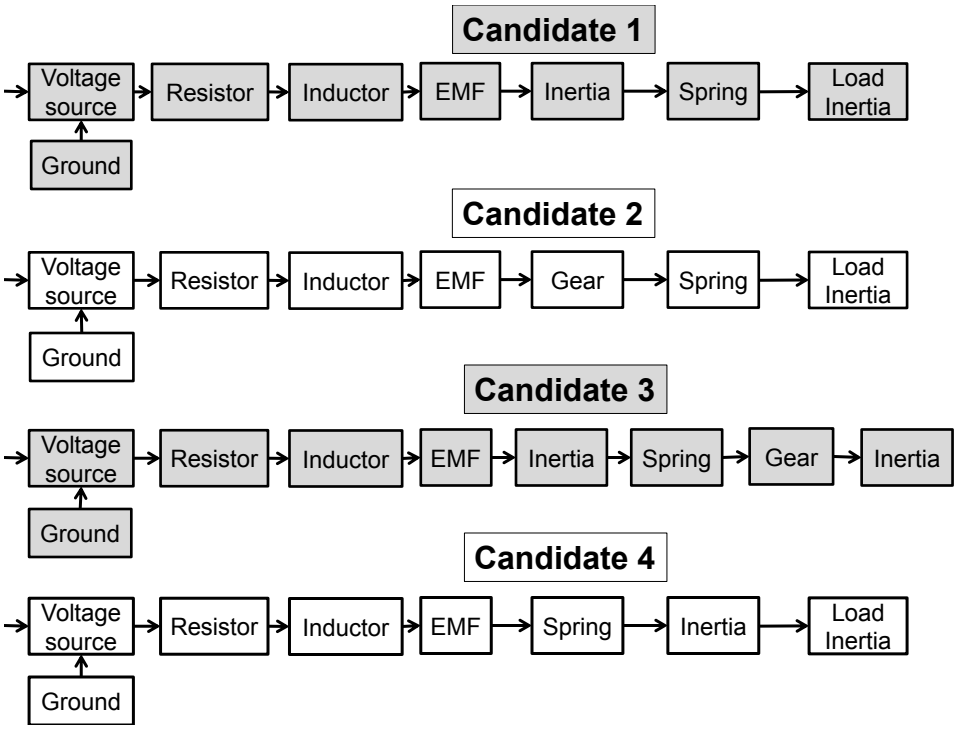

Figure 4.14: Figure showing four candidates selected for performance evaluation. The two most likely candidates after evaluation are highlighted.

the semantic definition aids in analytical insight into model behavior.

4.5 Summary

When models are used in a hybrid process configuration, they may need to be adapted in response to parameters external to the model such as the sampling rate of sensors on the physical process or the delay profile of a communication network. Models for physical systems are rarely designed with the intent that they be automatically updated/adapted in response to changes in external process parameters. It is especially difficult to consider the impact of all external parameters that affect model performance at the time when the model is built.

In Chapter III we used information about relative clock offset, which is an external parameter, to improve the performance of a model based state estimator. The contribution of this chapter is a demonstration that semantic information about a model

can be used to automatically adapt a model's internal structure, when required, to meet performance objectives of a hybrid process.

We used two examples of semantic assertions.

In the first example, the inclusion of a relationship between number of π-sections in a transmission line to the spectral resolution of the model output enabled an automated approach to optimize the model structure based on the timing uncertainty of physical sensors. The system level optimization balanced the fidelity of a model against measurement noise in the physical components. The result confirmed the intuitive impression that there is a practical limit on the spectral resolution of a mathematical model used in a hybrid process configuration since the physical components have finite accuracy and resolution.

In the second example, we demonstrated a method to automatically update the structure of a compositional model made up of several component models. The combinatorial space of all possible compositions is refined by including syntax for the interconnections between component models based on physical feasibility. A semantic assertion relating the sensitivity of the model output to each proposed structural modification is then used as a reward function to rank the set of feasible compositions. The method is applied to an example consisting of a DC motor connected to a rotary inertia load by a flexible shaft.

While semantic information about model order and model composition enabled automated adaptation of model structure, the algorithms presented in this chapter do not scale well with the size of the model. As seen in Section 4.4.3, a declaration of the network of interconnections between model components significantly improved the combinatorial complexity of the adaptation problem. In the upcoming chapters we will further explore the use of declarative descriptions of model topology to improve the computational complexity and convergence properties of adaptation and diagnosis algorithms.

CHAPTER V

Targeted model adaptation for large compositional models with declarative topology descriptions.

The work presented in this chapter has been submitted to the *IEEE Transactions on Automation Science and Engineering*.

5.1 Introduction

The hybrid process shown in Figure 5.1-a, is made up of a model $\mathcal{M}$ that emulates a physical process $\mathcal{P}$. A closed-loop model adaptation algorithm is used to adapt $\mathcal{M}$ when necessary in order to minimize the residual function $g(Y, \hat{Y})$. Y and $\hat{Y}$ correspond to outputs of $\mathcal{P}$ and $\mathcal{M}$ respectively. When the adaptation algorithm works as expected, the errors between model and process can be neglected or accommodated assuming that $\hat{Y}$ tracks Y with sufficient fidelity. When $\mathcal{M}$ represents a large system with hundreds of tunable parameters, the advantages of decoupling the adaptation strategy from functions that use the model, such as control and planning, are significant.

In this chapter we propose a step-by-step method to adapt a system level model made up of component models. Our method strategically draws internal measurements (where available) from the physical system $\mathcal{P}$ to first isolate only those com-

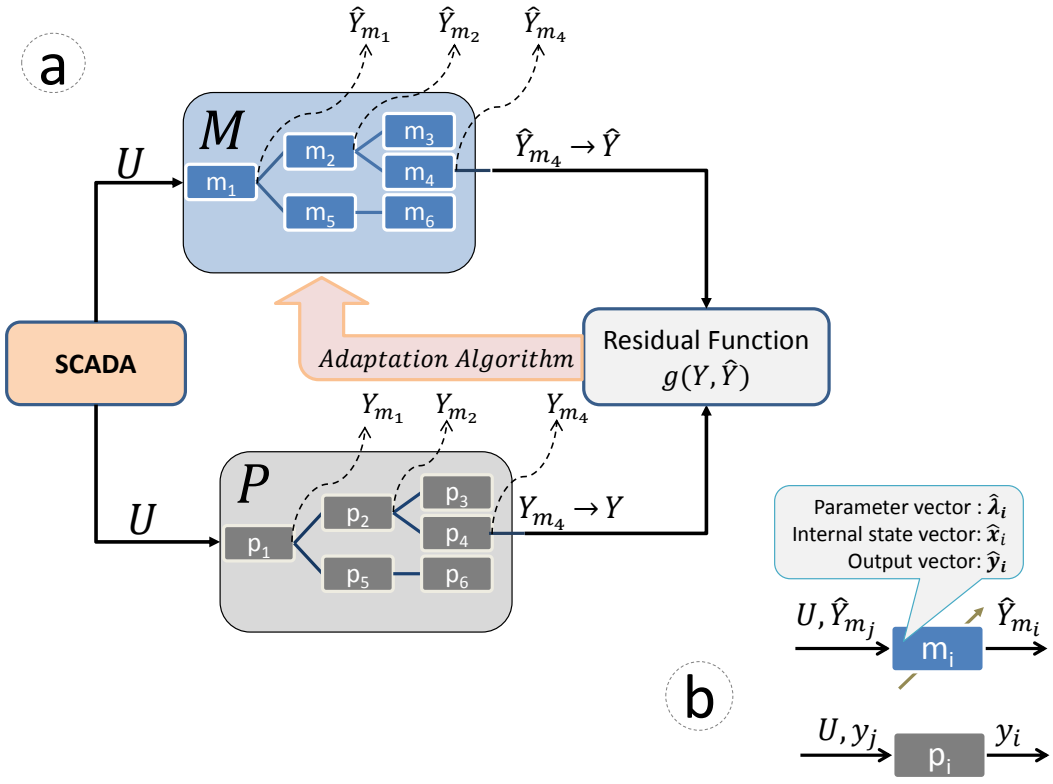

Figure 5.1: A schematic for a general model adaptation mechanism. $\mathcal{M}$ is tuned in response to a function of the error residual $g(Y, \hat{Y})$.

ponents in $\mathcal{M}$ that require adaptation. We show that this model deconstruction step significantly reduces the computational complexity of the model adaptation problem. Our method is suited to large networked systems where communication bandwidth is at a premium and the scale of the system-level model makes it impractical to use conventional methods for system identification. The control infrastructure for the electrical power grid is a good candidate to illustrate the proposed approach and so we apply our results to an electrical circuit shown in Figure 5.2.

In our analysis we make the following assumptions:

1. Large models are comprised of smaller component models. In Figure 5.1-a, $\mathcal{M}$ is comprised of component models m_i assembled compositionally to derive the desired output $\hat{Y}$.

2. The assembly of component models follows the object-oriented paradigm [114] (using a modeling language like Modelica [43]) so that each component model is a discrete reusable model block that can be treated as a model on its own, takes variable inputs, performs a function on them, and returns values $\hat{Y}_{m_i}$.

3. The network of interconnections between component models m_i within $\mathcal{M}$ is called the topology of the model - denoted Γ. In Figure 5.1-a, if Γ denotes the topology of $\mathcal{M}$ then $\mathcal{M} = \Gamma\{m_i\}$ with a set of model parameters $\hat{\lambda}$ and internal state $\hat{x}$.

4. Each component model corresponds to a real physical component p_i in $\mathcal{P}$ as shown in Figure 5.1-b. That is, $\mathcal{P} = \Gamma\{p_i\}(\lambda, x)$.

5. Each component model has a set of tunable parameters denoted $\hat{\lambda}_i$, internal state $\hat{x}_i$ and a set of outputs $\hat{Y}_{m_i}$. The output of the model is a function of one or more component outputs. In Figure 5.1-a, the model output $\hat{Y} = \hat{Y}_{m_4}$.

In the following section we will more formally present the general problem, and

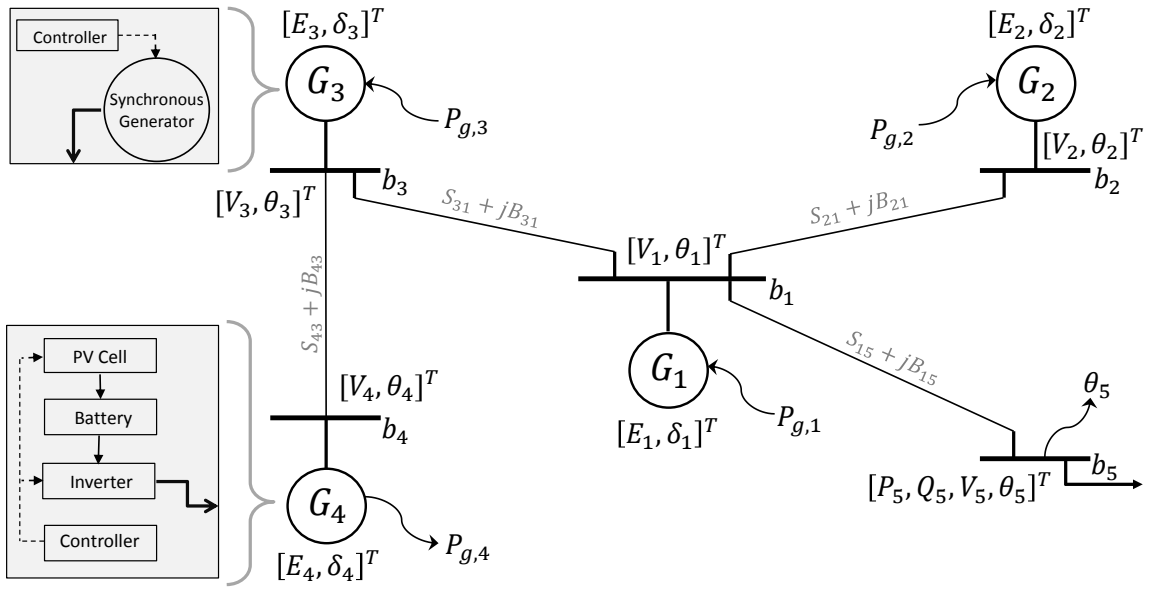

Figure 5.2: A schematic diagram of the four generator microgrid.

outline the specific adaptation problem that we address. In Section 5.3 we will present a motivating example in the domain of electrical power distribution and outline the challenge in adapting models of large electrical networks. In Section 5.4 we will present our method for decomposing the model in order to simplify the adaptation challenge and then apply our method to the motivating example in Section 5.5. In Section 5.6 we will summarize our contributions and discuss some future improvements to extend the scope of our method.

5.2 Problem Statement

Referring to Figure 5.1, assume that a physical system $\mathcal{P}$ has an object-oriented model $\mathcal{M}$ that is run in parallel with the operating physical system; the same inputs U are applied to both the system $\mathcal{P}$ and the model $\mathcal{M}$. For each component output Y_{m_i} there is a corresponding output Y_{p_i}. That is, every model component $m_i \in \mathcal{M}$ has an analogous component $p_i \in \mathcal{P}$. The internal state vector $\hat{x}$ and the parameter set $\hat{\lambda}$ of $\mathcal{M}$ is a union of the states and parameters of the individual component models.

The set of states and parameters in $\mathcal{P}$ are denoted x and λ respectively.

Reflecting the practical challenges in measuring $Y_{p_i}(\forall p_i \in \mathcal{P})$ we assume that network bandwidth limitations restrict the number of real-time measurements that can be drawn from the physical system [72] and apply a cost to each measurement drawn. In essence, sufficient number of measurements are not available from the physical system at all times to fully reconstruct the state x.

A discrepancy between the outputs of $\mathcal{M}$ and $\mathcal{P}$ or $\|Y - \hat{Y}\| > \sigma$ may signify some sort of incorrect model parameter, unmodeled disturbance or fault in the system. We do not address faults here. We focus on updating the model parameters to address the discrepancy, using the following step wise approach:

1. Express the dynamics of the model using a structure preserving state space realization.

2. Identify sets of model components within the system model $\mathcal{M}$ that must be adapted to resolve the discrepancy. Each set of model components that could be adapted to resolve the discrepancy is called an *adaptation candidate* $\mathrm{c}_i \in \mathcal{C}$.

3. Draw measurements Y_{p_i} from the components p_i of the physical system $\mathcal{P}$ to differentiate between the candidates until either no more measurements are available or the smallest set of candidates $\mathcal{C}_{min} \subseteq \mathcal{C}$ has been isolated.

4. Isolate and extract the state space representation corresponding to each model component $\mathrm{m}_i \in \mathcal{C}_{min}$.

5. Update the internal parameters of each partitioned component model using appropriate parameter identification methods to resolve the discrepancy $\|Y - \hat{Y}\| > \sigma$ and restore good agreement between $\mathcal{M}$ and $\mathcal{P}$.

This chapter provides a model decomposition strategy for large systems where the direct application of parameter identification algorithms is not practical. Con-

ventional decomposition methods use Spectral or Eigen norms to sort and isolate key system states whereas our approach exploits knowledge of the physical structure to extract critical subcomponents.

Our method offers three advantages over the existing methods:

1. Physical interpretation of the system states is retained: No basis transformations are applied during the decomposition. As a result, the isolated components are physically atomic and the state of the decomposed system is a subset of the original state vector.

2. Prior knowledge of observability constraints is not required: The algorithm draws measurements one at a time and does not require guarantees on state observability. For systems with very limited measurements, prior probabilities, expert intuition or historical evidence may be used to refine the candidate space.

3. The method is compatible with most graph search and parameter identification algorithms: Our approach uses computationally efficient graph exploration tools to decompose the system and conventional parameter estimation methods for the decomposed system.

5.3 Motivating example: Electrical Power Network

Recently significant efforts have been devoted to modeling the dynamic behavior of power networks, especially in the context of distributed generation and advanced metering [57]. Our motivating electrical system is a simplified version of the Consortium for Electric Reliability Technology Solutions (CERTS) Microgrid Testbed Demonstration [89] as shown in Figure 5.2. This illustrative problem was selected since it also addresses many of the challenges listed within the *Smart Grid Interoperability Standards Roadmap* published by the National Institute of Standards and Technology (NIST) [44]. The specific challenge from the roadmap document that

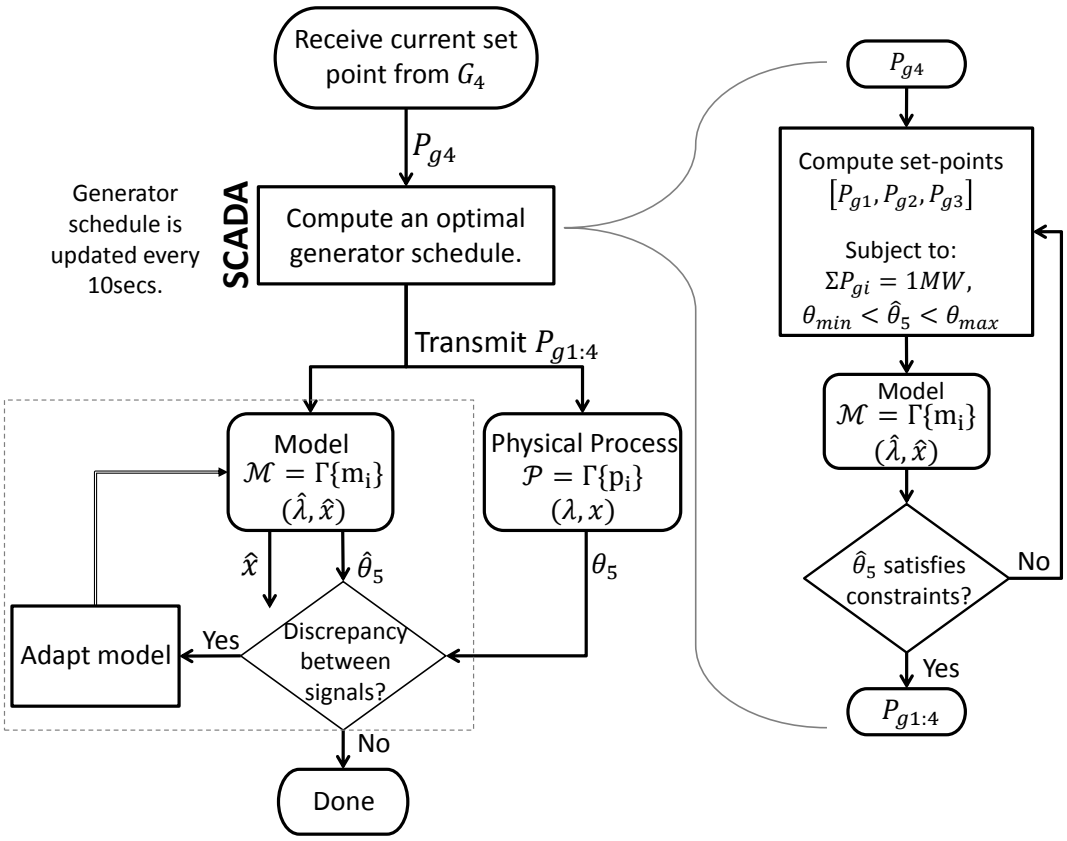

Figure 5.3: The diagram shows schematic view of the signal flow in the motivating example. Our focus in this chapter is the adaptation of $\mathcal{M}$ involving components shown within the rectangle.

we address is the lack of tools to provide a regional power distribution service with the ability to reliably predict the behavior of a network of distributed 'microgrids' [125]. The circuit shown in Figure 5.2 is a simple microgrid where the objective is to integrate a set of four distributed micro-generators labeled G_1 to G_4 into the electric grid. The micro-generators each have local controllers, but are dispatched by a remote SCADA (*Supervisory Control and Data Acquisition*) system. Supervisory control is also used to regulate the interface between the microgrid and the larger utility managed power system at Bus-5.

The schematic diagram in Figure 5.3 shows the application of a SCADA system used to optimally dispatch a set of distributed micro generators [57]. As in most cases, the remote SCADA system uses a model of the microgrid circuit to optimize the generator schedule [18]. In our example, the power output of the inverter G_4 is uncontrolled by the SCADA system and varies in time with changes in the available battery power and solar flux. In response to a change in the set point Pg_4 for G_4, the SCADA system optimizes the schedule for the synchronous generators G_1, G_2 and G_3 to retain $1MW$ cumulative supply from the microgrid. A model of the circuit is used to estimate the voltage phase angle θ_5 at Bus-5 for every schedule.

5.3.1 Modeling the power network

Individual micro-generators may differ from others in the microgrid not only in their system parameters but also in their structure and design. In our example G_4 is a battery based DC power source charged using a Photovoltaic cell, coupled to the AC circuit through an inverter, and is structurally dissimilar to the synchronous rotary machines used for G_1, G_2 and G_3. In order to scale the model-based control approach to a wide area network with potentially hundreds of micro-generators, each with a particular design and certain response characteristics, it is necessary to abstract the model of each generator in the circuit into a simplified generic model. We will use the

classical linearized version of the swing model [57] to model the microgrid in Figure 5.2.

To state the problem more generally, consider a power network with n generators and $m > n$ buses indexed by $[G_1, \ldots, G_n]$ and $[b_1, \ldots, b_m]$, respectively. Let $[b_1, \ldots, b_n]$ be the generator buses, each one connected to exactly one generator, and let $[b_{n+1}, \ldots, b_m]$ be the load buses. As usual in transient studies, the generator dynamics are given by the transient constant-voltage behind reactance model [110]. With the i^{th} machine, we associate the voltage modulus E_i, the rotor angle δ_i, the inertia J_i, the damping coefficient D_i, the transient reactance z_i, and a reference power signal $P_{g,i}$. With the i^{th} bus we associate the voltage modulus V_i, the phase angle θ_i, the active and the reactive power demands P_i and Q_i, respectively.

With this notation the simplified dynamics of the i-th generator $i \in [1, \ldots, n]$ are given by Equation 5.1:

$$\begin{aligned} \dot{\delta}_i(t) &= \omega_i(t), \\ J_i\dot{\omega}_i &= P_{g,i}(t) - \frac{E_i V_i}{z_i}\sin(\delta_i(t) - \theta_i(t)) - D_i\omega_i(t) \end{aligned} \tag{5.1}$$

We denote with S_{jk} and B_{jk} the conductance and susceptance of the branch lines between buses b_j and b_k. Then the net active (P) and reactive (Q) power flow out of each bus b_i is given by the non-linear power flow equation:

$$\begin{aligned} P_i &= \sum_{j=1, j\neq i}^{m} V_i V_j B_{ij}\sin(\theta_i - \theta_j) + \sum_{j=1, j\neq i}^{m} V_i V_j S_{ij}\cos(\theta_i - \theta_j) \\ Q_i &= \sum_{j=1, j\neq i}^{m} V_i V_j S_{ij}\sin(\theta_i - \theta_j) - \sum_{j=1, j\neq i}^{m} V_i V_j B_{ij}\cos(\theta_i - \theta_j) \end{aligned} \tag{5.2}$$

A linear small signal model can be derived from the non-linear model under assumptions that angular differences in phase and magnitude differences in line voltage across

the network are small. In brief, the assumptions are: $\forall i, j \in [1 \dots m]$ & $\forall k \in [1 \dots n]$, $|\theta_i - \theta_j| \ll 1$, $|\delta_k - \theta_j| \ll 1$, $S_{ij} = 0$, and $E_k = V_i = V_j = 1$. The linearized equations for power about a synchronized network in steady state yields the dynamic linearized swing equation and the algebraic DC power flow equation. These equations can be assembled into a state-space model for the network, producing a small signal version of the structure-preserving power network model derived in [118].

$$\begin{bmatrix} I & 0 & 0 \\ 0 & J & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\delta}(t) \\ \dot{\omega}(t) \\ \dot{\theta}(t) \end{bmatrix} = - \begin{bmatrix} 0 & -I & 0 \\ L_{gg} & D & L_{gl} \\ L_{lg} & 0 & L_{ll} \end{bmatrix} \begin{bmatrix} \delta(t) \\ \omega(t) \\ \theta(t) \end{bmatrix} + \begin{bmatrix} 0_{(1:n)} \\ Pg_{(1:n)} \\ P_{(1:m)} \end{bmatrix} \tag{5.3}$$

The matrix $L = \begin{bmatrix} L_{gg} & L_{gl} \\ L_{lg} & L_{ll} \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$ is the Laplacian matrix [21] of a susceptance weighted graph of interconnections between busses and generators. Following the notation in [118], L_{gg} is diagonal, L_{ll} is invertible, and $L_{lg} = L_{gl}^T$. Equation 5.3 is used to predict the change in voltage phase angle at each of the buses $\theta_{1\dots5}$ and the response of each of the generators in the circuit in response to changes in the generator schedule $[Pg_1, Pg_2, Pg_3, Pg_4]$. As stated earlier, this model is critical to the performance of any scheduling or control algorithm that might be employed by the SCADA system to dispatch the three controllable generators. Returning to our motivating example, given a generator schedule $[Pg_1, Pg_2, Pg_3, Pg_4]$, the model $\mathcal{M}$ is used to estimate the expected dynamic response of the generators in the circuit.

We will assume that the generators are of a PV-type [57], and therefore we can decompose the set of differential algebraic equations in Equation 5.3 into two systems, $\mathcal{M}_g$ and $\mathcal{M}_f$, by ignoring the interaction between the generator states and bus states represented by sub-matrix L_{lg}. The resulting systems $\mathcal{M}_g$ and $\mathcal{M}_f$ are solved sequentially to constitute the output of $\mathcal{M}$. $\mathcal{M}_g$ is a sub-model representing

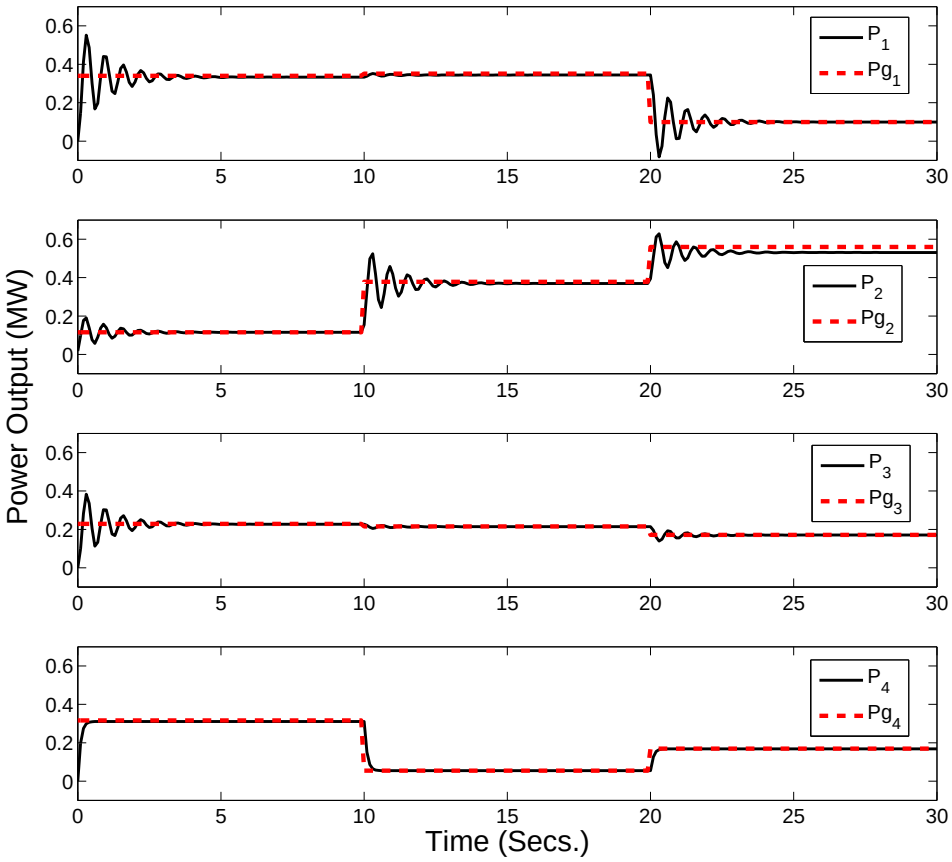

Figure 5.4: The figure shows the power output from four micro-generators (G_1, G_2, G_3, G_4) overlayed on the input schedule marked with a dotted line.

the generator dynamics using a set of ordinary differential equations in the explicit LTI form ($\dot{x} = Ax + Bu$). $\mathcal{M}_f$ is a sub-model representing the algebraic constraints on the relative phase angle between adjacent buses in the circuit in the form $\alpha = x\beta$. By decoupling the systems in this way, we simplify the simulation and parameter identification process.

The response of model, $\mathcal{M}_g$ for all four generators is shown in Figure 5.4 as the schedule is changed once every 10 seconds. The model $\mathcal{M}_f$ is used to estimate the phase angle at all the buses in the circuit. The solid line in Figure 5.5 shows the model estimated phase angle $\hat{\theta}_5$ at the utility interface bus, Bus-5.

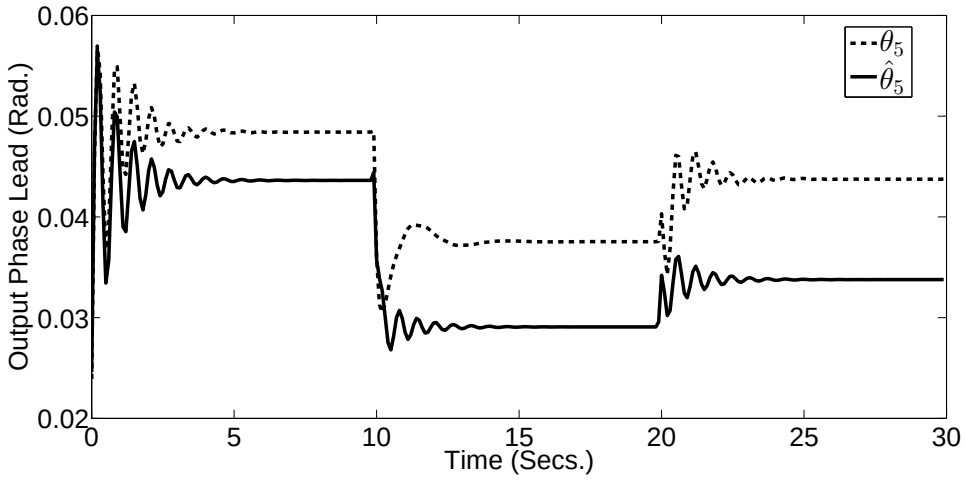

Figure 5.5: The figure shows a plot comparing the model estimate $\hat{\theta}_5$ against the measurement θ_5. A difference in the parameter values for G_2 between $\mathcal{M}$ and $\mathcal{P}$ is manifested as a large difference between $\hat{\theta}_5$ and θ_5.

5.3.2 Discrepancies between the model and the physical system

Referring to the flow chart in Figure 5.3, we see that the physical circuit $\mathcal{P}$ returns a measurement of the phase angle at every time step ($Y = \theta_5$). Also the output of $\mathcal{M}$ is $\hat{Y} = \hat{\theta}_5$. Using the available measurement from $\mathcal{P}$, we can compare Y and $\hat{Y}$ to identify when the difference between the two signals is beyond an acceptable threshold. If the threshold is violated we say there is discrepancy between $\mathcal{P}$ and $\mathcal{M}$.

Our analysis in this chapter addresses the problem of model adaptation when a discrepancy occurs between $\mathcal{M}$ and $\mathcal{P}$. We use a full AC non-linear model of the circuit, constructed from the power flow equations [5.2], as a proxy for $\mathcal{P}$. For $\mathcal{M}$ we continue to use the linear model based on Equation 5.3. The outputs from $\mathcal{P}$ comprise the 'ground truth' for the adaptation algorithm and the outputs of $\mathcal{M}$ are evaluated against the ground truth to trigger model adaptation.

Consider the model $\mathcal{M}$ with internal state $\hat{x}$ and a set of model parameters $\hat{\lambda}$. In our example the system parameters that comprise the set $\hat{\lambda}$ are the coefficients J_i, D_i in Equation 5.1, the values for coupling reactance z_i at each generator and the susceptance values of each branch line B_{jk}. The state $\hat{x}$ includes the phase angles at

each of the five buses $\hat{\theta}_{1...5}$ and at the four generator terminals $\hat{\delta}_{1...4}$.

Referring to Figure 5.1, when a discrepancy is detected between $\hat{Y}$ and Y, $\mathcal{M}$ may be adapted by correcting its internal state or by tuning its parameters. A common approach used to adapt the model state $\hat{x}$ is the "Dynamical State Observer" [100]. A state observer assumes a perfect model of the plant dynamics and knowledge of the internal parameter set λ. An error correcting function $L(Y - \hat{Y})$ is used to generate an estimate $\hat{x}$ of the true state of the system x. For a linear observer, the error correcting term becomes $L(Y - \hat{Y})$ where the matrix L is called the observer gain. Several algorithms exist to find the optimal L under constraints such as the presence of measurement noise [71].

For models with incorrect system parameters, a state observer (that assumes perfect modeling) may fail to converge to the true state. Parameter estimation may then be necessary to tune model parameters $\hat{\lambda}$ in response to error between Y and $\hat{Y}$. We focus on the problem of parameter estimation in this chapter.

In Figure 5.5, θ_5 and $\hat{\theta}_5$ are significantly different. The difference in the output values is caused due to incorrect model parameters ($\hat{\lambda} \neq \lambda$). This is a fairly common failure mode in models that are assembled compositionally. It results, most often, from the use of generic or nominal values for parameters that are difficult to physically obtain. In our example we assume that the system parameters for G_1, G_2 and G_3 are all identical when constructing $\mathcal{M}$. The discrepancy seen in Figure 5.5 is present because the actual parameter values for generator G_2 are different in $\mathcal{P}$ than in $\mathcal{M}$.

5.3.3 Parameter identification in response to model discrepancy - standard approach

In response to the discrepancy shown in Figure 5.5, we will first apply standard techniques to adapt the parameter set $\hat{\lambda}$ in $\mathcal{M}$. We will show that computational complexity of the parameter identification algorithm increases super-linearly with the

number of parameters in $\mathcal{M}$. This analysis provides justification for the need to first decompose $\mathcal{M}$ and target only those components in need of adaptation. Besides the computational resources required, we also discuss other penalties incurred by directly applying standard parameter identification techniques to the full model.

Equation 5.4 shows a state space representation for the sub-model $\mathcal{M}_g$ with input $U_{\mathcal{M}_g} := [P_{g1}, \cdots, P_{gn}]^T$, output $\hat{Y}_{\mathcal{M}_g} := [\hat{P}_1, \cdots, \hat{P}_n]^T$ and internal state $\hat{x}_{\mathcal{M}_g} := [\hat{\delta}(t), \hat{\omega}(t)]^T$.

$$\begin{aligned} \dot{\hat{x}}_{\mathcal{M}_g} &= \begin{bmatrix} I & 0 \\ 0 & J \end{bmatrix}^{-1} \begin{bmatrix} 0 & I \\ -L_{gg} & -D \end{bmatrix} [\hat{x}_{\mathcal{M}_g}] + \begin{bmatrix} I & 0 \\ 0 & J \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ I \end{bmatrix} U_{\mathcal{M}_g} \\ \hat{Y}_{\mathcal{M}_g} &= [L_{gg}, 0_{n\times n}][\hat{x}_{\mathcal{M}_g}] \end{aligned} \tag{5.4}$$

Equation 5.4 for an n generator, m bus circuit can be represented by a generic linear model of the form:

$$\begin{aligned} \begin{bmatrix} \dot{\hat{x}}_{\mathcal{M}_g} \\ \hat{Y}_{\mathcal{M}_g} \end{bmatrix} &= \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \hat{x}_{\mathcal{M}_g} \\ U_{\mathcal{M}_g} \end{bmatrix} \\ \hat{x}_{\mathcal{M}_g} &\in \mathbb{R}^{2n} \\ \hat{Y}_{\mathcal{M}_g}, U_{\mathcal{M}_g} &\in \mathbb{R}^{n} \end{aligned} \tag{5.5}$$

Subspace Identification [148], [81] could be used to estimate the parameter matrix $\hat{\Lambda} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ using sampled input-output data drawn from $\mathcal{P}$.

The subspace identification algorithm is as follows:

1. Obtain N sampled measurements of the inputs $U_{\mathcal{P}_g}$ and outputs $Y_{\mathcal{P}_g}$.
2. Construct a block Hankel matrix $W = \begin{pmatrix} U_{\mathcal{P}_g} \\ Y_{\mathcal{P}_g} \end{pmatrix}$ of the past measurements of the inputs and outputs.

3. Obtain N^+ additional measurements of the inputs and outputs. Find the oblique projection of the row space of the output matrix $Y^+_{\mathcal{P}_g}$ along the row space of the input matrix $U^+_{\mathcal{P}_g}$ on to the row space of the Hankel matrix W. $(O = Y^+_{\mathcal{P}_g}/_{U^+_{\mathcal{P}_g}} W)$.

4. Factorize the oblique projection matrix to get $O = \bar{U}\bar{S}\bar{V}^*$. Subspace identification theory [81] proves that the extended observability matrix $O_d = \bar{U}\bar{S}^{1/2}$.

5. Specify a desired model order and use the extended observability matrix to estimate a discrete state trajectory $\hat{x}_{\mathcal{M}_g}(k)|_{k\in[1,N]}$.

6. Identify the parameter matrix $\hat{\Lambda}$ while constraining the solution to the structure shown in Equation 5.4 (we use the SSEST algorithm to solve the constrained estimation problem).

Once the parameters $\hat{\Lambda}$ of $\mathcal{M}_g$ have been identified, the algebraic constraints in sub-model $\mathcal{M}_f$ are formulated into a linear system of equations as shown in Equation 5.6. Using a least squares approach we can identify the parameter matrix L_{ll} using sampled measurements of the phase angles $\theta_{(1:m)}(k)$ and the net power-flows at each bus $P_{(1:m)}(k)$. Since the parameter matrix L_{ll} is a sub-block of the Laplacian matrix of interconnections, buses with no branch lines connecting them correspond to zero elements in L_{ll}. Since we assume that the structure of interconnections in the model is known, only the non-zero elements in L_{ll} need to be identified. A constrained least squares formulation is used to enforce the desired properties for the matrix L_{ll}.

$$0_{(m\times 1)} = -L_{ll}.\theta(k) + P_{(1:m)}(k) \tag{5.6}$$

We use the Matlab® implementation of the $SSEST$ and $LSQLIN$ algorithms to solve Equations 5.4 and 5.6 respectively. Figure 5.6 shows the time taken to execute steps 3 through 6 of the identification algorithm for an increasing number of generators in $\mathcal{M}_g$.

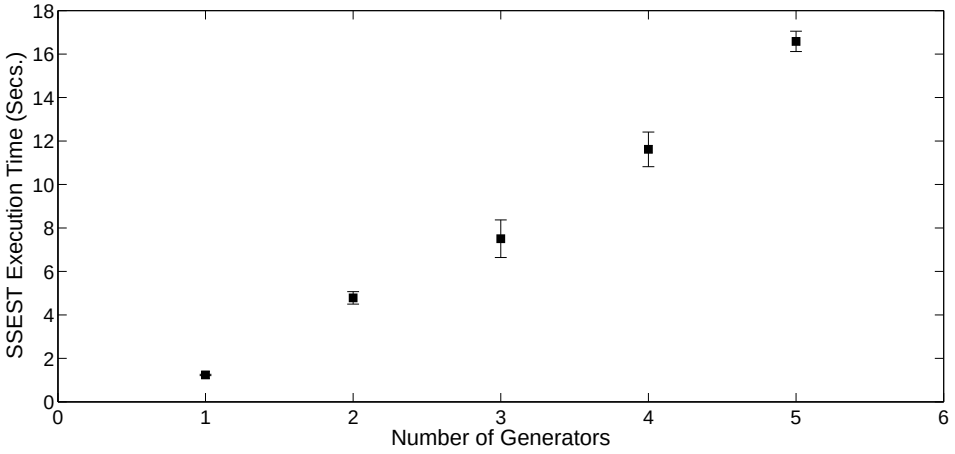

Figure 5.6: The plot shows the execution time for the SSEST algorithm with increasing number of generators. The execution time reflects the super-linear relationship between computational complexity and model size (and corresponding size of $\hat{\lambda}$). The error-bars show the spread of execution times for ten repeated executions with randomly generated initial estimates for $\hat{\lambda}$.

Clearly, the computational resources required to identify $\hat{\lambda}$ do not scale proportionally with the size of the model. We observe that there are three primary implementation challenges in directly identifying $\hat{\lambda}$ for the full model $\mathcal{M}$; they are discussed in brief below and our proposed solution is presented in the following section.

5.3.3.1 Computational complexity

The parameter identification algorithm scales poorly with the number of parameters to be identified. The computational complexity of the Matlab® SSEST+PEM algorithm shows an $\mathcal{O}(n^4 + N^2)$ relationship with system state [106]. Similarly, the constrained least squares solver LSQLIN is $\mathcal{O}(m^3)$. The computational complexity of the parameter identification problem for typical electrical microgrids with several dozen buses makes the standard approach impractical. A test scale microgrid featured in [52], for example, features 34 busses with 3 generators [52].

5.3.3.2 Observability of the global system state

Parameter identification by the subspace method fails in Step-3 when the Hankel Matrix is singular. Therefore, for a unique realization of the parameter matrix $\hat{\Lambda}$ in an n generator model $\mathcal{M}_g$ we need the rank condition $rank(Y^+_{\mathcal{P}_g}/_{U^+_{\mathcal{P}_g}} W) = 2n$ to hold.

Similarly, for a unique least squares solution for the parameters of an m bus model $\mathcal{M}_f$, we need the regressor matrix $(x^T x)$ to be invertible. That is, for a unique solution for parameter matrix L_{ll} in Equation 5.6 we need access to the phase angle θ_i and the net power flow P_i $\forall i \leq m$, at each time step ($k \in [1, N]$).

5.3.3.3 Real-time reporting of measurements

Control networks for power systems use switched Ethernet communication for wide area communication [90]. Switched networks are rarely able to ensure deterministic reporting rates for sampled data since current network protocols utilize a best effort transmission mode where data packets are delayed during transmission to regulate throughput. The transmission delays are typically non-deterministic especially under heavy traffic conditions resulting in reduced data quality at the receiving end [3]. While modern electrical power systems are well instrumented and measurements may be available from almost every bus on the network, the reduced data quality adversely affects the accuracy of parameter estimation methods [7].

This section presented a motivating example of a 5-bus electric power grid and highlighted three challenges in applying existing parameter identification methods namely computational complexity, unobservable states and lack of real-time measurements. Section 5.4 will propose an approach that addresses these challenges. Then, Section 5.5 will demonstrate our proposed approach on the same motivating example.

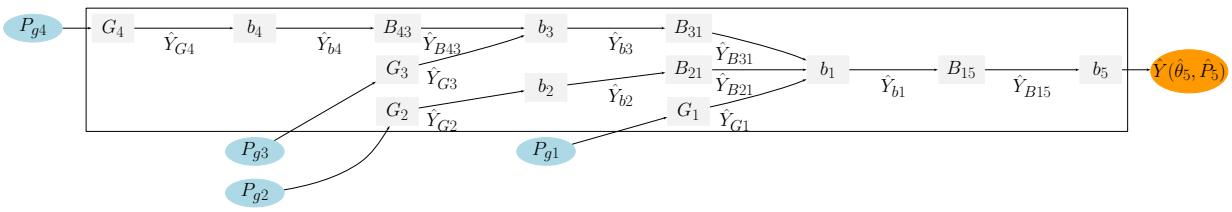

Figure 5.7: A compositional model of the 5-bus circuit from Figure 5.2. The model $\mathcal{M}$ is made up of several model components m_i shown as grey rectangular boxes. Each model component has an output $\hat{Y}_{m_i}$.

5.4 Guided Decomposition for Model Adaptation

This section presents the main contribution of this chapter: A systematic method to decompose a model into smaller and smaller components until only those components that are in need of adaptation are presented to the parameter identification process. This approach also takes into account network communication constraints and incomplete state observability.

The following assumptions are made about the system:

1. The model $\mathcal{M}$ is constructed compositionally using model components m_i and an explicit description of the interconnections between model components expressed using the syntax $\Gamma\{\cdot\}$. By ignoring the interaction in sub-matrix L_{lg}, as in Section 5.2, $\Gamma\{m_i\}$ can be represented as a directed acyclic graph as shown in Figure 5.7. The figure also shows the outputs from each model component $\hat{Y}_{m_i}$.

2. The syntax $\Gamma\{\cdot\}$ is also used to compose $\mathcal{P}$. The output Y_{p_i} of every physical component $p_i \in \mathcal{P}$ can be measured, when required, at a cost $\mathcal{V}Y_{p_i}$.

3. For every measurement channel Y_{p_i} drawn from $\mathcal{P}$ and $\hat{Y}_{m_i}$drawn from $\mathcal{M}$, a discrepancy classifier $g(\cdot,\cdot)$ is provided so that $g(Y_{p_i}, \hat{Y}_{m_i})$ returns a value for the binary decision variable $\mathcal{D}\hat{Y}_{m_i}$ indicating the presence of a significant discrepancy ($\mathcal{D}\hat{Y}_{m_i} = 1$) or not.

4. The classifier $g(Y, \hat{Y})$ is not affected by intermittent mismatches between $\mathcal{P}$ and $\mathcal{M}$. That is, the classification $\mathcal{D}\hat{Y}$ for any given model output $\hat{Y}$ is valid under all expected inputs U and remains unchanged until $\mathcal{M}$ is adapted to resolve the discrepancy.

5. Every model component can be classified into one of a finite set of known model classes or genotypes (e.g., $model\ class \in \{generator, bus, branch\}$). The model class must be known in order to construct the state space model for each component using the structure shown in Equation 5.3.

6. Each component model m_i in $\mathcal{M}$ is amenable to the parameter estimation techniques used in Section 5.3.3. That is, when a Hankel matrix is assembled from the outputs Y_{p_i} to identify the parameter matrix $\hat{\Lambda}_{m_i}$ for model m_i, it satisfies the necessary rank condition.

The flow chart in Figure 5.8 shows a schematic overview of the proposed method and each step in the process is discussed in detail below:

5.4.1 Step 1: Detect discrepancies

First, our algorithm marks the comparison between Y and $\hat{Y}$ and all available internal points of comparison between Y_{p_i} and $\hat{Y}_{m_i}$ as either good or discrepant by setting $\mathcal{D}\hat{Y}$(or $\mathcal{D}\hat{Y}_{m_i}$) to 0 or 1 respectively. From the perspective of practicality, it is necessary to flag a discrepancy only when the difference between Y_{p_i} and $\hat{Y}_{m_i}$ is significant.

In general terms, a discrepancy can be classified as significant when the magnitude of a discrepancy classifier function $g(Y_{p_i}, \hat{Y}_{m_i})$ is greater than a specified threshold σ_{class}. The choice of the function $g(\cdot, \cdot)$ varies based on the type of data being classified. Common statistical metrics include the magnitude of the Squared Bias (SB), the magnitude of the Root Mean Square Error (RMSE) or the magnitude of the Standard

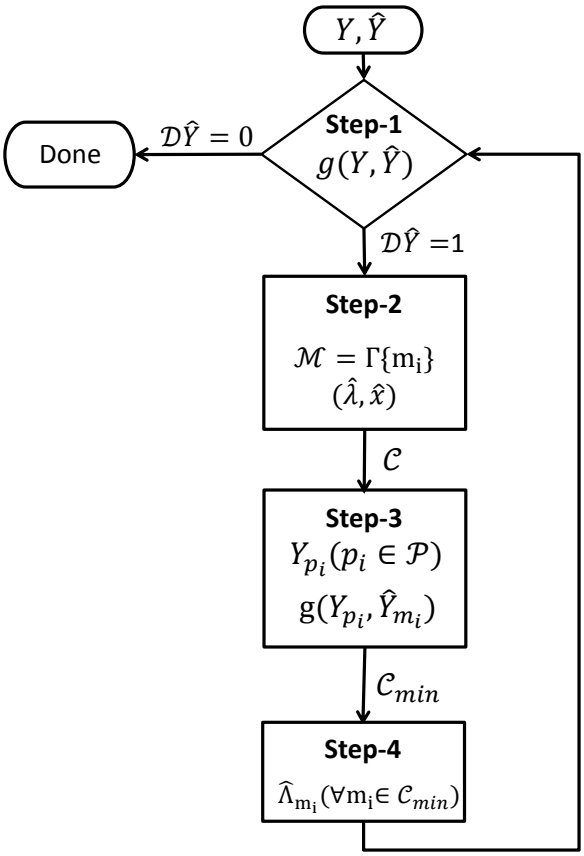

Figure 5.8: The figure shows a flowchart of the proposed guided decomposition method. The steps in the diagram are explained in Sections 5.4.1 through 5.4.4.

Deviation weighted Correlation between signals [11].

The choice of function $g(\cdot,\cdot)$ is also heavily influenced by the function of the model. For a model used to schedule generators at the SCADA level, the classifier function may focus on the steady state equilibria of the system for a given input. In such a case it might be prudent to use a bias function to detect deviations between the expected equilibrium and the true equilibrium. Active damping or transient suppression functions use models to identify the spectral properties of the power flow dynamics in an electrical network, highlighting the oscillatory modes of the system in the frequency domain, in such a case $g(\cdot,\cdot)$ may be chosen to classify discrepancies in the frequency domain. A survey of control-theoretic and other signal processing techniques commonly used to classify discrepancies can be found in [120].

In our application, we will restrict ourselves to the time domain and focus on the error between Y and $\hat{Y}$ at steady state. That is, for each step change in the generator set-point P_{gi}, we ignore the initial transients in the response and classify the error between the outputs once both systems have reached their steady state values for the current set-point. The classifier compares sampled measurements from the physical circuit $Y(k)$ and the corresponding output from the model $\hat{Y}(k)$. As the model is decomposed into smaller components further along the process, the same classifier is used to compare internal measurements $Y_{p_i}(k)$ to their corresponding model generated estimates $\hat{Y}_{m_i}(k)$.

In order to achieve the desired classifier characteristics, a first-order filter is used to filter the fractional error signal $\frac{(Y(k)-\hat{Y}(k))}{\hat{Y}(k)}$. Referring to the dynamic response of the generators in Figure 5.4, we see that the set-point is changed once every 10 seconds, by designing a filter with a time constant of 5 seconds we find that most of the transients in the system response are filtered out. Once the error signal is filtered, a magnitude threshold $\sigma_{class} = 10\%$ is used to classify the output $\hat{Y}(k)$ as discrepant or not.

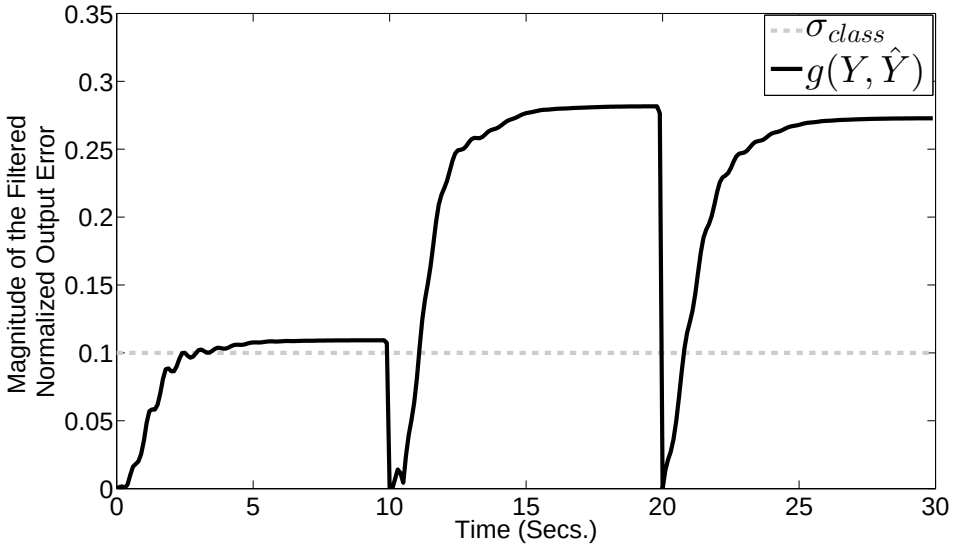

Figure 5.9: Output of the classifier $g(Y, \hat{Y})$ for the data shown in Figure 5.5. The figure shows that the threshold $\sigma_{class} = 0.1$ is violated for all three successive updates to the generator schedule.

Further, we define a classification epoch that spans the 10 second interval during which the set-point is unchanged, and we reset the classifier when Pg_i is updated. That is, every time Pg_i is updated the filter state is reset and the classification process is reinitialized. If the threshold σ_{class} is crossed during the 10 second interval, then the model output $\hat{Y}$ is classified as discrepant. Since the data from $\mathcal{P}$ are received as sampled values, the classification epoch of 10 seconds is discretized into K samples. The classifier used for the analysis is shown in Equation 5.7. The output of the classifier for the same experimental run shown in Figure 5.5 is shown in Figure 5.9. We see that the threshold is violated for three consecutive generator updates resulting in the classification $\mathcal{D}\hat{Y} = 1$. The consequence of the positive classification is that model adaptation is now required in order to resolve the discrepancy.

$$\mathcal{D}\hat{Y} = \begin{cases} 1, & \text{if } \sup_{k\in[1,K]} \left| \frac{(1-\tau)}{(Z-\tau)} \frac{(Y(k)-\hat{Y}(k))}{\hat{Y}(k)} \right| > 0.1 \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

5.4.2 Step 2: Identify adaptation candidates

Before we attempt to resolve the discrepancy in the model output, we first implement a strategy to decompose $\mathcal{M}$ and isolate only those components that require adaptation. The decomposition method relies on systematic exoneration of particular model components by logically reasoning about observed discrepancies and the known topological dependency between components. The result of the reasoning process is a set of hypotheses. Each hypothesis proposes one or more model components that possibly have incorrect parameters. The systematic decomposition procedure tests each hypothesis and refines the set to ultimately yield a selection of model components in $\mathcal{M}$ that have a high probability of being in need of adaptation.

This inference process is similar to that of diagnostic inference where the challenge is to identify a root cause fault responsible for an observed discrepancy [36], [119] and [149]. Our decomposition method uses many of the same tools as conventional diagnostic inference, except in our case they are used to identify a set of model components that are the root cause of the discrepancy between $\hat{Y}$ and Y.

Starting from the result of the first step of the decomposition process, which was the classification $\mathcal{D}\hat{Y} = 1$, we now delve into the compositional structure Γ of $\mathcal{M}$ (shown in Figure 5.7), to see that the output $\hat{Y}_{b_5}$ from model component b_5 is abstracted as the model output $\hat{Y}$. The objective of the reasoning process is to propose a set of model components that need to be adapted to resolve the discrepancy. For this simple example, we can do this by manual inspection of the component hierarchy and deduce that the observed discrepancy does not absolve any of the internal components of the model. Every model component m_i could potentially be malfunctioning resulting in the discrepancy in the output. A set of model components that could not all be functioning normally based on the observed discrepancy is called the *conflict set*. For the discrepancy $\mathcal{D}\hat{Y}_{b_5}$ the conflict set is a set of all the components in the model. The conflict is denoted: $\mathcal{D}\hat{Y}_{b_5} :< G_{1\cdots4}, b_{1\cdots5}, B_{15}, B_{31}, B_{21}, B_{43} >$.

As more measurements are received from the physical system new conflicts may emerge. In order to automate the recognition of conflicts, Modelica [43], [114] is used to model individual system components. Using Modelica we are able to build a compositional model of the electric circuit and automatically inspect the model to identify conflict sets for each observed discrepancy. In simple terms a conflict set for a given discrepancy is a set of components that constrain the value of the model output identified as discrepant. See [101], [34], [26] and [119] for a more detailed presentation of conflict recognition for similar model types.

The next step of the reasoning process is to propose a set $\mathcal{C}$ of adaptation candidates. Each candidate $c_i \in \mathcal{C}$ is a hypothesis for a set of model components that must be adapted as a group in order to resolve the discrepancy. For the example in Figure 5.7, we can see that if $\mathcal{D}\hat{Y} = 0$ then $\mathcal{C} = \emptyset$ (no component needs to be adapted). If $\mathcal{D}\hat{Y} = 1$ then the conflict set includes all the components in the model. Based on this conflict set, any component in the model is a valid hypothesis for an adaptation candidate. Therefore $\mathcal{C}$ is the power set of the conflict $\mathcal{D}\hat{Y}_{b_5} :< \cdot >$ minus the empty set. For a model with n components, where no candidate is absolved $|\mathcal{C}| = 2^n - 1$.

Several techniques exist to efficiently refine the set $\mathcal{C}$. A simple approach is to use some additional information about the system to discard or reinforce some of the candidates in the set. Fault diagnosis methods for large complex systems frequently use a rule-based or case-based reasoning approach to select candidates based on prior experience or domain expertise.

We assume that we have no additional information about $\mathcal{M}$ beyond the knowledge of its compositional structure and therefore use a model-based inference technique to refine the candidate set. There are several diagnosis algorithms in the research and commercial space including RAZ'R [131], LYDIA [46], DSI Express [62] or RODON [101] that use model-based reasoning to diagnose physical systems. The specific model-based reasoning algorithm we use is the General Diagnostic Engine

(GDE) [34] and [36].

The diagnosis algorithm requires:

1. Knowledge of the topological syntax $\Gamma\{\cdot\}$.
2. Knowledge of the output from each model component $\hat{Y}_{m_i}$.
3. Access to corresponding physical measurements Y_{p_i}.
4. A discrepancy classifier $g(\cdot, \cdot)$.
5. The ability to test hypotheses against model components m_i as required.

The algorithm builds a diagnostic lattice of the system as shown in Figure 5.10 depicting the space of all possible candidates in the model. The diagnostic process proceeds as more measurements from the internal components of $\mathcal{P}$ are received, compared against corresponding signals in $\mathcal{M}$ and classified. Each classified comparison between Y_{p_i} and $\hat{Y}_{m_i}$ results in a modification to the conflict set of the system. That is, a new set of components are determined that cannot all be functioning normally if the observed discrepancy is true. When new conflicts are detected, any previous candidates that no longer fully explain the conflict are discarded. The GDE algorithm features methods to manipulate these sets efficiently by only manipulating sets of components representing the smallest subset of components that still qualifies as a candidate (called a minimal candidate).

Figure 5.10 shows the first two steps of the candidate selection process. In response to the first conflict $\mathcal{D}\hat{Y}_{b_5} :< \cdot >$, since none of the components can be absolved, $|\mathcal{C}| = 2^n - 1$. By drawing an additional measurement (Y_{b_1}), we observe that $\mathcal{D}\hat{Y}_{b_1} = 1$ which generates a new conflict $\mathcal{D}\hat{Y}_{b_1} :< \cdot >$. The observations $\mathcal{D}\hat{Y}_{b_5}$ and $\mathcal{D}\hat{Y}_{b_1}$ considered together are inconsistent with the candidates $[b_5]$, $[B_{15}]$ and $[b_5, B_{15}]$ and are consequently eliminated. As more measurements are received from $\mathcal{P}$ additional candidates are either absolved or reinforced to further refine the candidate space.

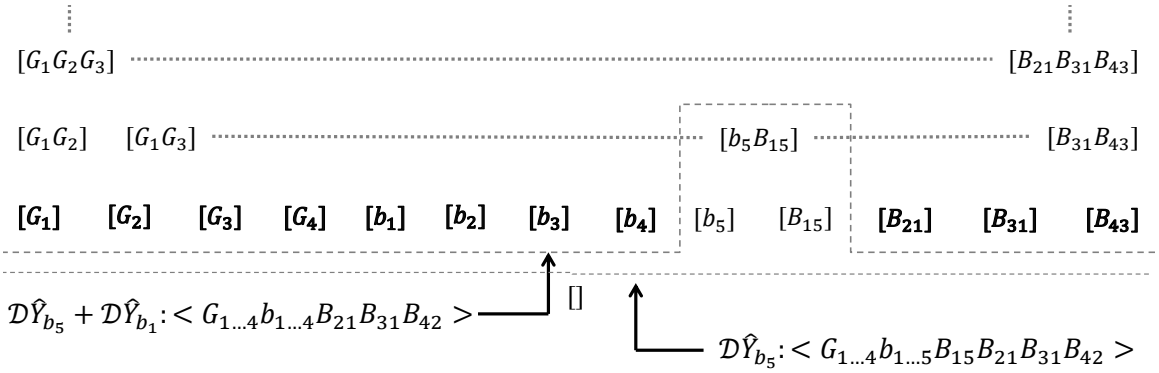

Figure 5.10: The figure show two steps of the candidate generation process. The minimal candidates are shown in bold and the dotted lines define a boundary below which candidates have been absolved.

In the upcoming section we will present a strategy to seek the best measurements from $\mathcal{P}$ from the perspective of shrinking the candidate space to isolate the true adaptation candidate with the fewest number of measurements.

5.4.3 Step 3: Select the best next measurement to refine the set of candidates - Guided Decomposition Approach

The decomposition approach we have discussed until now assumes that measurements from $\mathcal{P}$ can be drawn at will. Assuming that this is true, we can attach a diagnostic value to each prospective measurement Y_{p_i} from $\mathcal{P}$. Intuitively, measurements that result in greater differentiation between candidates are more valuable to the decomposition algorithm.

In a situation where there are several candidates generated in response to a conflict, we strategically extract measurements from $\mathcal{P}$ to differentiate between the candidates to isolate the true candidate with the fewest measurements. We do this using an approach called *guided probing* from fault diagnosis literature [34].

Referring to the diagnostic state illustrated in Figure 5.12, there are 12 potential measurements that can be drawn from $\mathcal{P}$ to further refine the set $\mathcal{C}$. For every comparison between Y_{p_i} and $\hat{Y}_{m_i}$ resulting in $\mathcal{D}\hat{Y}_{m_i} = k$, ($k = 1$ or 0), the candidates

in $\mathcal{C}$ can be divided into three categories:

1. The set of candidates that remain if $\mathcal{D}\hat{Y}_{m_i} = k$, called $R_i^{\mathcal{D}=k}$.

2. The set of candidates that are eliminated if $\mathcal{D}\hat{Y}_{m_i} \neq k$, called $S_i^{\mathcal{D}=k}$.

3. The set of candidates that cannot be eliminated irrespective of the value of $\mathcal{D}\hat{Y}_{m_i}$, called U_i.

The probability of each candidate being the true adaptation candidate can be computed using Bayes' rule. The conditional probability for each candidate is updated for each new measurement drawn from $\mathcal{P}$. Equation 5.8 shows the conditional probability for the first step of the decomposition process.

$$p(c_i|\mathcal{D}\hat{Y}_{m_i} = k) = \frac{p(\mathcal{D}\hat{Y}_{m_i} = k|c_i)p(c_i)}{p(\mathcal{D}\hat{Y}_{m_i} = k)} \tag{5.8}$$

The equation for conditional probability is reformulated for each category of candidates (See [34] for proof of this reformulation).

$$p(c_j|\mathcal{D}\hat{Y}_{m_i} = k) = \begin{cases} 0, & c_j \notin R_i^{\mathcal{D}=k} \\ \frac{p(c_j)}{p(\mathcal{D}\hat{Y}_{m_i}=k)}, & c_j \in S_i^{\mathcal{D}=k} \\ \frac{p(c_j)/2}{p(\mathcal{D}\hat{Y}_{m_i}=k)}, & c_j \in U_i \end{cases} \tag{5.9}$$

Some information may be available a-priori about the likelihood of some components over others. For example, generators may be more likely to have incorrect model parameters than bus elements. Using this information we can set the initial candidate probability $p(c_i)$ in Equation 5.9.

Given the conditional probabilities for each candidate based on a measurement from $\mathcal{P}$, we can use a value function to evaluate all the future choices for Y_{p_i} at every step. Continuing to use methods discussed in [34] we use information (or Shannon) entropy of the cumulative candidate probabilities ($H = -\sum p_i \log(p_i)$) as

a cost function for each potential measurement. Equation 5.10 shows the change in expected entropy for each possible measurement Y_{p_i}. The total expected entropy cost is a sum of both possible outcomes for $\mathcal{D}\hat{Y}_{m_i}$. The best possible next measurement from a diagnostic point of view is the measurement that minimizes ΔH_e.

$$\Delta H_e(Y_{p_i}) = p(\mathcal{D}\hat{Y}_{m_i}=1)\log p(\mathcal{D}\hat{Y}_{m_i}=1) + p(\mathcal{D}\hat{Y}_{m_i}=0)\log p(\mathcal{D}\hat{Y}_{m_i}=0) + p(U_i)\log 2$$

Where,

$$p(\mathcal{D}\hat{Y}_{m_i}=1) = \sum_{c_j \in S_i^{\mathcal{D}=k}} p(c_j) + \frac{\sum_{c_j \in U_i} p(c_j)}{2} \tag{5.10}$$

This guided decomposition approach is applied to our power systems model in Section 5.5.

5.4.4 Step 4: Identify parameters of the model-components in $\mathcal{C}_{min}$

Once the model is decomposed and the set of high probability candidates $\mathcal{C}_{min}$ has been isolated, we can apply the parameter identification methods discussed in Section 5.2 to resolve the model discrepancy.

A challenge in implementing this step of the process is that the parameter identification approach uses a state space model of the system while the decomposition process uses a compositional model. Therefore, before we can use the parameter identification algorithm we first have to produce a state space model for each component $m_i \in \mathcal{C}_{min}$.

From Section 5.1 we know that $\mathcal{M} = \Gamma\{m_i\}(\hat{\lambda}, \hat{x})$. The object-oriented modeling language we use explicitly portrays a model as a function of the compositional topology Γ, the set of model parameters $\hat{\lambda}$ and its internal state $\hat{x}$. This description syntax is used for all the component models $m_i \in \mathcal{M}$ as well so that model component m_i can be expressed as a function of Γ_{m_i}, $\hat{\lambda}_{m_i}$ and $\hat{x}_{m_i}$.

The state-space representation for m_i is not unique. Some general techniques used to algorithmically generate state space models from compositional models can be found in [147] and [79]. In the case of our motivating example we exploit the structure conserving Laplacian representation for the state-space model shown in Equation 5.3 and the assumption that $L_{lg} = 0$ to constrain the structure of the state-space model produced from the compositional form.

Equation 5.3 provides a specific format for each component model with knowledge of the relevant state variables $\hat{x}_{m_i}$, parameters $\hat{\lambda}_{m_i}$ and the classification of model components into dynamic models (such as Generators) in sub-matrix L_{gg} and algebraic models (such as buses and branch-lines) in sub-matrix L_{ll}.

The object-oriented description of the model components in the modeling language Modelica gives us the ability to define model classes in the manner we need. Each model component m_i is an instantiation of a pre-defined model class.

Once state-space models for the model components in $\mathcal{C}_{min}$ have been formed, the identification of the system parameters is a direct application of the methods in Section 5.3.3.

In Section 5.5 we will show the results of the candidate isolation and parameter identification techniques applied to our electrical network example.

5.5 Application of the method to the 5-bus example

Let us start with the flowchart in Figure 5.8. We know (based on the result in Figure 5.9) that $\mathcal{D}\hat{Y} = 1$. Therefore we proceed to Step-2 and explore the compositional structure of $\mathcal{M}$.

Based on the discrepancy $\mathcal{D}\hat{Y}_{b_5} = \mathcal{D}\hat{Y} = 1$, we generate the conflict $\mathcal{D}\hat{Y}_{b_5} :< G_{1\cdots4}, b_{1\cdots5}, B_{15}, B_{31}, B_{21}, B_{43} >$ and the corresponding candidate set $\mathcal{C}$ as shown in Figure 5.10.

In Step-3 we use the candidate refinement method discussed in Section 5.4.3.

One of the assumptions we made when proposing the cost function ΔH_e was that measurements Y_{p_i} could be drawn at will from $\mathcal{P}$. This is a fairly strong assumption from the perspective of practical implementation. We make this assumption for power distribution circuits in the light of some recent advances in networking technology and the enhanced penetration of the sensing and metering throughout the power grid that we outline below:

Reporting rates and data quality Networked solutions for real-time data acquisition and control over local area networks are fairly common place. This technology is mostly driven by requirements in the domain of industrial automation [112]. Recently the focus of this research has grown to include the automation of power distribution and monitoring. With this change in focus comes several new challenges mainly involving the great increase in physical scale from a network spanning tens of meters to tens of kilometers and the increase in the number of networked nodes from hundreds to thousands. Solutions for the challenges of increased scale include improvements to the backhaul network to support the demand for bandwidth [90] and specifications for data quality required for networked state estimation [91].

For all the simulations in this chapter, we abide by the data reporting rates proposed in [74]. That is, we sample data no faster than the proposed real-time transmission rate for regional distribution networks. By limiting the sampling rate, we assume that there are no issues related to data quality and that measurements reported from $\mathcal{P}$ are perfect. The extension of this work to a case where measurement data quality is uncertain is a natural next step.

On-demand availability of measurements A significant development in the network communication protocols used for controlling and monitoring power systems is the proposed IEC-61850 networking standard [72]. IEC-61850 is an

object-oriented messaging protocol [75], which means that the design of the protocol includes designated message classes for sampled values, commands etc. as well as a standardized communication model for every networked node. From a control perspective, it is possible to uniquely identify data on the network by translating its address into data-type, source and function.

This technology is critical to the practical application of our methods since it allows the decomposition algorithm to evaluate the bandwidth utilization cost and efficiently address each desired physical measurement Y_{p_i}.

Pervasive sensing infrastructure In our analysis we assume that it is possible to measure the output of every physical component $p_i \in \mathcal{P}$. This assumption is motivated by electric grid instrumentation roadmaps in [44] and [38]. The roadmap documents propose a greater penetration of sensors at the level of electric distribution as well as the use of advanced PMUs to offer unprecedented access to measurements from within electric microgrids.

While we assume that measurements are available from every component for the analysis in this chapter, this assumption is not essential to its implementation. With the use of the the IEC-61850 protocol, it is possible to identify measurements that are not available and to attribute a high enough cost to them that they are never invoked by the decomposition algorithm.

For now we will use a simple link model [155] to calculate the network utilization cost $\mathcal{U}Y_{p_i}$ for each measurement Y_{p_i}. Assuming finite bandwidth availability, Total Bandwidth Utilization $= \Sigma \mathcal{U}Y_{p_i} + \Sigma \mathcal{U}P_{gi} = 1$.

For a flat network, the utilization cost for each network transmission is the same. However, most communication networks are hierarchical. The IEC-61850 protocol also implements a virtual communication hierarchy by logically grouping nodes.

Figure 5.11 is an illustration of a communication hierarchy applied to our power

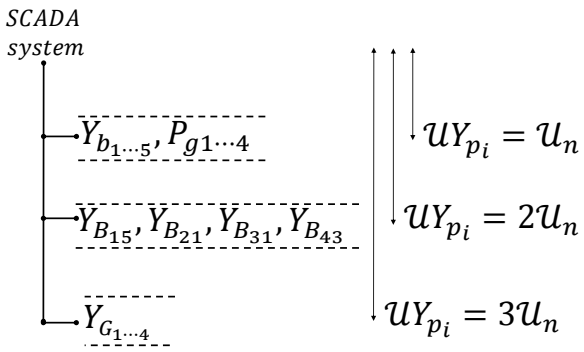

Figure 5.11: The figure illustrates the network hierarchy in the model used to calculate the link utilization cost for each measurement.

grid example. $\mathcal{U}_n$ is the *per link* utilization cost. In order to satisfy the total bandwidth constraint, $9 \times \mathcal{U}_n + 4 \times 2\mathcal{U}_n + 4 \times 3\mathcal{U}_n = 1$. Therefore, $\mathcal{U}Y_{b_i}, \mathcal{U}P_{gi} = 0.0345$, $\mathcal{U}Y_{B_{jk}} = 0.069$ and $\mathcal{U}Y_{G_i} = 0.1034$.

The total cost for drawing a measurement Y_{p_i} for the decomposition process is $\mathcal{V}Y_{p_i} = \mathcal{U}Y_{p_i} + (\Delta H_e(Y_{p_i}) + 1)$. Table 5.1 shows the total cost $\mathcal{V}Y_{p_i}$ calculated at each step of the decomposition process. The signal with the lowest cost is shown in bold lettering and is drawn from $\mathcal{P}$. Once a measurement is drawn, the inference process discussed Section 5.4.3 is repeated to calculate the new candidate probabilities resulting in a new set of values for $\mathcal{V}Y_{p_i}$ for the remaining measurements. Figure 5.12 is a graphical illustration of the decomposition process as measurements are drawn in the order specified in Table 5.1. The shaded regions in the figure illustrate the shrinking space of probable candidates as new measurements are added.

Step-3 of the decomposition process terminates when either one candidate is identified with high enough probability or when no other measurements are available. In the case of our example, the process terminates when $\mathcal{C}_{min} = [G_2]$. Table 5.1 also shows the probability of candidate $[G_2]$ after each probing cycle. We see that the likelihood of $[G_2]$ as an adaptation candidate increases continuously until the termination condition $p(c_j) > 0.9$ is reached.

Proceeding to Step-4, we see that G_2 is a generator model of the form shown

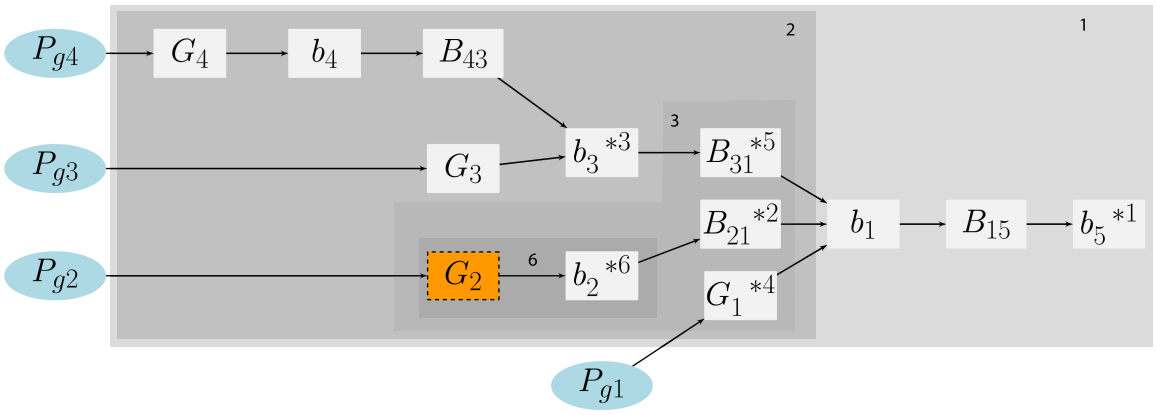

Figure 5.12: Graphical illustration of the model decomposition process. Measurements are drawn from the components marked with an asterisk following the order in Table 5.1. The shaded regions in the figure show the shrinking space of probable candidates after probing steps 1, 2, 3 and 6.

Table 5.1: Measurement costs $\mathcal{V}Y_{p_i}$ and the conditional probability of candidate $[G_2]$ after each new measurement.

$\mathcal{V}Y_{p_i}$	Y_{G_1}	Y_{G_2}	Y_{G_3}	Y_{G_4}	Y_{b_1}	Y_{b_2}	Y_{b_3}	Y_{b_4}	Y_{b_5}	$Y_{B_{21}}$	$Y_{B_{31}}$	$p([G_2])$
Initial	1.00	1.00	0.54	0.59	1.00	0.43	0.35	0.47	**0.32**	0.39	0.36	*0.12*
Probe-1	0.48	0.57	0.57	1.00	0.63	0.44	0.44	0.54	-	**0.39**	0.45	*0.07*
Probe-2	0.50	1.00	0.54	0.59	0.68	0.61	**0.35**	0.47	-	-	0.36	*0.21*
Probe-3	**0.49**	0.85	1.00	1.00	0.85	0.74	-	1.00	-	-	0.51	*0.38*
Probe-4	-	1.75	1.00	1.00	1.98	1.69	-	1.00	-	-	**0.50**	*0.72*
Probe-5	-	0.70	1.00	1.00	0.71	**0.68**	-	1.00	-	-	-	*0.99*

in Equation 5.5 and accordingly apply the subspace identification algorithm to it as discussed in Section 5.3.3. The model for the generator has two dynamic states ($n = 2$) and assuming that the sampling frequency is $10Hz$ for a data collection period of $10secs.$, $N = 100$. Therefore the complexity of the subspace identification algorithm $\mathcal{O}(n^4 + N^2) = \mathcal{O}10^4$ as opposed to $\mathcal{O}10^5$ for the entire system.

Figure 5.13 shows the output of $\mathcal{M}$ and $\mathcal{P}$ after the parameters for generator model for G_2 have been corrected. The figure shows that good correspondence has been restored between the two systems.

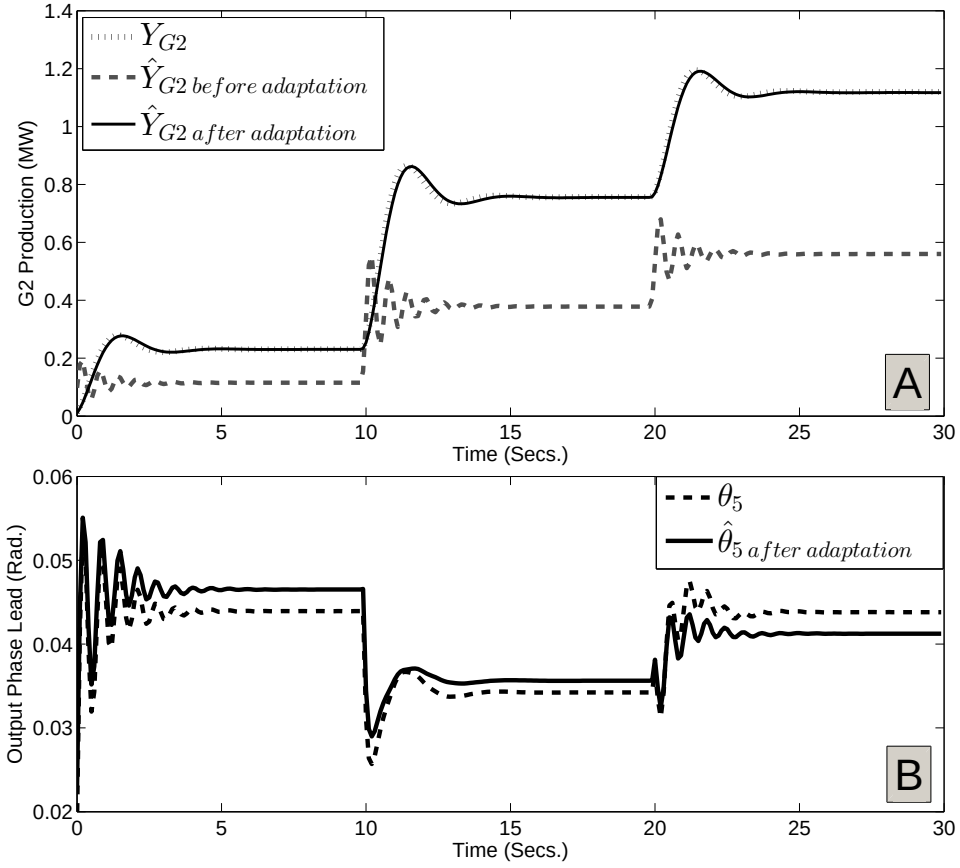

Figure 5.13: Sub-Figure A shows the output of m_{G_2} before and after adaptation. Sub-Figure B shows the output of the full model $\mathcal{M}$ after adaptation. Close matching is observed between the two systems once the parameters for m_{G_2} have been corrected.

5.6 Conclusion

This chapter addresses some of the challenges in adapting simulation models of large systems in a networked setting. An electrical power system is used as an illustration of such a system. The core contribution is an approach to simplify the adaptation of a system level model vis-à-vis updating its internal parameters. The strategy presented here systematically decomposes the system model while isolating model components that need to be updated. This targeted adaptation approach significantly reduces the complexity of the parameter identification process.

The decomposition algorithm compares the output of specific model components to corresponding outputs in the physical system in order to either absolve or reinforce hypotheses for sets of model components that require adaption. A framework is also proposed to optimize the decomposition algorithm, trading off the cost of obtaining a measurement against its contribution towards differentiating between hypotheses. The method presented in this chapter assumes some specific conditions in its implementation. These pertain to the application domain of electrical power systems (known topology), modeling methods (compositional models, structure preserving state-space form) and network properties (access to real time measurements).

CHAPTER VI

Modifying model topology to reduce noise and complexity in yield analysis for manufacturing process workflows

In this chapter we extend the topological exploration and inference methods present in Chapter V and propose methods to simplify model topology and relax the effects of measurement noise to improve the performance of the inference process. The work presented in this chapter has been submitted to *IEEE Transactions on Semiconductor Manufacturing*.

6.1 Introduction

In this chapter we will apply the diagnostic approach developed in Chapter V to isolate sources of yield loss in a large semiconductor manufacturing process. Yield is defined as the ratio of the number of usable parts after the completion of a manufacturing process to the number of potentially usable items at the beginning of the process. It is an important performance metric for manufacturing facilities, and the economic significance of yield loss is particularly highlighted in the high value, high throughput and highly automated manufacturing processes used to produce semiconductor integrated circuits [88].

The schematic in Figure 6.1 shows a simple semiconductor manufacturing workflow ($\mathcal{P}$). For the rest of our analysis in this chapter we will only consider front-end processes in $\mathcal{P}$. Front-end processes refer to the processing steps applied to the silicon wafer to build up or etch semiconductor structures. The output of the front end process is a matrix of fully formed dies on a wafer. Yield assessed by end-of-line sample testing is called the "wafer sort yield". The front end processes in Figure 6.1 are indicated with a dashed line. The wafer sort yield of $\mathcal{P}$ depends on the performance of each manufacturing step p_i as well as on variations in process variables such as temperature, voltage, ion concentration and particle contamination [25], [55], [87]. Wafer measurements (such as dimensions of important features, thickness of deposits, chemical properties of dopant layers) and measurements of process parameters (such as plasma temperature) may be available from intermediate steps in the process, though the number of measurement sites and frequency of measurement is typically limited due to cost and throughput constraints. In Figure 6.1, intermediate measurements are only available from process steps p_2 and p_4. All wafer measurements and parameter measurements from p_i are collectively called z_i. Since only a small sample of wafers are subject to end-of-line testing, wafer classification algorithms are used to estimate yield on every wafer based on available in-process measurements $\mathcal{Z} = \{z_2, z_4\}$ [16], [60], [69], [29], [13]. While wafer classification methods partially replace the need for exhaustive end-of-line testing, classification is usually done at the end of the manufacturing line once all measurements corresponding to a particular wafer have been collected. If a wafer (or die) is classified as defective, there is no mechanism to correct or reverse the defects.

Modern process control techniques envision a prediction driven approach for quality control [111]. Accurate forecasts of process measurements based on process models and historical data allow the system to reactively mitigate the effect of random quality variations as well as to reject defective wafers early in the workflow if further

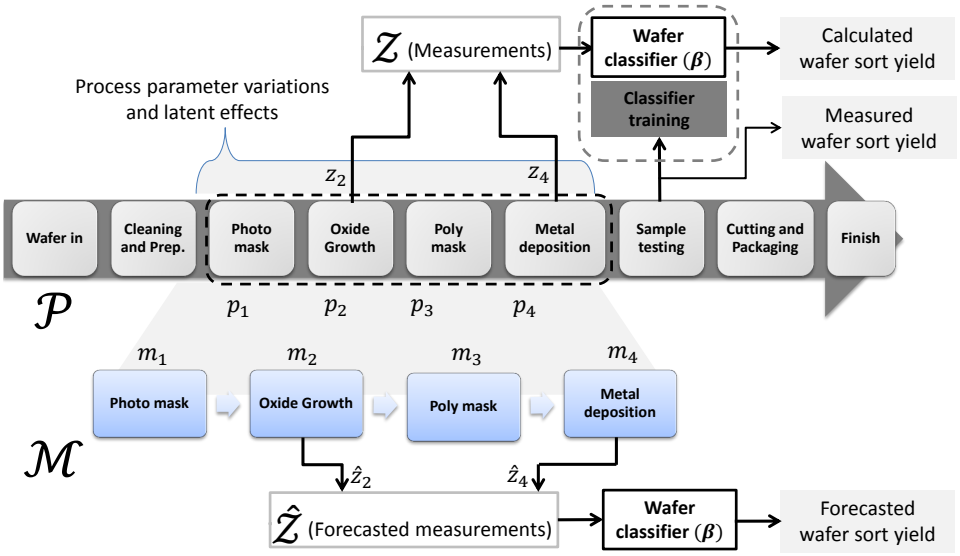

Figure 6.1: The schematic shows a nine step manufacturing process ($\mathcal{P}$) and a model of four front-end processes ($\mathcal{M}$). The figure also shows two sites where physical measurements are available from $\mathcal{P}$.

processing is deemed imprudent [37] [41], [113].

Measurement forecasts for each wafer are provided by the model ($\mathcal{M}$) of the manufacturing process shown in Figure 6.1. $\mathcal{M}$ is comprised of model components $m_1 \cdots m_4$ corresponding to process steps $p_1 \cdots p_4$. Each model component m_i is used to forecast measurements of wafer properties and process parameters of the corresponding process step p_i. With accurate modeling we can compositionally assemble model components together ($\mathcal{M} := m_1 \rightarrow m_2 \rightarrow \cdots m_4$), as shown in Figure 6.1, in order to forecast measurements for the entire line and therefore forecast the end-of-line yield.

Figure 6.2 shows three process steps (1, 2 and 3) of the manufacturing line. The model m_2 and the corresponding physical process p_2 may be thought of as one abstract 'step' (Step-2) in the process. m_2 takes historical measurements from p_2 as well as external inputs such as the measurement forecasts from preceding steps and internal model parameters to forecast the expected measurement $\hat{z}_2$ (for a particular wafer) at the end of Step-2. Assuming the forecast is accurate, it is commonly used by predictive

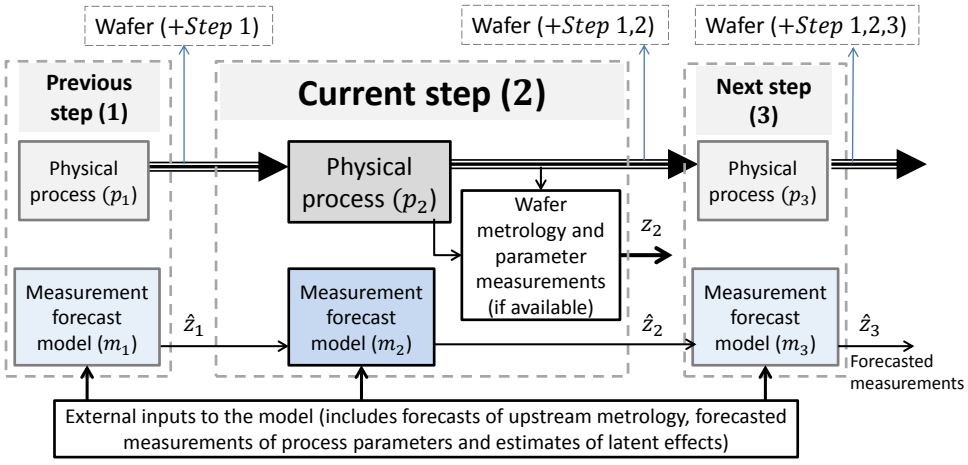

Figure 6.2: The schematic shows the parallel implementation of a measurement forecast model m_2 and a physical process p_2 in Step-2 of the manufacturing line. The figure also shows that measurement z_2 depends on Step-1 as well as other external inputs.

process control algorithms [37] [113] to correct variables in p_2 to compensate for minor process drifts. Larger differences between the forecast and the actual measurement require investigation into the source of the discrepancy. Each model component m_i forecasts a measurement at the end of step i and the forecasts are carried forward through successive model components culminating in a forecast for end-of-line wafer sort yield.

The problem we are addressing in this chapter is that a discrepancy or mismatch is observed between the "Forecasted wafer sort yield" using $\mathcal{M}$ and the "Calculated wafer sort yield" $\mathcal{P}$ for a manufacturing process. Clearly, this discrepancy could result from a mismatch between any subset of preceding process steps p_i ($i \in \{1 \cdots 4\}$) and their corresponding forecast models m_i. Our objective is to perform a forensic analysis by comparing $\hat{\mathcal{Z}}$ and $\mathcal{Z}$ to isolate the steps responsible for an observed end-of-line yield discrepancy.

Typical manufacturing process workflows have hundreds of steps and only a few sites where physical measurements are available. Further, measurements from a single step depend on upstream steps (which may not be directly measurable), internal

process parameters and other external factors. Clearly, the analysis to identify the steps responsible for an end-of-line discrepancy is not trivial and a formal analysis technique is needed to maximize the diagnostic impact of available measurements. We use Bayesian inference to analyze the network of dependencies between process steps and iteratively exonerate or implicate key steps when measurement comparisons are made. However, computational complexity and uncertainty in measurements due to poor data quality limit the size of the manufacturing workflow to which the Bayesian diagnosis approach can be applied directly.

Our contributions to the identification and isolation of sources of yield loss in a semiconductor manufacturing line are:

1. A graph theoretic formulation to represent a manufacturing line as a network of dependencies and a case study of a semiconductor manufacturing line, presented in Section 6.2.

2. A method for systematic comparisons between model forecasts and measured data suited for use with a Bayesian diagnosis scheme, presented in Section 6.3.

3. Graph abstraction and relaxation strategies to reduce the complexity of the diagnosis process, presented in Section 6.5.

4. An algorithm to strategically reverse some of the graph simplifications, when needed, to refine the inference result, presented in Section 6.6.

5. A solution verified to reduce the net uncertainty and noise in the inferred result of the diagnostic analysis, presented in Section 6.7.

The analysis techniques we develop in this chapter make no assumption about the physical properties of the individual steps and can be used with any process that can be reduced to a network of causal dependencies. An application of the method

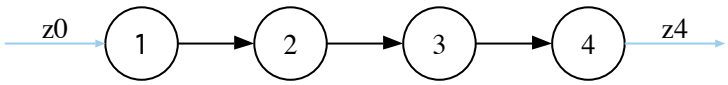

Figure 6.3: A graph representation of a four step sequential process workflow.

to a full scale model of a manufacturing line is presented in Section 6.8 and the contributions of the chapter are summarized in Section 6.9.

6.2 Graph representation for manufacturing processes

We will represent a manufacturing process as a graph of edges and vertices. In the context of semiconductor manufacturing, Step-2 in Figure 6.2 represents a front end process such as a plasma etching step. The physical process p_2 corresponds to a specific instance of a tool and a process recipe used to perform the etching operation. Step-2 is also equipped with instrumentation to measure wafer and process parameters at the output of p_2. The model m_2 used to forecast measurements may be based on process physics, such as the model provided in [65] to predict the profile of etched geometry using chemical kinetics, or based on statistical inference as in [129] and [134]. All the elements contained in Step-2 can be abstracted into an atomic representation called a vertex.

Step-2 interacts with adjacent steps through an explicit set of dependencies, such as the order in which steps are executed. In Figure 6.2, Step-2 sequentially follows Step-1 and every product processed at Step-2 is first processed at Step-1. Step-2 is therefore causally dependent on Step-1, represented by an edge from Step-1 to Step-2 in graph representation.

A set of components with a declarative description of dependencies between components is said to have a compositional structure [123]. The sequence of process steps (Step-1 $\rightarrow$ Step-2 $\rightarrow$ Step-3) shown in Figure 6.2 is compositional in nature. We will use a directed graph (digraph) notation to represent the compositional structure of a

manufacturing process. A graph is a general mathematical abstract used to express networks of co-dependent objects, processes or agents [21]. It is expressed as a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ comprising a set $\mathcal{V}$ of vertices together with a set $\mathcal{E}$ of edges which are 2-element subsets of $\mathcal{V}$. In case the dependency between vertices is asymmetric, the set $\mathcal{E}$ is made up of ordered pairs of vertices such that $e = (x, y) \in \mathcal{E}$ is considered to be directed from x to y. y is the direct successor of x and x is said to be a direct predecessor of y. The graph in Figure 6.3 represents the compositional structure of a 4-step manufacturing process in a serial line configuration where process steps occur sequentially from 1 to 4. Realistic manufacturing lines are generally more complex in construction than the structure in Figure 6.3 incorporating branches and parallel segments to improve reliability, balance production throughput and accommodate greater product diversity [85].

Additionally, the presence of models within each step introduces additional dependencies in the compositional description. Models used in the prediction framework take as inputs (and therefore depend on) several external factors including process variables and forecasted measurements from other steps. The compositional representation of a real manufacturing workflow can therefore become quite complex when every step is represented along with all its dependencies.

We will now describe graphs that capture the structural features of realistic semiconductor manufacturing workflows.

6.2.1 Serial-Parallel lines

Semiconductor manufacturing lines are usually designed to distribute operations between process stations in such a way as to minimize production bottlenecks and maximize overall throughput and yield. The optimization of a manufacturing line to achieve all goals is achieved in part by modifying the arrangement of individual process steps in the workflow. A common modification used to improve line balance

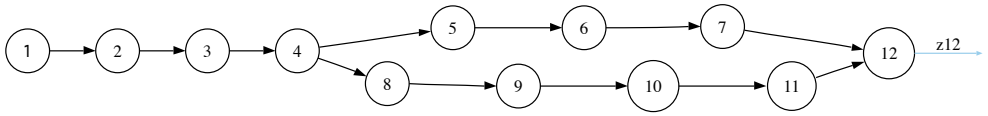

Figure 6.4: A graph representation of a serial-parallel line.

is to perform exclusive or repeating operations in parallel, introducing serial-parallel segments within the workflow [85]. Figure 6.4 shows a graph of a 12-step serial-parallel line. The presence of parallel segments in the line introduce branches in the graph. In Figure 6.4, vertices 4 and 12 are branch vertices.

6.2.2 Independent source vertices

Manufacturing processes frequently include steps where components are introduced into the line somewhere between the beginning and the end. In the context of semiconductor manufacturing, mask application or metal deposition steps might be considered entrant processes. In the graph notation we represent entrant processes as a source vertex (in-degree $= 0$). The edge $(15, 5)$ in Figure 6.5 indicates the introduction of an entrant process 15 at Step-5.

Similar notation is also used to represent dependencies of a step on process parameters such as supply gas pressure or supply voltage [55]. In Figure 6.5, vertex 15 could represent a process parameter that affects the output of vertex 5. A single parameter may also influence multiple vertices. Consider for example, a shared high voltage supply or a common plasma generator. Shared process parameters are represented by source vertices with out-degree > 1. In Figure 6.5, vertex 13 influences the output of both vertices 3 and 9. While not all process parameters may be instrumented for measurement, if a measurement is available then we assume that its value is deterministically known and included in the set of physical measurements $\mathcal{Z}$. If a process parameter is unmeasured, its involvement in the observed yield discrepancy can be inferred based on the output measurements of other downstream vertices.

6.2.3 Latent effects

Based on discussions with semiconductor manufacturing experts, being able to include the effects of exogenous errors, random defects and measurement uncertainty is essential for the practical application of Bayesian diagnostic tools. These non-deterministic *latent effects* are un-modeled and often distributed throughout the manufacturing process. They are cumulatively manifested as a maximum practical threshold for the end-of-line yield, i.e a finite probability that a manufactured wafer will be defective even if every controllable process variable is tuned correctly. This defect probability is assumed to be attributed to a set of latent effects that could include the probability of damage due to contaminants, preexisting defects in the wafer substrate or any other unattributed stochastic process [87]. The number of latent effects used to justify the cumulative defect probability is somewhat arbitrary, but if information is available on the relative contribution of each latent effect to the cumulative defect rate, and further if a direct causal connection is found between a certain latent effect and subset of process steps, then it can be introduced into our digraph representation as a source vertex v_l with an error contribution p_{v_l}.

Consider, for example, that the processes represented by vertices 7, 11 and 12 share the same part handling equipment or wafer transfer system. While no explicit model may exist for the performance of the handling equipment, a statistical error rate (p_{14}) recording a probability of misalignment or damage may be available. Vertex 14 represents the wafer transfer system's contribution to the cumulative defect rate with the additional information that it directly affects vertices 7, 11 and 12.

6.2.4 Loops or cycles

The analysis method in this chapter is directly applicable to connected acyclic digraphs such as Figure 6.5. However, manufacturing lines often contain loops where certain steps are repeated multiple times. It is therefore necessary to relax the acyclic

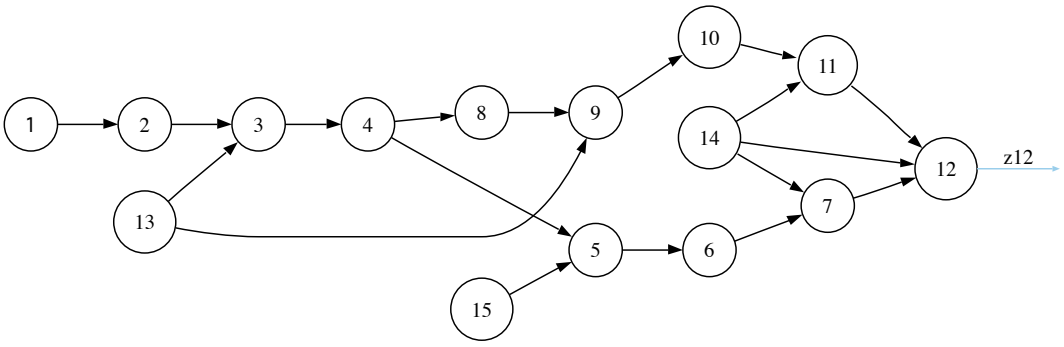

Figure 6.5: A graph representation of a manufacturing line including external factors represented by vertices 13, 14 and 15.

condition and admit small loops (or cycles) [156]. Let $P_1(uv)$ and $P_2(vu)$ be two different simple paths between vertices u and v; then the walk $P_1 + P_2$ contains a cycle. Assume P_1 is a forward path in the directional sense of a digraph, then we specify a 'small' cycle as a limit on the length of P_1 ($|P_1| \leq L_f$). While there is no formal method to limit L_f, we use the heuristic $L_f = \frac{|\mathcal{E}_1|}{30}$. Based on the sample set of graphs of practical semiconductor production lines that we have access to, graphs that meet this small cycle condition are compatible with the methods presented in this chapter.

Let the graph in Figure 6.5 be called $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$. $\mathcal{G}_1$ captures key structural attributes of a realistic manufacturing process and will serve as a running example to explain our technique throughout this chapter. Note that the vertex set $\mathcal{V}_1$ represents all steps, process parameters and latent effects considered in the manufacturing line. Every measurement $z_i \in \mathcal{Z}_1$ represents a set of physical observations from a vertex $v_i \in \mathcal{V}_1$. Since we assume that some of these vertices are not directly measured, then typically $|\mathcal{Z}_1| < |\mathcal{V}_1|$. However, depending on the modeling detail, a forecasted measurement may be available for every vertex in $\mathcal{V}_1$. We are only concerned with forecasted measurements $\hat{z}_i \in \hat{\mathcal{Z}}_1$ that have a corresponding physical measurement $z_i \in \mathcal{Z}_1$ so that we may draw comparisons between the two. It is assumed for the rest of this chapter that $|\hat{\mathcal{Z}}_1| = |\mathcal{Z}_1|$.

6.3 A scheme for comparing the outputs of $\mathcal{M}$ and $\mathcal{P}$.

As discussed in Section 6.1, our objective is to isolate a subset of vertices in $\mathcal{V}_1$ that can be identified as sources of yield loss. We trigger the start of our comparison analysis when a discrepancy is recorded between the calculated wafer sort yield and the forecasted wafer sort yield. Our definition of a 'discrepancy' is described below:

Suppose $|\mathcal{V}| = m$ and $|\mathcal{Z}| = n$. Elements $z_i \in \mathcal{Z}$ are variables representing specific measurement sites in the process. Each wafer sample $[x]$ exiting the line is a specific instance $\mathcal{Z}[x]$ of the available measurements. The wafer sort yield for wafer $[x]$ can be calculated based on $\mathcal{Z}[x]$ using a measurement based classifier [55], [60]. A classifier uses measurements $\mathcal{Z}$ to group produced wafers based on a metric of interest. If the metric chosen is the quality of the wafer, then a classifier may be used to classify $[x]$ based on the type, severity and number of defects. Without loss of generality, we can restrict the measurement based wafer classification of calculated wafer sort yield to a binary choice. Each wafer is classified as *acceptable* or *defective* based on available measurements $\mathcal{Z}[x]$ for the wafer. We use a data driven classification approach discussed in [16], [33] and [29] to classify each wafer.

First, a linearly independent basis of k principal features $\mathbb{F} = \{f_1, \cdots, f_k\}$ is identified for the set of possibly correlated measurements $\mathcal{Z}$. Assume $\mathbb{F}$ is a linear normed basis and $T : \mathcal{Z}^n \to \mathbb{F}^k$ is a norm preserving orthogonal transformation. Let I and I' be two adjacent regions of $\mathbb{F}$. A hyperplane $H = \{\alpha \cdot f = a \ \forall f \in \mathbb{F}\}$ separates I and I' if I and I' lie on opposite sides of H. A support vector machine (SVM) [141] model is used to define H in order to provide a maximal marginal separation [163] between acceptable wafer samples and defective wafer samples. I and I' are accordingly labeled the *acceptable region* and the *defective region* of $\mathbb{F}$. The development of $\mathbb{F}$, T and H together comprise the training process for the wafer classifier. Once the classifier is trained, every wafer sample $\mathcal{Z}[x]$ may be classified as either defective or acceptable.

For brevity we denote the classification algorithm as a function $\beta(\cdot)$ acting on a wafer sample so that $\beta(\mathcal{Z}[x]) = 1$ if wafer x is classified as *defective* and $\beta(\mathcal{Z}[x]) = 0$ otherwise.

The classifier $\beta(\cdot)$ is also used on the forecasted measurements $\hat{\mathcal{Z}}[x]$ to predict yield excursions for wafer x before production is complete. If $\beta(\hat{\mathcal{Z}}[x]) = 1$ then remedial action may be taken preemptively during production or the production on wafer x may be aborted to limit the resource investment on x. We are interested in a case where the wafer x is erroneously forecasted to be acceptable during production ($\beta(\hat{\mathcal{Z}}[x]) = 0$) and then re-classified as defective ($\beta(\mathcal{Z}[x]) = 1$) once the wafer reaches the end of the manufacturing line and all the elements in $Z[x]$ are acquired. This is a serious condition since the prediction framework using $\mathcal{M}$ was unable to predict a yield excursion for wafer x resulting in an irrecoverable defective wafer. We call this condition a *discrepancy*.

The opposite case where $\beta(\hat{\mathcal{Z}}[x]) = 1$ and $\beta(\mathcal{Z}[x]) = 0$, while unlikely, may also be considered a discrepancy and can be easily considered in our analysis. We limit ourselves to the former case for ease of illustration.

We also know that the wafer has been classified as defective after all available measurements have been considered. We assume this classification is more reliable than a classification based on a prediction and so we use $\beta(\mathcal{Z}[x]) = 1$ as a ground truth for our diagnostic analysis and attempt to identify where the misclassification $\beta(\hat{\mathcal{Z}}_1[x]) = 0$ originated. It must be pointed out that the choice of ground truth or datum is purely a notational convenience allowing us to call $\beta(\hat{\mathcal{Z}}[x]) = 0$ a discrepancy and $\beta(\hat{\mathcal{Z}}[x]) = 1$ not, without specifying that $\beta(\mathcal{Z}[x]) = 1$ at every stage.

The observed discrepancy $\beta(\hat{\mathcal{Z}}[x]) = 0$ is a symptom of one or more model components $m_i \in \mathcal{M}$ differing in their behavior from their corresponding process step, we call this a model-process *conflict*. The goal of our analysis is to identify a subset of conflicted steps responsible for the end-of-line discrepancy.

A straight forward approach to isolate steps is to assume independence between measurement sites $z \in \mathcal{Z}$ and to compare the output of each measurement site z_i to a corresponding forecast $\hat{z}_i$ such that a norm $\|z_i - \hat{z}_i\| > \delta_i$ indicates a discrepancy originating from Step-i. This approach is not practically feasible since measurement sites often show strong mutual correlation resulting in a feature basis $\mathbb{F}$ of significantly reduced dimension ($n >> k$). The resulting kernel space of T implies a surjective mapping from $\mathcal{Z}$ to $\mathbb{F}$ and the lack of a unique inverse transformation ($T^{(-1)} : \mathbb{F} \to \mathcal{Z}$). In other words, the decision plane $H \in \mathbb{F}$ is ill posed when mapped on to $\mathcal{Z}$. As a result, the marginal separation metric defined on $\mathbb{F}$ cannot be reduced to an error norm between an individual measurement site $z_i \in \mathcal{Z}$ and a corresponding forecast $\hat{z}_i \in \hat{\mathcal{Z}}$.

We need a systematic approach to draw comparisons between elements in $\mathcal{Z}$ and $\hat{\mathcal{Z}}$ in order to reason about the comparisons and propose best candidates for sources of end-of-line yield loss.

Returning to the graph notation for Figure 6.5, each vertex represents a process step and we can infer from the classification $\beta(\hat{\mathcal{Z}}_1[x]) = 0$ that at least one vertex in the set $\mathcal{V}_1$ is conflicted. The set of all possibly conflicted vertices is called a *conflict set*.

The objective of our analysis is to find the minimal conflict set $c^* \subseteq \mathcal{V}_1$ that completely explains the observed discrepancy; i.e., c^* only contains vertices that must all be conflicted in order to explain the observed discrepancy. We start with all possible candidates c_i for c^*. The set of all possible candidates is called the *candidate set* $\mathcal{C}$. It is easy to see that $\mathcal{C}$ is the power set of $\mathcal{V}_1$ or $\mathcal{C} = 2^{\mathcal{V}_1} = \{[\emptyset], [1], [2], \cdots, [1,2,3], \cdots, [1,2,3,4,5,6], ...\}$. The candidate $[\emptyset]$ is discarded from $\mathcal{C}$ after the classification $\beta(\mathcal{Z}_1[x]) = 1$ since it does not explain the discrepancy. The remaining candidates are reinforced by the observation since any combination of vertices in $\mathcal{V}_1$ are valid candidates for c^*.

We begin by designing a replacement strategy for $\mathcal{G}_1$. Consider a measurement site $z_i \in \mathcal{Z}$ drawn at random. We can substitute the measurement z_i in place of the corresponding $\hat{z}_i \in \hat{\mathcal{Z}}$ to produce the hybrid data sample $(\hat{\mathcal{Z}}[x] \setminus \hat{z}_i[x]) \cup z_i[x]$ for wafer x. We call this operation a *single point replacement* denoted $D(z_i)$ (essentially constraining a single vertex to its datum value). By applying the wafer classifier to the hybrid data sample, we see that both outcomes $\beta(D(z_i)) = 1$ and $\beta(D(z_i)) = 0$ are possible.

The outcome $\beta(D(z_i)) = 1$ implies that the end-of-line discrepancy can be resolved by replacing the predicted measurement $\hat{z}_i$ with the real measurement z_i. Suppose a simple forward path P_{hi} exists from v_h to v_i where v_h is the first vertex with a physical measurement along the path $\bar{P}_{hi}$, then the vertices touched by P_{hi} comprise the set $V_{P_{hi}}$. Intuitively, the vertex or vertices that lie in the set $\{V_{P_{hi}}\} \setminus v_h$ may have a role to play in the discrepancy and therefore any suitable candidate for c^* must include the set $\{V_{P_{hi}}\} \setminus v_h$.

Now suppose $\beta(D(z_i)) = 0$. This means the substitution has not corrected the end-of-line discrepancy, implying that the vertices responsible for the discrepancy are not fully contained in the set $\{V_{P_{hi}}\} \setminus v_h$. As a result, any candidate comprised only of vertices in $\{V_{P_{hi}}\} \setminus v_h$ can be exonerated. Both outcomes for $\beta(D(z_i))$ result in refinement of the set $\mathcal{C}$ and the algorithm proceeds as more replacements are made on $\mathcal{G}_1$.

Recall that the mapping from $\mathcal{Z}$ to $\mathbb{F}$ is surjective in nature. As a result, the projection of $D(z_i)$ on $\mathbb{F}$ is not unique and it is possible that $T((\hat{\mathcal{Z}} \setminus \hat{z}_i) \cup z_i) \in \mathit{span}\ T(\hat{\mathcal{Z}} \setminus \hat{z}_i)$. Consequently, the inference problem is under-constrained when substitutions are made one at a time. We address this problem by increasing the number of simultaneous substitutions by one after a round of n single point replacements. At the limit, $n-1$ simultaneous substitutions are made generating $\sum_{r=1}^{n-1} \frac{n!}{r!(n-r)!} = 2^n - 2$ replacements. The practical limit on the number of simultaneous replacements required

depends on the graph structure. In most cases, we find that limiting the algorithm to one ($D(z_i)$) and two ($D(z_i, z_j)$) point replacements ($\frac{n^2+n}{2}$ replacements) results in sufficient candidate differentiation for conflict convergence. In future references to replacements, we use $D(v_i)$ instead of $D(z_i)$ for notational convenience. $D(v_i)$ implies that the z_i replaces predicted output $\hat{z}_i$ for vertex v_i.

The replacement policy presented here offers a systematic approach to exhaustively compare the outputs of $\mathcal{M}$ and $\mathcal{P}$, but makes no effort to simplify the problem. This replacement strategy becomes computationally complex for large graphs and at the worst case, when $(n-1)$ simultaneous substitutions are needed, is intractable for larger problems such as practical semiconductor manufacturing lines. We will quantify this complexity in Section 6.4 and then present methods to simplify the problem while reducing noise in the diagnosis process in Sections 6.5 through 6.7.

6.4 A review of Bayesian methods for diagnosis

With every new replacement, we analyze the graph, compute the impact on $\mathcal{C}$ and refine it by discarding candidates that are no longer valid. The formal strategy we use to achieve this refinement is the Bayesian inference algorithm GDE presented in [36], [119] and [149]. Our refinement strategy uses the same basic inference framework as used in Chapter V, but additionally considers special structural properties of graphs used to model semiconductor manufacturing lines.

We begin by assigning a prior probability to each candidate $p_0(c_i)\ \forall c_i \in \mathcal{C}$ by using a-priori knowledge about the likelihood of some conflicts over others if available. For example, an oxide growth step may be more likely to have incorrect model parameters than a metal deposition step. For the analysis in this chapter, we assume a uniform probability distribution across the elements in $\mathcal{V}_1$ (all model components are equally likely to be conflicted) to compute the prior candidate probabilities for our analysis. The output of the classifier ($\beta(D(\cdot))$) is observed after each replacement event. The

refinement algorithm systematically exonerates or reinforces particular candidates by logically reasoning about the observed value for $\beta(D(\cdot))$ and the known topological dependency between vertices. At each step, the inference algorithm proposes a hypothesis for a set of candidates with a high probability of being the minimal candidate c^*. With each new replacement considered, the algorithm tests the previous hypothesis and updates the candidate probabilities to ultimately isolate the true minimal candidate c^* that is consistent with all available observations of $\beta(D(\cdot))$. We additionally specify that the choice for c^* must meet the confidence threshold $p(c^*) \geq 0.95$. When the confidence threshold is reached, the diagnosis process terminates successfully.

We can break down the inference approach into two operations that are repeated for each replacement event:

Op-1 Using the known relationship between elements in $\mathcal{V}_1$ (represented by $\mathcal{E}_1$), each candidate in $\mathcal{C}$ is tested for consistency against $\beta(D(\cdot))$. This is in essence a graph search problem and a detailed description of graph search algorithms used for diagnosis can be found in [101], [34], [26] and [119].

Op-2 Any previous candidates that no longer fully explain all previous observations are discarded. The GDE algorithm can manipulate these sets efficiently by only considering the smallest subsets of vertices that still qualify as candidates.

Given the observation $\beta(D(\cdot)) = k$, ($k = 1$ or 0), Op-1 divides the candidates in $\mathcal{C}$ into three categories:

1. The set of candidates that remain if $\beta(D(\cdot)) = k$, called $R^{\beta(D(\cdot))=k}$.

2. The set of candidates that are eliminated if $\beta(D(\cdot)) \neq k$, called $S^{\beta(D(\cdot))=k}$.

3. The set of candidates that cannot be eliminated irrespective of the value of $\beta(D(\cdot))$, called U.

In Op-2, the probability of each candidate in $\mathcal{C}$ is updated using Bayes' rule. Equation 6.1 shows the posterior probability for the first refinement step.

$$p\Big(c_j|\beta\big(D(\cdot)\big)=k\Big)=\frac{p\Big(\beta\big(D\left(\cdot\right)\big)=k|c_j\Big)p_0(c_j)}{p\Big(\beta\big(D\left(\cdot\right)\big)=k\Big)} \tag{6.1}$$

The equation for the posterior probability is reformulated for each category of candidates (See [34] for proof).

$$p\Big(c_j|\beta\big(D(\cdot)\big)=k\Big)=\begin{cases}0, & c_j\notin R^{\beta(D(\cdot))=k}\\ \frac{p(c_j)}{p\Big(\beta\big(D(\cdot)\big)=k\Big)}, & c_j\in S^{\beta(D(\cdot))=k}\\ \frac{p(c_j)/2}{p\Big(\beta\big(D(\cdot)\big)=k\Big)}, & c_j\in U\end{cases} \tag{6.2}$$

The binary classification for $\beta(D(\cdot))$ is a decidedly simplistic representation of all the possible differences that might exist between a model and real measurements. We restrict ourselves to a binary choice for $\beta(D(\cdot))$ to simplify notation but as shown in [34] this is not a limitation of the inference algorithm.

Complexity of the algorithm

We have found that the modeling formalism presented in Section 6.2 can be scaled to a full size semiconductor manufacturing line. The authors in [146] and [86] discuss other research efforts to represent a plant wide model using compositional semantics. Our observation about such a modeling approach, like theirs, is that modeling a plant with sufficient detail results in very large graphs. As an example, a graph representation using workflow models for a simplified 'textbook' DRAM process [158], had 134 vertices and 182 edges. Discussions with industry partners indicate that the graph representation may be two or three times as large for a practical manufacturing process [88].

The computational requirements of the inference algorithm are a limiting factor to the size of the graph that can be considered. In Op-1 of the inference process we use a graph traversal algorithm to propagate the observation through the topology to identify the conflict set. Graph traversal is $\mathcal{O}(\mathcal{V} + \mathcal{E})$.

In Op-2 of the inference process, the set $\mathcal{C}$ is manipulated and updated. As the authors in [34] show, the complexity of the refinement process is the same as a binary sort of $\mathcal{C}$ which is $\mathcal{O}(2^{\mathcal{V}})$.

Since both steps are repeated for after each replacement, the complexity of the inference process is $\mathcal{O}(2^{2|\mathcal{V}|})$. In practical terms, we find that the superlinear relationship of computational complexity to graph size limits an implementation of the algorithm to less than 100 vertices on a typical modern desktop computer.

In the following sections we will present abstraction and relaxation methods specifically suited for graphs of manufacturing process models. We will show that we are able to improve the scalability of the inference algorithm and improve the confidence margins for set c^*.

6.5 Abstraction and Relaxation methods for process graphs

As discussed in Section 6.2, we use a causal model for our representation because topological primitives such as predecessors and successors which are easily identified in a digraph are also used directly as a template for deductive reasoning.

Human diagnosticians excel at inferring and deducing the conflict set c^* at the root of a discrepancy by including in their analysis statistical correlation, hypothetical constraints and sometimes even unobservable entities in order to make observations fit the mold of a causal schema. The authors in [121] point out that human inference is rapid, sometimes at the expense of precision, due to the compulsive urge to make theories fit a causal structure. Expert diagnosticians make better choices for these "intuitive" causal constraints and therefore are able to retain the accuracy of the

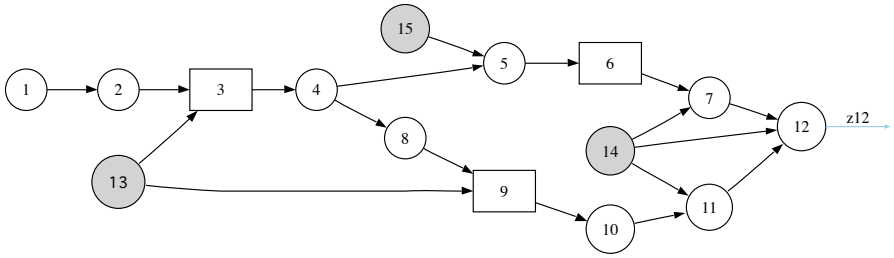

Figure 6.6: The figure shows a simple process flow shown as a directed graph. The numbered nodes represented steps in the workflow. Step 3,6 and 9 are executed by the same tool.

inferred result.

We will now introduce three simplification strategies that might be considered intuitive by a diagnostician. In a graph theoretic sense these *hypothetical constraints* enable us to either abstract or release some of the dependencies in the causal digraph. The simplifications reduce the computational complexity of the inference problem by reducing the number of vertices and edges that have to be considered for analysis.

6.5.1 Folding correlated vertices

In semiconductor manufacturing processes, etching and deposition steps may repeat several times in the workflow with different process recipes being used for each instance. In many cases the same etching or deposition tool is used for multiple instances.

Let us start with the graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ shown in Figure 6.6. The graph is structurally identical to Figure 6.5 however, vertices $3, 6$ and 9, shown as boxes, illustrate three steps in the process that use the same tool (*Tool-A*). Let the set $F^A := \{3, 6, 9\}$ represent the set of instances of *Tool-A*. Given that *Tool-A* is used multiple times in a manufacturing process, the analysis can be simplified by assuming that a yield discrepancy arising from $v_i \in F^A$ is correlated with the rest of the vertices in F^A. For our analysis we set the conditional conflict probability between elements in $F^A = 1$, i.e. $p(\beta(D(v_i))|\beta(D(v_j))) = 1 \; \forall v \in F^A$.

In essence, assuming that conflicts are correlated between instances of a tool is an example of a hypothetical constraint or an expert's intuition. In order to reintroduce this simplifying constraint into the causal schema of a directed graph, we substitute vertices in F^A with a single 'folded' vertex v_i'.

In Figure 6.7 we graphically show the resulting graph $\mathcal{G}_2$ after the vertex folding operation. The folded vertex in the figure is labeled $1'$. Note that $\beta(D(v_i')) = \bigvee \beta(D(2^{F^A}))$ i.e. $\beta(D(v_i'))$ is a disjunction of binary classifications applied to every subset of F^A.

We say that $\mathcal{G}_1$ collapses into $\mathcal{G}_2$, denoted $\mathcal{G}_1 \rhd \mathcal{G}_2$, iff there exists a homomorphism $h : \mathcal{G}_1 \to \mathcal{G}_2$, that is to say, a mapping $\mathcal{V}_1 \to \mathcal{V}_2$ such that, if $\{x, y\} \in \mathcal{E}_1$ then $\{h(x), h(y)\} \in \mathcal{E}_2$. We also specify four additional constraints on the collapse operator $\rhd$:

(1) The transformed graph $\mathcal{G}_2$ is a connected graph.

(2) Folded vertices form a vertex cut on $\mathcal{G}_2$.

(3) The small cycle constraint $|P(x, y)| < L_f$ is not violated.

(4) A planar embedding exists for $\mathcal{G}_2$ i.e. an isomorphism for $\mathcal{G}_2$ exists such that no edges cross each other except at common vertices.

We implement $\mathcal{G}_1 \rhd \mathcal{G}_2$ using a recursive fold-then-test algorithm for a given fold set $F^A \subseteq \mathcal{V}_1$ as shown in Algorithm 1.

For typical graphs, constraints (1) and (2) are rarely violated. In some cases the merging of $F^A \to v_1'$ terminates due to a violation of constraints (3) or (4). In such a case we spawn a new folded vertex v_2' and apply $h : F \to v_2'$ to the remaining elements in F^A. As a result, multiple folded vertices may be generated for a single fold set. This commonly occurs when the same tool reappears regularly throughout the process or when instances of a tool repeat after many ($> L_f$) intermediate process steps.

Algorithm 1: Vertex folding

Data: $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1), F^A, L_f$
Result: $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$
Subroutines:
$P(xy)$: Finds the longest path from x to y.
$merge(\mathcal{V}, \mathcal{E})$: Merges vertices in set $\mathcal{V}$ and produces the resulting contracted edge set $\mathcal{E}'$ using the vertex identification method in [136].
Test-1$(\mathcal{G})$: Tests if $\mathcal{G}$ is connected.
Test-2$(\mathcal{G})$: Tests if v_i' is a cut vertex for $\mathcal{G}$.
Test-3$(\mathcal{G})$: Tests if $\mathcal{G}$ satisfies the Boyer-Myrvold planarity condition [23].
Main routine:
Partially order F^A by a breadth first predecessor-successor relationship (F^A is a causal set when $\mathcal{G}_1$ is acyclic).
$1 \rightarrow i,\ \mathcal{V}_1 \rightarrow \mathcal{V}_2$
while $F^A \neq \emptyset$ **do**
  Select the first two vertices $x, y \in F^A$.
  if $|P(xy)| < L_f$ **then**
    Vertex set $V_i^A = \{x, y\}$
    $v_i', \mathcal{E}_2' = merge(V_i^A)$
    $\mathcal{V}_2' = \mathcal{V}_2 \setminus V_i^A \cup v_i'$
    $\mathcal{G}_2' = (\mathcal{V}_2', \mathcal{E}_2')$
    if *Test-1*$(\mathcal{G}_2')$ *&* *Test-2*$(\mathcal{G}_2')$ *&* *Test-3*$(\mathcal{G}_2')$ **then**
      $\mathcal{V}_2 = \mathcal{V}_2',\ \mathcal{E}_2 = \mathcal{E}_2',\ \mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$
      $F^A = F^A \setminus y$
    end
  else
    $F^A = F^A \setminus x$
    i++
  end
end

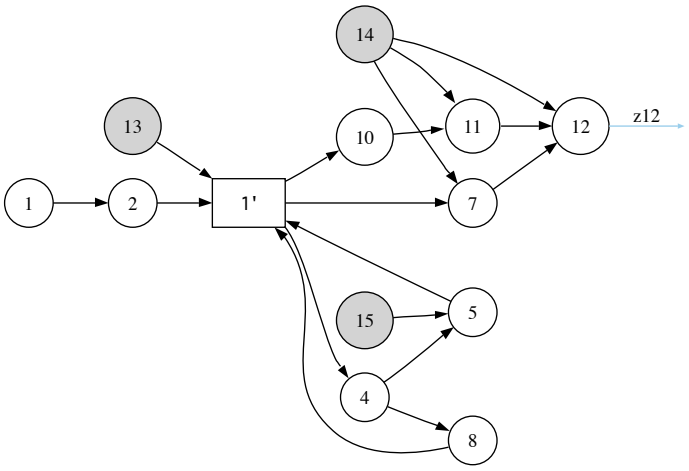

Figure 6.7: The figure shows the process workflow after instances $\{3, 6, 9\}$ of a single tool have been folded into a single vertex $1'$.

6.5.2 Clustering strongly connected vertices

Consider the output of the folding operation $\mathcal{G}_1 \triangleright \mathcal{G}_2$ shown in Figure 6.7. The folding operation introduces strongly connected components (or cycles) in $\mathcal{G}_2$ even when $\mathcal{G}_1$ is acyclic. Cycles may also be present in $\mathcal{G}_1$ prior to folding. In [122] the authors point out that when a graph contains cycles, there is no exact solution to the Bayesian inference process since some of the causal paths are circular. If an asymptotic result is obtained, then it varies based on the particular structure of the graph and the search algorithm used [109] [150]. We remove circular reasoning traps and restore exactness in the converged result of our diagnostic analysis by abstracting vertices in a cycle into a 'cluster' of co-dependent elements.

To identify clusters we use the notion of a strongly connected component (SCC) which is maximal subgraph of a directed graph such that for every pair of vertices (x, y) in the subgraph, there is a directed path from x to y and a directed path from y to x. The algorithm described in [143] is a numerically efficient method to identify SCCs in a graph. Once SCCs have been isolated, SCC subgraphs may be abstracted into 'clusters' in order to find an acyclic embedding for $\mathcal{G}_2$. The reader is directed

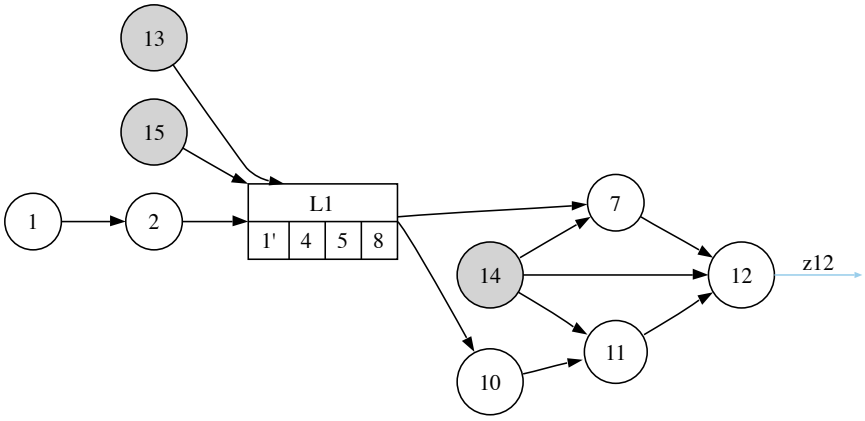

Figure 6.8: The figure illustrates the resulting graph after the set strongly connected vertices in Figure 6.7 have been abstracted into a cluster.

to the work presented in [20], [39] and [2] for tree decompositions suited to Bayesian inference over generalized directed graphs.

We use the cluster-tree elimination [35] approach on folded graph $\mathcal{G}_2$ to generate an augmented tree $\mathcal{G}_3$ whose vertices are either clusters representing SCCs or unaltered vertices from the underlying subtrees in $\mathcal{G}_2$. It is possible that the cluster-tree for $\mathcal{G}_2$ is a degenerate graph with no underlying tree structure. However, the constraints we placed on the size of re-entrant cycles in Section 6.2 and on the folding algorithm in Section 6.5.1 primarily serve to restrict the cardinality of an SCC and to retain intervening subtrees between abstracted SCCs.

Figure 6.8 is a cluster tree for Figure 6.7 showing an abstract cluster L_1 for SCC $\{1', 4, 5, 8\}$ made up of loops $\{1', 4, 5\}$ and $\{1', 4, 8\}$. Note that the cluster abstract is graphically similar to a vertex in that it inherits the inclusive disjunction $\beta(D(L_i)) = \bigvee \beta(D(2^{L_i}))$. We refrain from referring to the cluster abstract as a vertex since the elements in L_i are not aggregated when generating the candidate set $\mathcal{C}$. In other words, L_i is treated like a single vertex for Op-1 of the Bayesian diagnosis process presented in Section 6.2 but not for Op-2.

6.5.3 Relaxing latent effects

In Section 6.2, we discussed the introduction of source vertices $\{13, 14, 15\}$ to represent either process parameters or latent effects in a manufacturing process. Vertices representing deterministic process parameters are treated like regular vertices, where the forecasted value $\hat{z}_i$ for a process parameter may be a baseline or expected value. Latent effects, however, are never measurable and have no explicit model forecasts. We assume that latent effects, represented by vertices $e_i \in E$, are simply specified as a finite prior probability on classification k.

Consider for example the finite probability of wafer misalignment (which always results in a defective wafer) represented by vertex 14 in Figure 6.8, denoted by a prior probability $p_{14}(k = 1)$. Vertex 14 can never be absolved since it cannot be measured. Also, since an implicit upstream dependence of vertices is encoded into our inference problem, vertex 14 is culpable for observations from its successor vertices $7, 11$ and 12. We can, therefore, always implicate vertex 14 for discrepancies observed from its direct successors by specifying that the probability of classification k after replacing $\hat{z}_7$ with z_7 is affected by the added possibility of wafer misalignment represented by the cumulative probability $p_{14}(k) + \Big(p\big(\beta\left(D(7)\right) = k\big)\Big)$.

We also relax latent effects with multiple direct successors into independent identically distributed (IID) random variables acting parallelly on each direct successor. It is also possible for multiple latent sources to impact a single vertex or cluster as seen with vertices 13 and 15 on L_1 in Figure 6.8. In such a case we aggregate the latent sources using an opinion pool. Several pooling methods are presented in [50], but extending our IID assumption, our aggregate probability is a linear sum of all incident latent sources. These assumptions allow us to remove source vertices and outgoing edges corresponding to latent effects from $\mathcal{G}_3$, but incorporate their influence on their direct successors using *auxiliary variables* $p_{e_i}(k)$ for all vertices e_i that correspond to latent effects. $p_{e_i}(k)$ is 'attached' to each direct successor of e_i. Figure 6.9

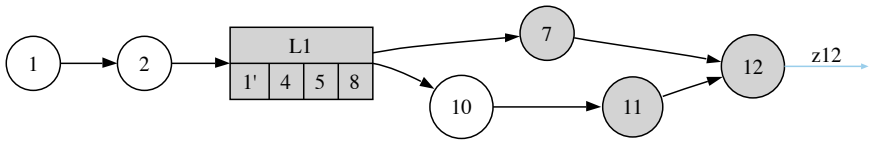

Figure 6.9: The figure shows the graph $\mathcal{G}_4$. Vertices 13,14 and 15 have been abstracted into auxiliary variables.

shows the graph $\mathcal{G}_4$ with latent sources removed. Auxiliary variables corresponding to their respective latent effects are attached to the vertices (and clusters) shaded in grey.

Auxiliary variables are included in the diagnostic analysis as follows. Note that the marginal likelihood $p\left(\beta\left(D(\cdot)\right)=k\right)$ in Equation 6.1 is a normalizing constant for the posterior probability $p(c_j|\beta(D(\cdot))=k)$. For a system with no latent effects, $p\left(\beta\left(D(\cdot)\right)=k\right)=\sum p(c_j)\ \forall c_j \in S^{\beta(D(\cdot))=k}$ since we marginalize out the conditional probability for each candidate $c_j \in S^{\beta(D(\cdot))=k}$ to get the prior probability of classification $p\left(\beta\left(D(\cdot)\right)=k\right)$. If set Aux contains all the auxiliary variables attached to candidates in $S^{\beta(D(\cdot))=k}$, then Equation 6.3 provides the normalizing constant for each new computation of posterior candidate probability.

$$p\left(\beta\left(D(\cdot)\right)=k\right)=\sum p(c_j)+\sum p_{v_i}(k) \qquad (6.3)$$
$$\forall c_j \in S^{\beta(D(\cdot))=k},\ p_{v_i} \in Aux$$

Abstractly, we can consider an auxiliary variable to be a representation of uncertainty in each diagnostic step. While latent effects are a source of uncertainty, auxiliary variables may also be used to encode measurement uncertainty at each measurement site as well as uncertainty in the classification algorithm.

The abstraction and relaxation methods presented in Sections 6.5.1, 6.5.2 and 6.5.3, applied sequentially, transform the native process graph $\mathcal{G}_1$ in Figure 6.6 with 15 ver-

tices and 18 edges to the graph $\mathcal{G}_4$ shown in Figure 6.9 with 7 vertices and 8 edges. The reduction in the size of the process graph significantly reduces the number of computations required to implement the diagnosis procedure in Section 6.4.

6.6 Strategic structure recovery

The graph simplification strategies presented in Section 6.5 reduce the computational requirements of the diagnosis process at the cost of reducing the granularity with which the probabilities of candidates in $\mathcal{C}$ are differentiated. Consider the vertex clustering strategy in Section 6.5.2, where the posterior probabilities of all candidates containing vertices that lie in a cluster are updated as a group. For a k vertex cluster, $\sum_{j=1}^{n} \frac{k(n-1)!}{(n-j)!j!}$ candidates are processed as a group preventing iterative differentiation between candidates inside the group. The auxiliary relaxation strategy in Section 6.5.3 removes edges and vertices related to latent effects to reduce the memory requirements for analysis, but also limits the posterior probability of candidates attached to auxiliary variables by modifying the marginal likelihood as shown in Equation 6.3.

The practical consequence of the simplifications applied to the process graph is that once the diagnosis process is applied there is a greater possibility that no clear choice for c^* is found. Note that $c^* = c_i$ s.t. $p(c_i) = max\{p(\mathcal{C})\}$ in keeping with the Bayesian inference strategy used for diagnosis. Recall that the diagnosis process terminates successfully only when $p(c^*) \geq 0.95$.

If a suitable c^* is found, there is a possibility that it may include a cluster vertex or a vertex with an attached auxiliary variable. To account for this possibility, we say that the diagnosis terminates successfully on a simplified graph only when $p(c^*) \geq 0.95$ and when every vertex in c^* has been fully reversed to the native state prior to simplifications. When the termination condition is not met for a simplified process graph after all replacements have been considered, a strategy is needed to

reverse abstractions and to restore relaxed constraints in order to provide sufficient granularity to identify c^*. Since reversing simplifications also increases the complexity of the analysis, we adopt a strategy of ordered iterative structure recovery to maximize the diagnostic value of each reversal.

Suppose $\mathcal{L} = \{L_1 \cdots L_n\}$ is the set of all clusters in $\mathcal{G}_4$ and $E = \{e_1 \cdots e_m\}$, $Aux = \{a_1 \cdots a_m\}$ represent the set of latent vertices and the corresponding auxiliary variables respectively. Then the net uncertainty in the group of candidates with at least one vertex in $L_i \in \mathcal{L}$ is used to rank the L_i against other clusters in $\mathcal{L}$. Similarly, the net uncertainty in a group of candidates containing at least one vertex attached to an auxiliary variable a_i is used to rank e_i against other latent sources in E. We use the information (or Shannon) entropy [49] of the candidate group as a metric of net uncertainty. $H^{L_i} = -\sum p(c_i)\log(p(c_i))$ ($\forall c_i$ with at least one vertex in L_i). Then $max\{H^{\mathcal{L}} \cup H^{\mathcal{E}}\}$ corresponds to an element L_i or e_i which is either unfolded or reintroduced into the graph to produce a new graph $\mathcal{G}_5$. The iterative structure recovery scheme is detailed in Algorithm 2. Note that a cluster reversal operation dissolves any logical disjunction used for measurements from vertices in the cluster and the reintroduction of a latent source vertex generates $\sum_{j=1}^{n} \frac{(n-1)!}{(n-j)!j!}$ new candidates. The diagnosis process is reinitialized on $\mathcal{G}_5$.

6.7 Improvements in noise tolerance

Noise, in the diagnostic context, is the net uncertainty in each inference step. The auxiliary variables used to represent latent effects are a source of noise since they represent the uncertainty in each classification $D(\cdot)$ and consequently any inference drawn from it. The introduction of auxiliary variables modifies the marginal probability of each Bayesian update as shown in Equation 6.3. By substituting the noise marginal from Equation 6.3 into Equation 6.1, we see that when the set of auxiliary variables Aux is non-empty, every update to the posterior candidate probability is

Algorithm 2: Structure recovery

Data: $\mathcal{G}_4 = (\mathcal{V}_4, \mathcal{E}_4), E, A, \mathcal{L}, \mathcal{C}$
Result: $\mathcal{G}_5 = (\mathcal{V}_5, \mathcal{E}_5)$
Initialize: $H^{L_i} = 0,\ H^{e_i} = 0\ \forall L_i \in \mathcal{L}$ and $\forall e_i \in E$
foreach $c_j \in \mathcal{C}$ **do**
  foreach $L_i \in \mathcal{L}$ **do**
    if c_j *contains vertex* v *such that* $v \in L_i$ **then**
      $H^{L_i} = H^{L_i} - p(c_j)\log(p(c_j))$
    end
  end
  foreach $e_i \in E$ **do**
    if c_j *contains vertex* v *such that* a_i *attached to* v **then**
      $H^{e_i} = H^{e_i} - p(c_j)\log(p(c_j))$
    end
  end
end
$H^{L_i}_{max} = max\{H^{L_i}\}\ \forall L_i \in \mathcal{L}$
$H^{e_i}_{max} = max\{H^{e_i}\}\ \forall e_i \in E$
if $\{H^{L_i}_{max} > H^{e_i}_{max}\}\ \wedge\ \{L_i$ *contains a folded vertex*$\}$ **then**
  Restore the internal cycle structure of L_i
  Unfold vertices in L_i
  Return: $\mathcal{G}_5 = (\mathcal{V}_5, \mathcal{E}_5)$
else
  Restore the vertex e_i and corresponding edges
  Return: $\mathcal{G}_5 = (\mathcal{V}_5, \mathcal{E}_5)$
end

reduced by a factor of ρ given by Equation 6.4.

$$\rho = \frac{\sum p(c_j)}{\sum p(c_j) + \sum p_{v_i}(k)} \tag{6.4}$$
$$\forall c_j \in S^{\beta(D(\cdot))=k},\ p_{v_i} \in Aux$$

Note that ρ penalizes the convergence rate of the diagnosis process as well as limits the maximum probability for c^*. A naive strategy to include noise or uncertainty in each Bayesian update would be to aggregate all auxiliary variables into the noise marginal to represent the net uncertainty in each classification. In other words, every auxiliary variable in the graph $\mathcal{G}_4$ is a member of set Aux.

However, as seen in Equation 6.4, the performance of the diagnosis process is improved by minimizing $|Aux|$. In our proposed approach, we minimize $|Aux|$ at each iteration of Equation 6.1 by only considering auxiliary variables attached to the candidates in set $S^{\beta(D(\cdot))=k}$ for each new substitution considered. As vertices are absolved during the diagnosis process, the number of auxiliary variables in Aux is also reduced showing incrementally increasing improvements in the convergence rate. While the cumulative improvement in the convergence rate of the proposed approach is heavily dependent on the graph structure and the number of latent sources, we observe that the improvements are most significant during the final iterations as $\rho \to 1$ allowing the diagnosis process to maximize the probability of the true candidate c^*.

6.8 Application of the method to process graphs

Our approach was applied to an example of a $35\mu m$ semiconductor fabrication process. The workflow featured 106 steps with 93 unique tools. Six latent sources were considered in the analysis, each impacting up to 5 different steps. Figure 6.10 shows a folded representation of the workflow. 8 clusters were identified in the graph

and latent sources were relaxed into auxiliary variables to yield a 40 vertex graph. The discrepancy identification algorithm was manually restricted to only consider single point replacements in order to aid the presentation of results in this section.

It was assumed that 20 steps in the process were instrumented for measurement, using which, the simplified graph was diagnosed as prescribed in Section 6.4. The computational requirements for diagnosing the simplified graph showed an $\mathcal{O}(2^{60})$ proportional reduction compared to the native graph. The true adaptation candidate c^* for our example process was known a-priori and Figure 6.11(A) shows the probability of the true adaptation candidate $p(c^*)$ as each new substitution was considered. The figure shows that $p(c^*)$ was reinforced after each iteration, however its final probability after 20 single point replacements were made did not meet our termination condition ($p(c^*) = 0.64 < 0.95$).

At this point the structure recovery algorithm was initiated and one cluster was unfolded reintroducing 8 vertices into the graph. Figure 6.11(B) shows the evolution of $p(c^*)$ after the diagnosis process is re-applied to the graph after unfolding a single cluster. The recovered structure allowed greater differentiation between candidates and $p(c^*)$ was reinforced further until it reached our successful termination condition after 15 single point replacements.

Another illustrative example was designed using the graph in Figure 6.10 to highlight the improvements in noise tolerance and convergence rate using our proposed auxiliary relaxation approach. A true candidate was manually selected and the diagnosis process was run twice. Every auxiliary variable was included in the marginal in the naive case and our proposed approach for selecting relevant auxiliary variables was used in the other case. Figure 6.12 shows a comparison of convergence trajectories for the naive case and our approach (shown as a solid line) over 20 replacement steps. The figure shows that the proposed approach has significantly improved noise tolerance evidenced by the higher probability for c^* by the end of 20 iterations. The

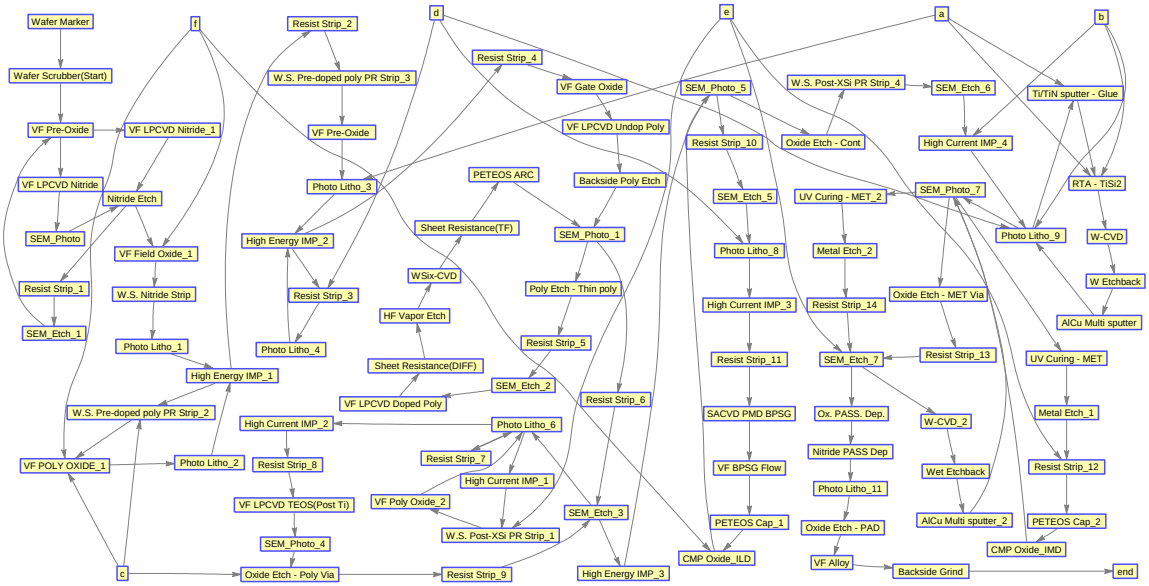

Figure 6.10: A digraph representation of a $35\mu m$ semiconductor fabrication process after vertices representing multiple instances have been folded. Vertices (a,b,c,d,e,f) correspond to latent effects.

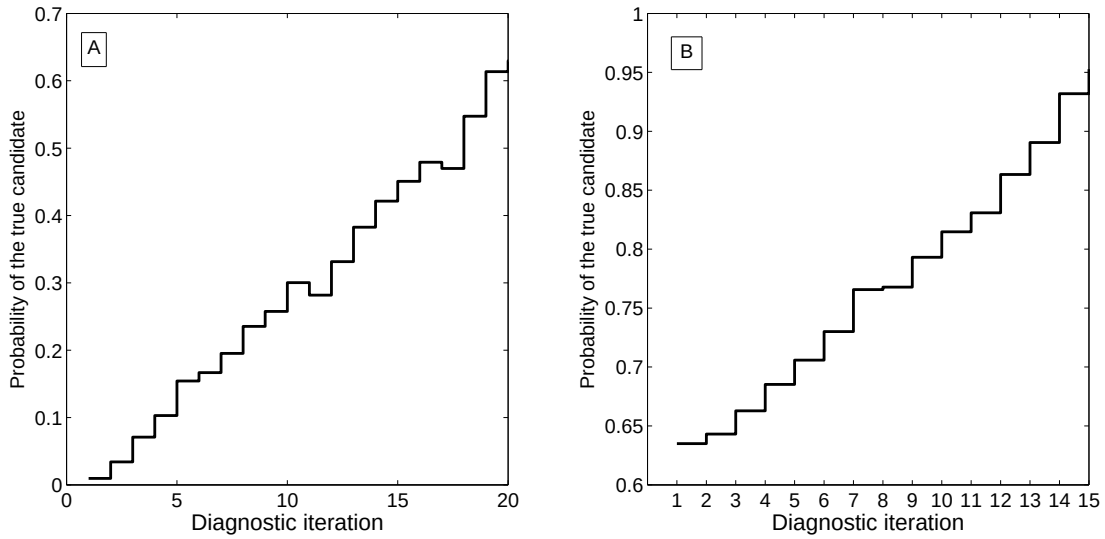

Figure 6.11: A graph showing the probability of the true candidate c^* as the diagnosis process updates its posterior probability with each new measurement replacement. Subplot-A shows that $p(c^*)$ does not reach the successful termination condition when diagnosis is applied to the simplified graph. Subplot-B shows the continuation of the diagnosis process after one cluster has been reversed using the structure recovery algorithm. The diagnosis terminates successfully after 15 iterations.

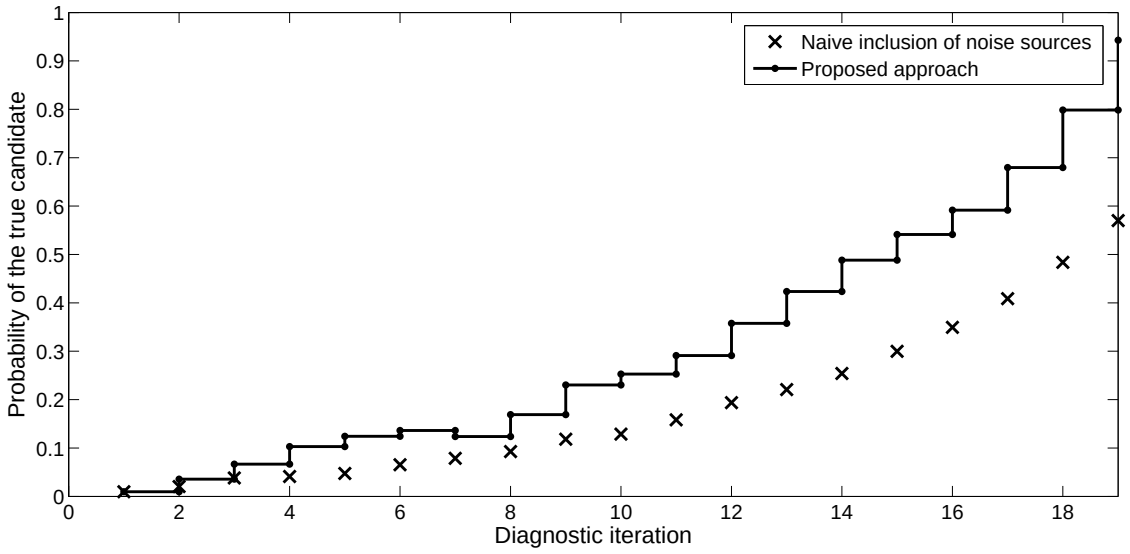

Figure 6.12: A graph showing the probability of the true candidate c^* as 20 replacements are sequentially performed. The cross markers show the trajectory for the naive inclusion of all uncertainties and the solid line shows the improved convergence when our proposed method is used.

convergence rate of the diagnosis process is also enhanced for every iteration.

6.9 Conclusion

In this chapter we present a diagnostic strategy to isolate sources of yield loss resulting from a discrepancy between measurements from a manufacturing process and its model. The manufacturing process and its model together comprise a hybrid process. Each process step is semantically defined based on its impact on the end of line yield. An explicit description of the interconnections between process steps is also provided. The model of the manufacturing workflow is therefore a semantic network. This representation of the manufacturing process enables the use of a Bayesian inference process to isolate process steps responsible for end of line yield loss. Our primary contributions in this chapter are tools and methodologies to pose a manufacturing process as a Bayesian diagnosis problem. We address specific challenges in doing so by developing a graph structure to represent the dependencies in a manufacturing

process and then address challenges of computational complexity, measurement and process noise. An itemized listing of our contributions is presented below:

1. A graph representation for manufacturing processes employing a model based prediction framework for quality control. The graph notation is used to represent both the physical structure of the manufacturing line as well as the network of dependencies between atomic model components. The graph notation is compatible with Bayesian diagnosis algorithms and includes latent effects that are not explicitly measured or modeled. The graph representation also considers structural properties of semiconductor manufacturing lines such as loops, branches and entrant processes.

2. A systematic approach to investigate the impact of individual process steps on the wafer sort yield of a given process. The substitution approach we propose is compatible with current measurement based yield calculation methods and provides a stream of discrepancy observations required for Bayesian diagnosis.

3. A set of graph simplification strategies to reduce the number of vertices and edges in the graph, making the application of the approach to larger and more realistic manufacturing lines computationally feasible. The reduction strategy includes identifying and folding multiple instances of a single machine, abstracting circular sub-graphs and relaxing the influence of disturbances and noise on the process.

4. A strategy to reverse specific simplifications one at a time when doing so improves the accuracy of the final result. The entropy contribution of each potential reversal is used as a metric for its strategic value in improving the accuracy of the final result.

5. An analysis of noise sources that reduce the rate of convergence of the diagnosis process and limit confidence in the final result. A method to reduce the impact

of noise from latent effects is presented. The method prunes the set of noise sources at every diagnostic step to only consider latent effects that directly impact probable sources of yield loss. The method is demonstrated to improve both the rate of convergence and the confidence in the final result.

CHAPTER VII

Distributed control using semantic networks and models

In this chapter we formulate a hierarchical control problem that establishes a foundation on which we can apply many of the research concepts presented in this dissertation. The results presented in this chapter are preliminary in nature and intended to highlight how semantic modeling and declarative descriptions of topology might be employed to improve the decentralized, model-based control of multiple agents. An extended abstract of this work was invited for submission in a special issue on Control Theory and Technology of the *IEEE Transactions on Smart Grid (TSG)*. The chapter also provides a discussion of future improvements and extensions to the work by using semantic networks.

7.1 Introduction

The rising number of Plug-in Electric Vehicles (PEVs) is a growing concern for the distribution utilities, as they anticipate difficulties in accommodating the additional PEV charging load [53].

Network upgrades are a trivial solution to the above issue, but would require massive investment by the distribution utilities, whose costs are ultimately passed

to consumers. The benefits of coordinated charging of PEVs to preempt overloading of the distribution system are widely discussed in academic publications. Citing improvements in cost and energy efficiency, the authors in [102] and [54] propose broadcasting incentives to all connected PEVs, in order to discourage charging during demand peaks and encourage charging when demand is low. Incentives (or penalties) may be computed using heuristic tariff rules [40], as are in use today, or by deriving a Nash-equilibrium incentive profile at the substation level, to manifest a valley-filling behavior for the aggregate PEV charging load [105], [138]. A key assumption in the substation level broadcast incentive approach is that the PEVs are price takers: their individual objectives or strategies have no significant effect on the price. The Nash-equilibrium incentive scheme lacks robustness to varying incentive response of individual consumers and non-deterministic communication delays, especially if PEV loads represent a significant fraction of overall load on distribution networks. The broadcast incentive approach, however, easily scales to a very large PEV population.

The authors in [70] implement PEV charging control at the distribution-transformer level. Their approach reconciles the conflicting objectives of the distribution utility to regulate (or restrict) the gross consumer demand on the network, and the individual objectives of each connected PEV (i.e. to charge at an acceptable rate). Their approach uses a centralized agent to compute an incentive signal so that the aggregate charging load is maintained within utility limits. Additionally, a model predictive control (MPC) scheme is used to reject load disturbances.

With potentially hundreds of connected PEVs in a single distribution network, the communication overhead involved in actively regulating the PEV population, while addressing concerns of robustness for the closed loop system, severely limits the practical scale of the MPC approach [45]. Additionally, most existing coordination strategies do not consider the commercial incentives that were already in place to discourage high peak loads among large customers, before the advent of PEV charg-

ing stations (PEVCs). Where such large customers add PEVCs to their buildings, they may reduce costs by the active regulation of the PEVC loads. It seems likely that future schemes for the coordination or control of PEV charging will be implemented at least partially through commercial incentives, similar to those used for larger customers today. The existing incentives have naturally evolved to discourage peak overload situations, so they are of interest not only as consumer incentives, but also as surrogate measures for network load constraints.

Our approach shows how the local coordination of charging among small groups of PEVCs ($\sim$4 PEVCs) could be designed to respond to existing pricing structures in such a way as to reduce the risk of overload for a much larger network, without the need for bandwidth-intensive communication with a central controller at the distribution substation. This hierarchical coordination is achieved by "grouping" PEVCs with selected building loads, so that together they can form a more constant load, and peak building loads can be mitigated by temporary reductions in charging power. Charging rates for the group are then optimized using a cost function based on existing utility tariffs for large customers, which effectively penalizes load peaks. This strategy has commercial as well as load-balancing advantages. For example when PEVCs are added to a single commercial building, with no change in tariff structure, we show how customers can charge their vehicles at much lower cost than is normally possible in residential areas.

This chapter is organized as follows. In Section 7.2 we present a motivating example of a power distribution network, consumer incentives and network constraints. The section also highlights the negative impact of uncoordinated PEV charging on the network and on consumer costs. In Section 7.3 we present our approach for hierarchical coordination of PEVCs using incentive arbitration. In Section 7.4 we present quantitative results showing improvements of our approach applied to the motivating example and summarize our contributions. Finally, we propose future extensions to

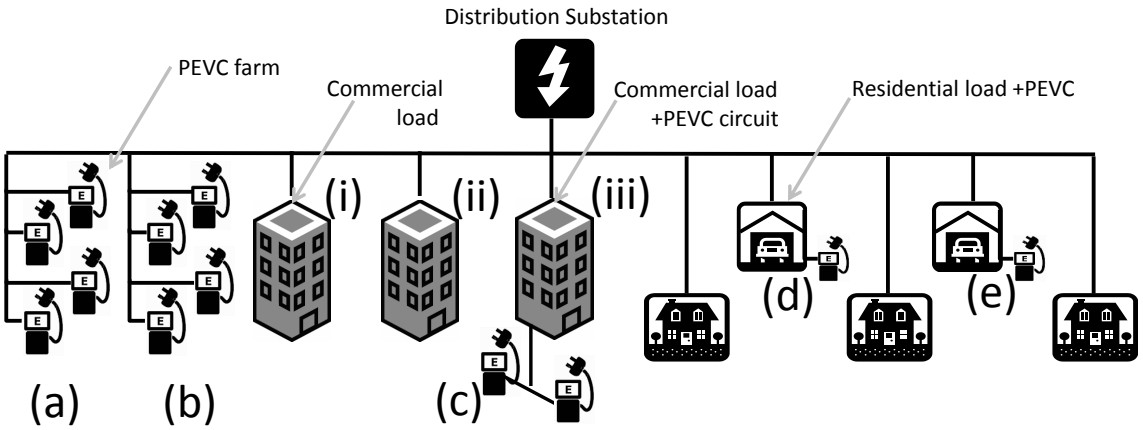

Figure 7.1: A schematic diagram of a electric distribution circuit. A single distribution transformer supplies a mixture of commercial buildings (i, ii, iii) and residential loads. The network also features twelve PEVCs on five circuits (a,b,c,d,e).

the work, utilizing topology generation and semantic modeling in Section 7.5.

7.2 Impact of PEVC loads on a distribution network: Problem Statement

The hypothetical distribution network we use for our analysis is shown in Figure 7.1. The network reflects a small district, such as a university campus or a business park, supplied by a distribution substation. A population of $\mathcal{N}$ PEVCs is connected to the network. The dynamic state of each PEVC $n \in \mathcal{N} := \{1, \ldots, \mathcal{N}\}$ at time step k is represented by the state vector $x_n[k] := [S_n[k], P_n[k], W_n[k]]^T$, ($S_n[k] \in \mathbb{R}^+$, $P_n[k] \in \mathbb{R}^+$ and $W_n[k] \in \mathbb{Z}^+$), where $0 \leq S_n[k] \leq 1$ is the normalized instantaneous state of charge of the PEV connected to n. $P_n^{min} \leq P_n[k] \leq P_n^{max}$ is the instantaneous power draw by n. W_n represents the binary operational state of PEVC n. $W_n = 1$ when n is actively charging a PEV, else $W_n = 0$. The distribution network also supports $\mathcal{M}$ commercial and residential loads, the dynamic state of each building $m \in \mathcal{M} := \{1, \ldots, \mathcal{M}\}$, at time step k, is represented by a measurement of building $L_m[k]$ kW.

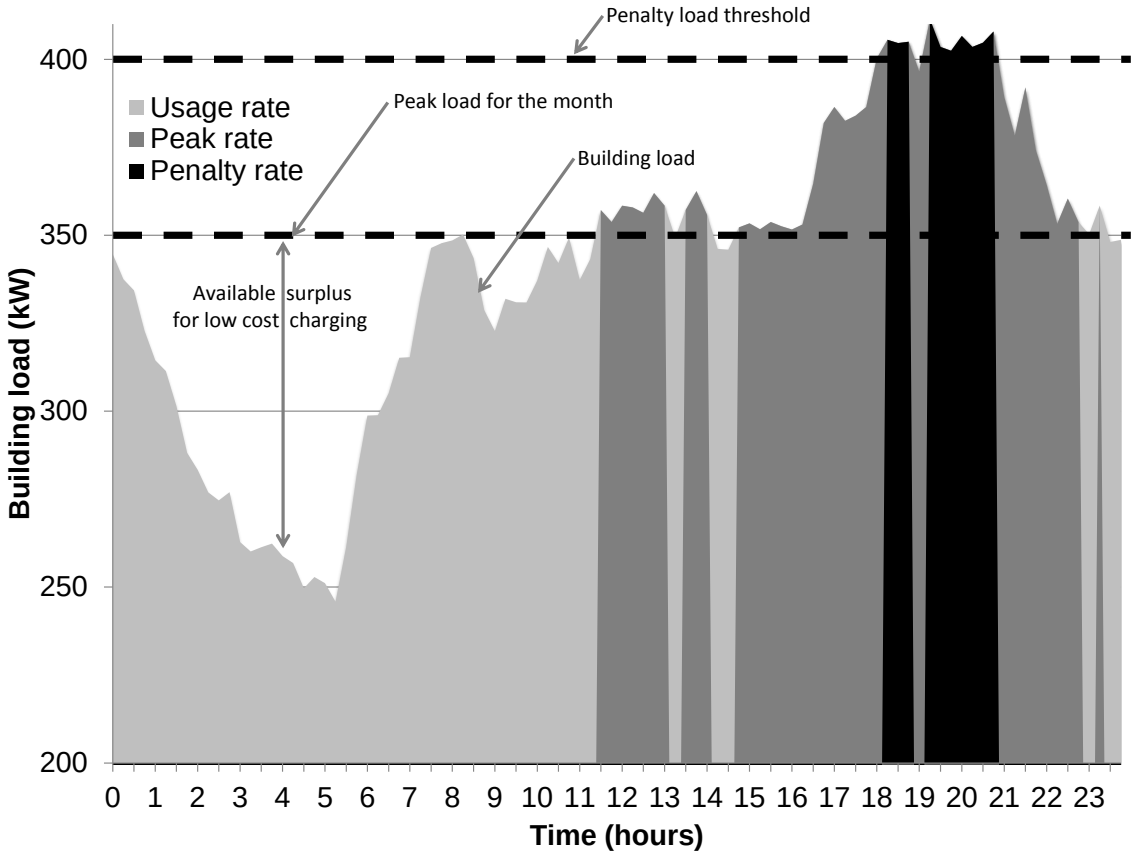

Figure 7.2: Load profile for a large residential building at the University of Michigan recorded over a 24-hour period. Under price tariffs such as D6, which are typically used for large buildings, peak loads are heavily penalized with a monthly capacity charge, proportional to the highest peak power recorded in the month. In this example, the peak power, for the month so far, is set hypothetically at 350 kW. If the day's peak load exceeds this level, then the capacity charge will be increased significantly. However, if additional loads can be supplied without exceeding the peak, then any additional PEVC load will be charged at low cost.

7.2.1 Tariffs for small and large customers

The distribution network in Figure 7.1 includes several large commercial consumers. Large consumers are usually subject to tiered pricing plans such as the Michigan-D6 tariff scheme [40]. The D6 tariff heavily penalizes large peak loads via a capacity charge, proportional to the highest single load recorded within one month, combined with a relatively low charge for energy usage. In simplified form, energy is charged at the low rate of \$0.04 per kWh, but there is an additional monthly capacity charge of \$15 per kW of the maximum power peak during the month. This forms a strong incentive to minimize load peaks. As an example, consider the load profile shown in Figure 7.2, which records the load of a large residential building at the University of Michigan over a 24 hour period, and suppose that this load is billed under the D6 tariff. If PEVC loads are added to this load at "off-peak" times (e.g 0100 to 0600 hrs), such that the monthly peak load is not increased, then the cost of PEV charging is only \$0.04 per kWh. However, if a PEVC load is added during a peak period (e.g. 1600 to 2200 hrs.), increasing the monthly peak load, then the cost for PEV charging would then be around \$0.16 per kWh, 4 times higher than the off-peak equivalent. There is therefore a heavy penalty for increasing the peak load of the building beyond whatever threshold it would otherwise have reached. For illustrative purposes, the peak load threshold in Figure 7.2 is set at 350kW.

We include the above capacity charge in our control system design in order to ensure that it acts to reduce consumer costs while also reducing peak loads. Further, large load excursions (greater than a penalty load threshold) are even more heavily penalized. In Figure 7.2 the penalty threshold is set at 400kW, purely for the purpose of illustration.

7.2.2 Electrical network constraints

We assume that a single transformer located at the distribution substation supplies power to all the loads in the network, such that the aggregate instantaneous power flowing through the transformer is given by Equation 7.1.

$$T[k] = \sum_{n \in \mathcal{N}} (W_n[k] \cdot P_n[k]) + \sum_{m \in \mathcal{M}} L_m[k] \tag{7.1}$$

$T[k] \leq T^{limit}$ represents a simplified limit constraint on the transformer. The limit constraint is a simplification of typical supply constraints used for substations [110], [59]. The assumption of a simplified limit constraint on the transformer is not a requirement of our approach but is used merely for ease of illustration. Figure 7.3 shows the transformer limit constraint and the aggregate building load in the distribution network over a typical 24-hour period. The figure highlights a key motivation for regulating the PEVC load on the circuit, i.e. the anticipated peak PEV charging hours in the late evening (1600-2200hrs.) coincide with peak building loads on the network. In the upcoming sections, we will focus our analysis on the 8-hour period between 1600hrs. and 2400hrs.

Listed below are several assumptions we make about our hypothetical distribution network:

- Tariffs such as Michigan-D6 are used for all building loads.
- Legislation is implemented so that all PEVCs exceeding 1.8kW (120V, 15A) load must be approved by the relevant utility, to ensure that they comply with communication and control standards required for coordinated load regulation.
- On rare occasions, the utility may communicate a "distress" signal to all PEVCs in an overloaded network, mandating a temporary reduction in power. However, owing to the local control mechanism described below, this action should be

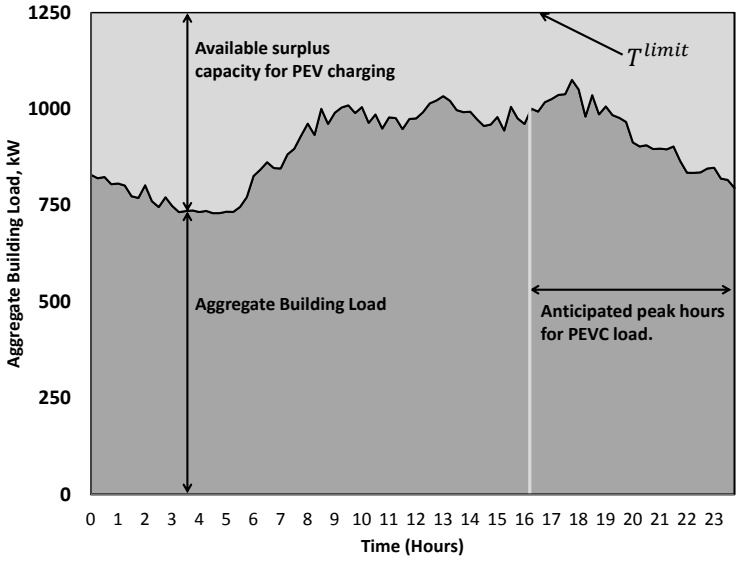

Figure 7.3: Over a 24-hour period, the aggregate building load on the distribution transformer peaks at $\sim 1000kW$, whereas the transformer limit is $\sim 1250kW$. The peak in aggregate building load coincides with anticipated peak hours for PEV charging (during the evening hours). This plot highlights the need for active regulation of PEVC loads to ensure that T^{limit} is not violated.

necessary only on rare occasions, and for this reason, we do not consider it in this study.

- All PEVCs are guaranteed at least 1.8kW (Level 1 charging rate) during their charging periods, apart from rare occurrences such as equipment failure.

- PEVs connected to active PEVCs can expect charge completion over an 8-hour period or less, and they are provided with an estimated completion time, but they are always subject to a small probability of unplanned delay due to overloaded networks, equipment failure, etc.

7.3 Incentive based charging using the proposed hierarchical implementation

Figure 7.4 shows a schematic representation of our hierarchical control implementation. By delegating control objectives and constraints to small groups of loads, we aim to reduce the communication between a load regulator at the substation and each PEVC. By delegating control functions, the scalability of closed loop load leveling is improved. The hierarchical approach also allows groups of loads to optimize local objectives, such as varying charging requirements of each PEV and specific tariff schemes for each building.

In Section 7.3.1, we first optimize charging trajectories for the entire PEVC population in the distribution network. The optimized trajectories are computed based on simple charging models for connected PEVs, and load forecasts for the non-PEV loads in the network. Then, each centrally optimized PEVC charging trajectory is assigned to a "peer group". A peer group is a logical network of peers made up of a few PEVCs and building loads. Within each peer group we implement a decentralized control methodology to address the objectives of individual PEVs and/or buildings. The decentralized controller, presented in Section 7.3.2, uses a consensus approach to address load disturbances, and utilizes object-oriented models for PEV charging dynamics and building costs.

7.3.1 Centralized charging trajectory optimization

The principal objective of coordinated charging is to abide by network constraints at all times. We formulate substation objectives as proposed in [70] and formulate the PEV charging dynamics and network constraints as a mixed integer program, to optimize the charging trajectories for all connected PEVs.

The evolution of a PEV connected to PEVC n is assumed to follow the simplified

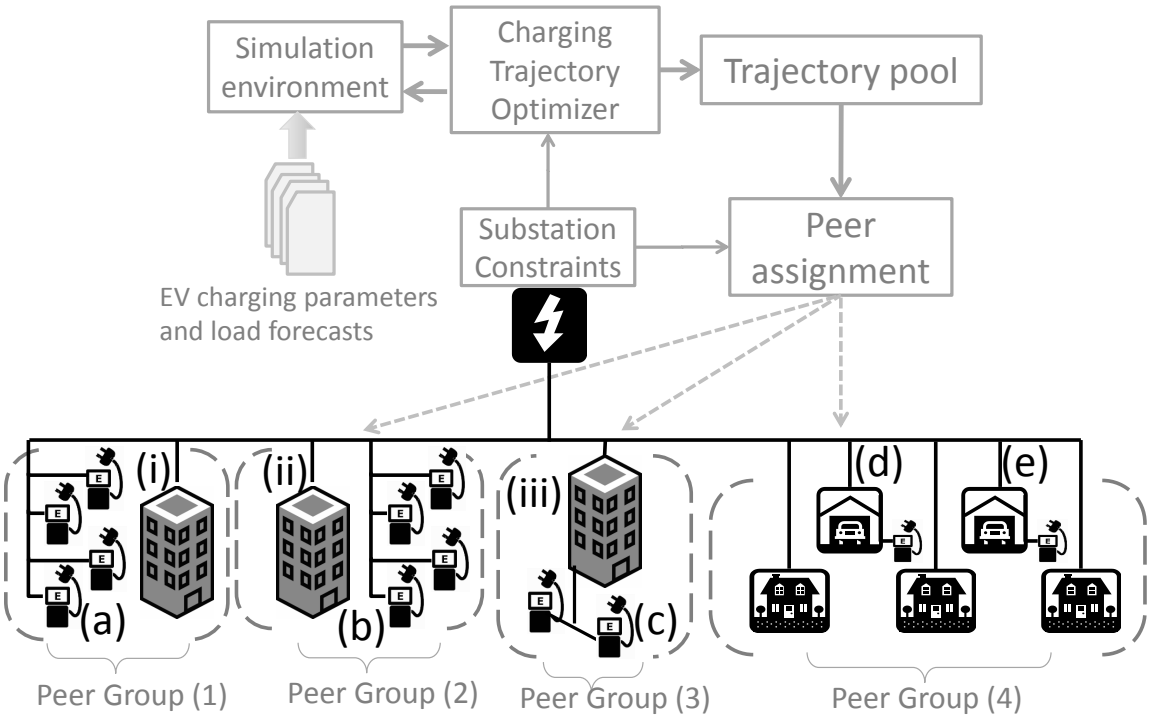

Figure 7.4: A schematic representation of the proposed control system hierarchy for the distribution network in Figure 7.1. The dashed brackets indicate peer groups. Peer groups are made up of PEVC loads together with building loads for improved load leveling within each group and at the substation level.

dynamic recursion in Equation 7.2,

$$S_n[k+1] = S_n[k] + \left[W_n[k] \frac{\eta_n \cdot T_s}{Q_n \cdot 60} \right] P_n[k] \tag{7.2}$$

where S_n is the normalized state of charge, P_n is the charging power in kW, W_n is the binary charging status of the PEVC, η_n is the charging efficiency and Q_n is the battery capacity in kWh. The indices $[k]$ and $[k+1]$ correspond to time instants (t) and $(t + T_s)$ $\forall t \in \mathbb{R}^+$. T_s is the update interval in minutes.

An optimal trajectory $P_n[1:K]$ $\forall n \in \mathcal{N}$ minimizes the cost function in Equation 7.3 and is subject to the constraints in Equations 7.4(a-e). $P_n[1:K]$ is a vector of charging loads for PEVC n over K time steps. The parameters used to optimize the distribution network in Figure 7.4 are provided in Table 7.1. Parameter values are selected based on published data for typical PEV charging characteristics [59] and electrical network constraints furnished by the University of Michigan, Utilities and Plant Engineering Department.

$$\arg \min_{P_{n \in \mathcal{N}}[1:K]} \sum_{n \in \mathcal{N}} \left[(\mathbf{1}_{[K \times 1]} - S_n[1:K])^T \mathbb{I}_{[K \times K]} (\mathbf{1}_{[K \times 1]} - S_n[1:K]) \right] \tag{7.3}$$

$$0 \leq S_n[k] \leq 1 \tag{7.4a}$$

$$P_n^{min} \leq P_n[k] \leq P_n^{max} \tag{7.4b}$$

$$W_n[k] \in \{0, 1\} \tag{7.4c}$$

$$W_n[k] \geq (1 - S_n[k]) \tag{7.4d}$$

$$T[k] \leq T^{limit} \tag{7.4e}$$

In order to meet the global supply constraint represented by Equation 7.4(e) over

Table 7.1: Parameters used to optimize PEV charging trajectories for the distribution network shown in Figure 7.4. Values for P_n^{max}, η_n and Q_n are randomly drawn from within the specified limits.

Parameter	Value
$\mathcal{N}$	12 PEVCs
T^{limit}	1.25 MW
T_s	15 minutes
K	32 (8 hours)
P_n^{min}	2 kW
P_n^{max}	$[10 - 15]$ kW
η_n	$[0.7 - 0.9]$
Q_n	$[40 - 50]$ kWh

the optimization epoch ($[1 : K]$), it is necessary to include the aggregate building load as shown in Equation 7.1.

We use the "similar days" load prediction algorithm [28] in which load histories from similar days (weekdays for weekdays, holidays for holidays etc.) are averaged to forecast building loads for the time interval $[1 : K]$. Figure 7.5 compares 24-hour load data from Figure 7.2 (dashed line) and the forecasted load (solid line) for the same day. Load forecasts for all buildings in the distribution network are added together to forecast the aggregate building load on the distribution transformer. Figure 7.6 shows the forecasted aggregate building load. The figure also demonstrates the impact of uncoordinated PEV charging on the distribution transformer.

Optimized charging trajectories for the PEVC population $\mathcal{N}$ are computed for eight hours into the future based on load forecasts. Figure 7.6 shows the predicted total load (solid black line) on the distribution transformer based on forecasted building load and optimized charging trajectories for all active PEVCs. The figure also shows that, under ideal circumstances, the transformer constraint in Equation 7.4(e) is obeyed, even under peak load conditions.

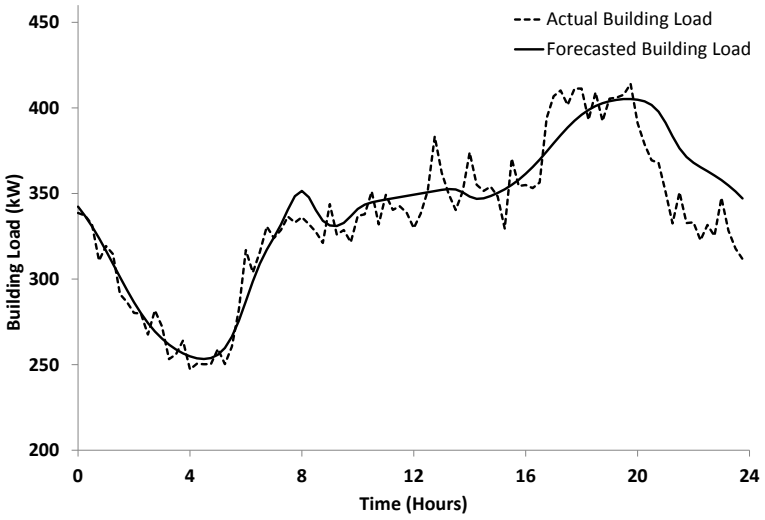

Figure 7.5: Building load data from Figure 7.2 overlaid on a *similar days* forecast for the same day.

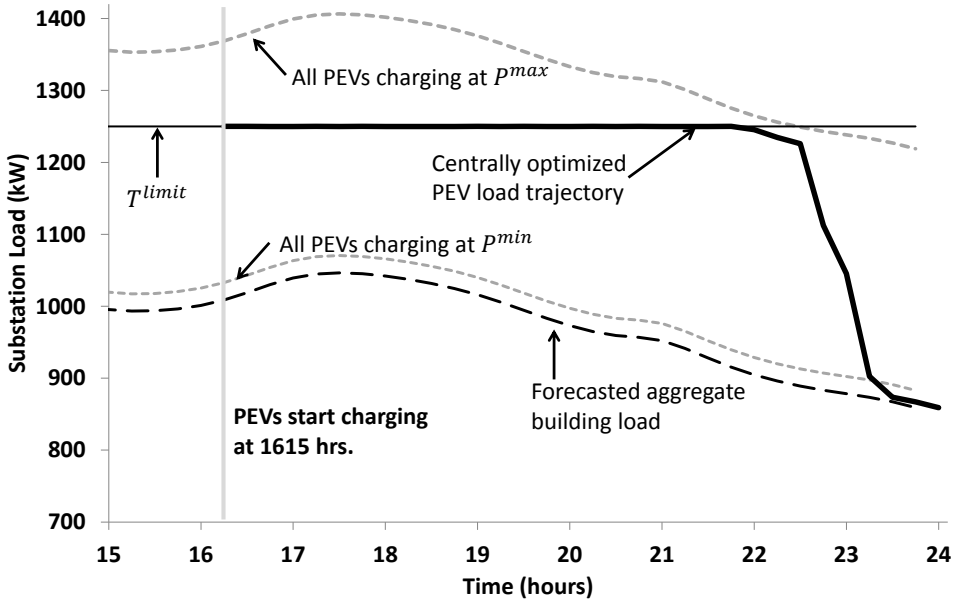

Figure 7.6: If every PEVC continually loads the network at P^{max}, the transformer limit ($T^{limit} = 1250kW$) is clearly violated. By centrally optimizing the charging load for every connected PEV, the total load on the distribution transformer is limited to T^{limit} even during the peak charging hours between 1600 and 2200 hrs.

7.3.2 Decentralized incentive arbitration

The operating conditions for a real distribution network are rarely ideal. Actual building loads differ from forecasts and all PEVCs cannot be expected to rigidly follow a centrally optimized charging trajectory. We propose a local decentralized control strategy designed to ensure that smaller groups of PEVCs and buildings reject load disturbances in order to achieve the substation goals of peak limiting and load leveling. We also show that the local controller may be additionally used to minimize the cost of energy for buildings and PEVCs while abiding by substation level constraints.

In Figure 7.1, a circuit of PEVCs (PEVC circuit (c)) is shown physically connected to the power supply of a single commercial building (iii). In this case, the total PEVC energy is metered and billed together with the building load. Building (iii) and PEVC circuit (c) can together be considered a "peer group". In Figure 7.4, PEVC circuit (c) and commercial building (iii) are together called Peer Group (3).

In cases where PEVCs are added to the existing circuit of a single commercial building, as in Peer Group (3), a local controller may be used to actively regulate PEVC load in circuit (c) to ensure that the building load for building (iii) reaches daily peaks no higher, or minimally higher, than it would without PEVCs. The marginal cost of energy for PEVC charging would then, in the limiting case, be as low as \$0.04 / kWh. By participating in a peer group, PEVs are able to benefit from the low commercial sub-peak rate for energy.

PEVC circuits (a) and (b), which are directly connected to the distribution network independently of buildings, for example in a public parking lot, may also be logically included in a peer group with buildings fed by the same substation. By assigning PEVC circuit (a) and building (i) to Peer Group (1), a local load controller for Peer Group (1) may be able to reduce the energy cost for both the building and charging PEVs.

We propose a decentralized dynamic program, implemented using a peer-to-peer

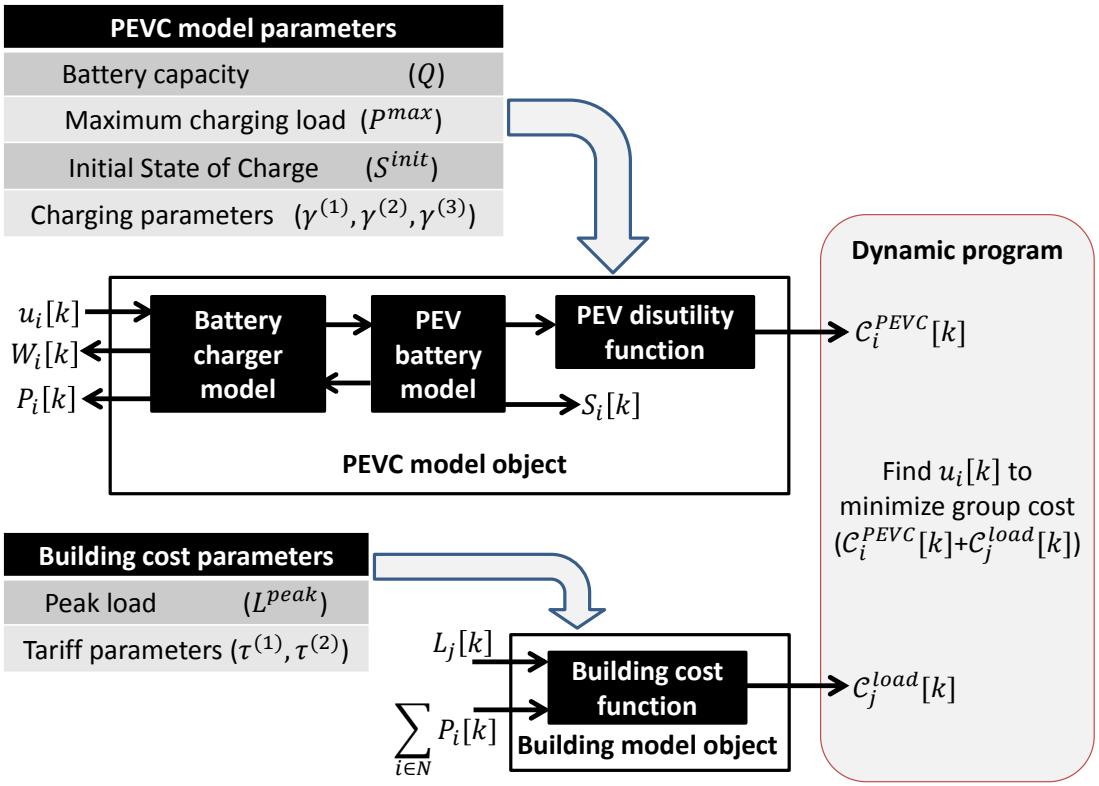

Figure 7.7: An object-oriented modeling approach used for the decentralized control within peer groups. The schematic shows generic model objects for PEVCs and building loads which are instantiated with relevant parameters when required. A dynamic program is used to find a value for $u_i[k]$ which minimizes the group cost.

consensus policy, to control PEVC loads within each peer group. Consider a peer group made up of a subset of PEVCs $N \subset \mathcal{N}$ and a subset of building loads $M \subset \mathcal{M}$. Each active PEVC ($i \in N$) is assumed to be a rational agent seeking to minimize the disutility function shown in Equation 7.5. The disutility function expresses deviations from an ideal three-stage charging profile for a Li-Ion battery [95], as a pseudo-cost $\mathcal{C}^{PEVC}$. Each building load ($j \in M$) is also an agent in the peer group and is subject to the energy costs. Equation 7.6 encodes a building's energy tariff and a quadratic penalty on peak load excursions as pseudo-cost $\mathcal{C}^{load}$. An optimal choice for charging incentive u_i at every time step k minimizes the cumulative pseudo-cost ($\mathcal{C}_G$ shown in Equation 7.7) of all the peers in a group. We use a dynamic program to determine the optimal incentive for each PEVC in the peer group, subject to the constraints in Equation 7.8(a-e).

The sum of centrally optimized trajectories $P_i[1:K]\ \forall i \in N$ is used as a constraint for the peer group, as shown in Equation 7.8(a). The aggregate load from all the locally controller peer groups, therefore, never exceeds the T^{peak}.

The hierarchical combination of global optimization at the substation level and local control at the peer group level improves the scalability of PEVC load regulation since centralized re-computation of $P_i[1:K]\ \forall i \in \mathcal{N}$ is only required when load disturbances result in a violation of the group constraint in Equation 7.8(e).

$$\mathcal{C}_i^{PEVC}[k] = \begin{cases} \gamma_i^{(1)}(P_i^{max} - u_i[k])^2 & \text{if } S_i[k] \leq 0.5 \\ \gamma_i^{(2)}(1 - S_i[k])^2 & \text{if } S_i[k] \leq 0.9 \\ \gamma_i^{(3)}(u_i[k] - P_i^{min})^2 & \text{if } S_i[k] < 1 \end{cases} \tag{7.5}$$

$$\mathcal{C}_j^{load}[k] = \tau_j^{(1)}[k]\left(\sum_{i\in N} u_i[k] + L_j[k]\right) + \frac{\tau_j^{(2)}[k]}{\left(\left(L_j^{peak} + \sum_{i\in N} P_i^{min}\right) - \left(\sum_{i\in N} u_i[k] + L_j[k]\right)\right)^2} \tag{7.6}$$

$$\arg\min_{u_{i\in N}[k]}\left[\mathcal{C}_G[k] = \left(\sum_{i\in N}\mathcal{C}_i^{PEVC}[k] + \sum_{j\in M}\mathcal{C}_j^{load}[k]\right)\right] \tag{7.7}$$

$$0 \leq S_i[k] \leq 1 \tag{7.8a}$$

$$P_i^{min} \leq u_i[k] \leq P_i^{max} \tag{7.8b}$$

$$W_i[k] \in \{0, 1\} \tag{7.8c}$$

$$W_i[k] \geq (1 - S_i[k]) \tag{7.8d}$$

$$\sum_{i\in N} u_i[k] \leq \sum_{i\in N} P_i[k] \tag{7.8e}$$

The centralized trajectory optimizer used to optimize the entire PEVC population in Section 7.3.1 uses a simplified representation of PEV dynamics (Equation 7.2).

In reality, battery charging dynamics for PEVs vary significantly between manufacturers. Further, the response of Equation 7.5 to changes in u also varies widely across the population based on design or user preferences. We capture the diversity of elements in $\mathcal{N}$ in our local control scheme by defining a 'PEVC model object' [43] for a typical PEV charger [102] which can be instantiated with relevant model parameters at the time of execution. Figure 7.7 shows a schematic representation of our object-oriented modeling approach.

We consider three design parameters: Battery capacity, initial state of charge and

maximum charging rate. Additionally, we include three parameters $(\gamma^{(1)}, \gamma^{(2)}, \gamma^{(3)})$ that represent the local policy of each PEV to the incentive signal and state of charge.

A 'Building model object' is also defined which captures the generic D6 tariff structure. The building model object is instantiated with parameters pertaining to the specific building being considered. In our case, the monthly peak threshold and two scaling parameters $(\tau^{(1)},\ \tau^{(2)})$ are necessary to assess the actual cost of energy.

We use Modelica [43] to develop model objects. Each object is compiled into a stand-alone executable program which is agnostic to execution environment and may be in parallel across multiple computers. The modularity of the object-oriented modeling approach enables a decentralized implementation of the dynamic program to optimize u, where each PEVC ($i \in N$) implements a full copy of the local controller. Each copy of the local controller includes object models and cost functions for all PEVCs and buildings in the peer group.

At every time step k, N solutions for the optimum incentive signal $u_i[k]\ \forall i \in N$ are generated. A consensus algorithm [30] is used to 'elect' an incentive signal that satisfies Equation 7.7. As discussed in [24], a peer group of agents seeking asymptotic consensus are bound by a set of topological communication constraints. An intuitive example of a topological constraint is that every agent in a peer group must be able to communicate with every other agent. In a graph theoretic sense, the peer group must be a connected graph [21]. There are several other topological conditions discussed in literature. We discuss the topological considerations for a peer group as a part of future work in Section 7.5.

7.4 Results & Conclusions

We implemented the proposed hierarchical control scheme on the distribution network shown in Figure 7.4. Figure 7.8 shows the total load on the distribution transformer. The gray solid line shows the expected total load of all the buildings and

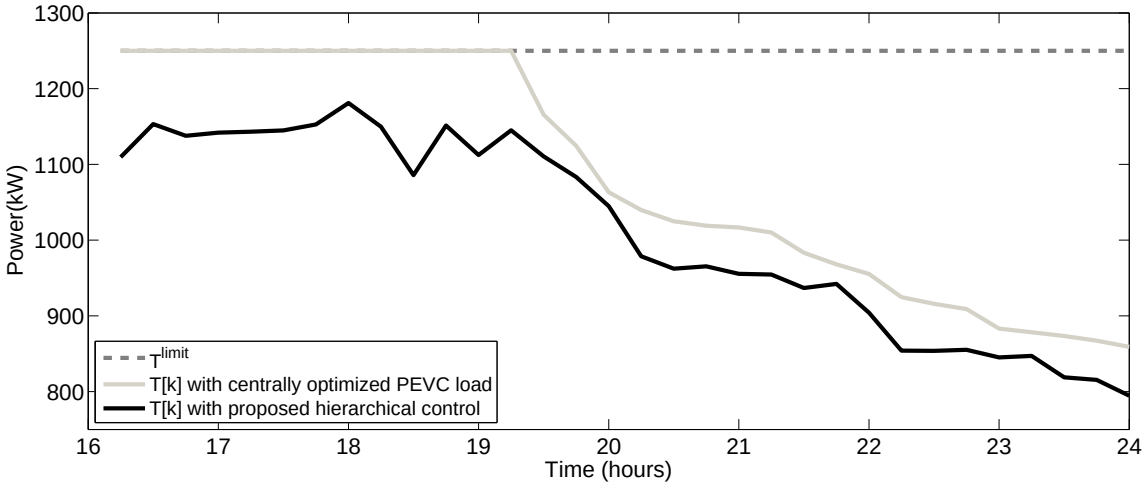

Figure 7.8: The actual load data for the building shown in Figure 7.2 together with PEVCs, overlaid on a *similar days* forecast for the building alone, as used for central trajectory optimization. The PEVC loads have been controlled so that the sum of building and PEVC loads (grey solid line) does not exceed the monthly peak threshold for the building (black dashed line).

PEVCs together if building load forecasts are accurate and PEVCs follow the centrally optimized charging trajectory perfectly. The solid black line shows the aggregate load on the substation transformer when local controllers are used for each peer group. As evidenced in the figure, our selection of parameters ($\tau^{(1)}$, $\tau^{(2)}$, $\gamma^{(1)}$, $\gamma^{(2)}$, $\gamma^{(3)}$) was fairly conservative, resulting in an aggregate transformer load that is significantly lower than its peak limit of 1250 kW (dashed grey line). In the case of uncoordinated charging, the transformer limit would be routinely exceeded.

Figure 7.9 illustrates the effect of local control on Peer Group (1). If all the PEVCs in circuit (a) charged at the rate specified by the central optimizer, the aggregate peer group load (grey dotted line) would significantly exceed the prevailing building (i) peak load threshold (solid grey, thin line). Large excursions over the peak load threshold result in heavy cost penalties for both the building and PEVs. The local controller, however, leads to a total load (solid black, thick line) that is mostly prevented from exceeding the prevailing peak load threshold.

Figure 7.10 shows individual PEVC charging trajectories for all four PEVCs in

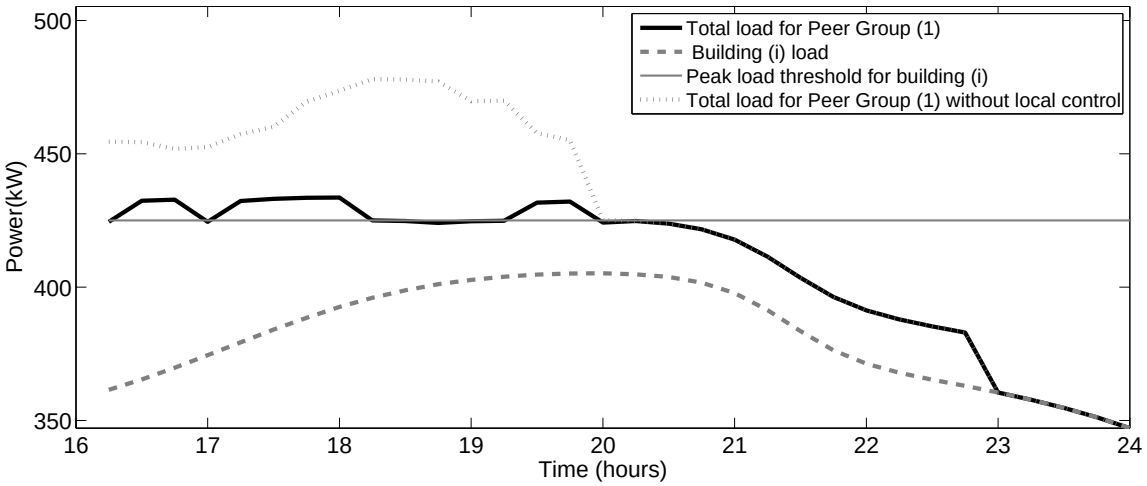

Figure 7.9: The total load for Peer Group (1). The prevailing peak load threshold for building (i) is shown at 425 kW. The lower line (gray dashed) shows the building load without PEVCs. The highest line (gray dotted) shows the group load if all PEVCs follow centrally optimized trajectories, without local control. The solid black line shows the combined load of all PEVC and building loads in Peer Group (1), when local load control is used. Excursions above the peak load threshold are mostly prevented.

Peer Group (1). The dashed grey line shows the trajectories initially determined by central optimization. The solid black line shows the actual trajectories of the four PEVCs when they are subject to local control. As seen in the figure, the four PEVCs have been dynamically re-balanced after starting, to take account of unexpected deviations from predicted levels, both in PEVCs and in the building load. The sudden drop in charging rate seen in the load profiles for PEVC 2, 3 and 4 occur when the disutility function in Equation 7.5 switches to between charging stages. While individual PEVC may be dynamically rebalanced, the constraints on the local controller ensure that the total load for group never exceeds the load scheduled by the central optimizer.

In summary, the problem of optimizing the charging load of a fleet of PEVs represents an area with growing research potential. The methods we present in this chapter address the challenges of modeling and control from the perspective of large scale implementation without altering the core optimization algorithm being used.

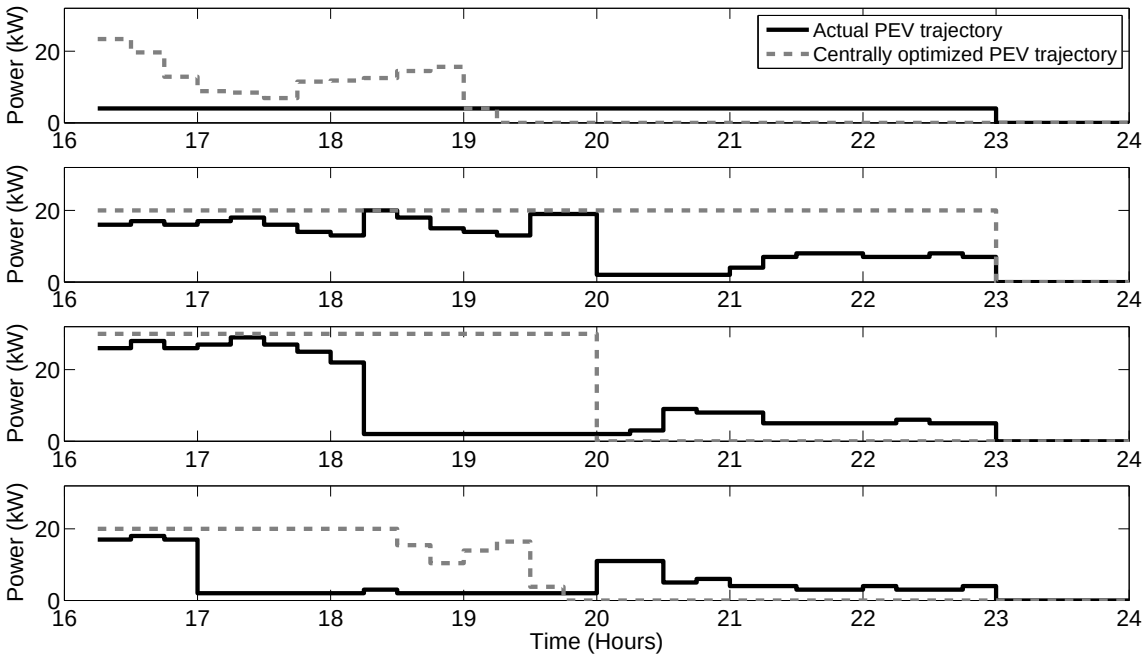

Figure 7.10: Charging trajectories for four PEVCs in circuit (a). The dashed line shows the centrally optimized trajectory for PEVCs generated using an 8-hour load forecast to ensure load leveling at the distribution transformer. The solid black line shows the locally regulated charging trajectory for PEVCs in circuit (a) when grouped into Peer Group (1).

Together they comprise an implementation framework which is agnostic to the preferred charging policy.

We have demonstrated the use of object-oriented modeling to address some of the challenges associated with the multi-agent coordination required to minimize consumer energy costs, while also reducing network overloading. We also show how existing tariff schedules, as applied to large commercial customers, are naturally compatible with our optimization approach. We demonstrated how utilities could accommodate multiple PEVC's without significant capacity increases, and simultaneously, PEV owners could access charging energy, via existing tariff schedules, at costs 3-4 times lower than peak rates.

7.5 Future work

The successful operation of our hierarchical control formulation depends on the effectiveness of each peer group at minimizing energy costs within each group while providing the desired load leveling properties at the substation level.

An effective peer group must satisfy the following objectives:

- The aggregate PEVC load in a peer group must be sufficient to maximally leverage periods of low building load.
- Each PEVC in a peer group must be able to receive accurate measurements of building load and the state of other PEVCs.
- The topology of the communication network connecting members of a peer group must facilitate the convergence of a consensus algorithm on the optimal incentive signal $u_i \ \forall i \in N$.
- The peer group must reject un-modeled behavior from other PEVCs (within limits).

This chapter has not addressed the mechanism by which we intend to assign peer groups. As a next step in our research, we intend to design an algorithm to generate peer groups in order to satisfy the listed peer group objectives.

Factors to be considered when assigning peers to groups can be broadly classified into two categories:

1) Semantic expressions for how the properties of individual peers (e.g., P^{max} and L^{peak}) affect the performance of the local controller.

2) Topological requirements for the network of interconnections between peers to ensure that an asymptotic consensus incentive can be found.

Clearly, both these factors are aspects of a semantic network. For example, the energy cost function ($\mathcal{C}^{load}$) for a building and the expression of PEV disutility ($\mathcal{C}^{PEVC}$) are both semantic definitions, and can be used to assess the the implications of introducing a new peer into a peer group. By evaluating the change in group cost in response to different potential peers and incentive inputs, we can develop a metric of robustness for a peer group depending on the aggregate rational response of each peer group to varying building loads and un-modeled PEVC behavior. An aggregate rational response is a property of a population of agents (in this case PEVCs) considered en-masse. Drawing from the domain of multi-agent or distributed cooperative systems [24], [31], we specify that every peer group must be tolerant to Byzantine, Altruistic and Rational members (BAR-T) [1] and can show that a BAR-T peer group exhibits aggregate disturbance mitigation.

A peer assignment algorithm can also be designed to ensure that the topology within each peer group meets the requirements of the consensus algorithm. As seen in Figure 7.11, each peer group may be declaratively represented as a graph of interconnections. The graph provides information about physical interconnections between buildings and PEVCs. For example, PEVC circuit (c) is physically connected to building (iii). Since one of the constraints of a consensus algorithm is that every

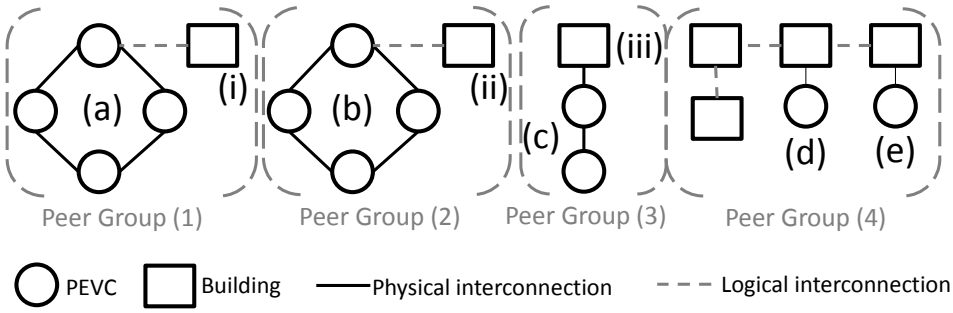

Figure 7.11: A graph representation of the network of interconnections within a peer group. Interconnections include physical connections between peers as well as logical interconnections introduced by the peer assignment algorithm.

peer in a network be able to communicate with every other peer, a peer group must be a connected graph. PEVC circuit (c) may, therefore, be grouped with building (iii) to form Peer Group (3).

In some cases, no dedicated physical communication link exists between agents, e.g. between PEVC circuit (a) and building (i). Here, we can use the logical addressing feature of the IEC-61850 protocol to programmatically create a virtual (or logical) connection between agents (when possible). The IEC-61850 protocol is a virtual networking protocol for power system substations. One of its features is the ability to construct a virtual communication hierarchy between substation devices [72]. Once a logical connection is formed, a peer-to-peer communication link may be established so that the network traffic associated with incentive arbitration is communicated over a dynamically created virtual local area network. In the case of PEVC circuit (a) and building (i), a logical connection between the two agents enables the formation of Peer Group (1) as illustrated in Figure 7.11.

As seen in this chapter, the hierarchical control of PEVC charging loads is an interesting application for semantic networks and for the analysis methods presented in previous chapters. Several other application areas and future extensions for semantic networks are discussed in Chapter VIII.

CHAPTER VIII

Conclusions and Future work

This dissertation addressed modeling, diagnosis and control problems inherent in systems that can be described as hybrid processes. Keeping these systems functioning well requires addressing discrepancies between a dynamic physical process and a corresponding model. Our research developed means to address discrepancies in hybrid processes when measurements from the physical process are noisy, bandwidth limited, delayed by a communication network or expensive to obtain. These limitations are motivated by industrial manufacturing systems and electrical power systems where the size of models and the use of digital communication networks pose significant challenges to conventional methods for model adaptation, diagnosis and control. The main contributions of this research, as well as its future work, fall into two categories:

1. Adaptation and control using semantic models.

2. Model adaption, diagnosis and control using declarative representations of model topology.

We define the term "Semantic networks" to collectively represent both these categories.

8.1 Adaptation and control using semantic models

Our contributions demonstrate the use of semantic information to adapt and construct models to achieve the performance goals of a hybrid process. We presented three use cases to showcase specific improvements.

In the first case, a networked state estimator was considered as an example of a hybrid process. Measurements from the physical process were transmitted over a packet switching network. The network introduced random delays to each communicated packet which significantly affected the performance of the state estimator. *We showed, after evaluating the delay and jitter profiles of common networks, that the accuracy of the estimated state can be improved by using precise knowledge of time at each measurement site in the network.* Using estimates of clock offset and oscillator jitter for each sensor in the network, a state estimator was designed to adapt to changing delays and jitter in the network. This work was presented in Chapter III.

In the second case, a model with variable fidelity was used in a hybrid process configuration. *Semantic information relating the choice of model order to the accuracy of the model based estimates was used to develop an automated approach to tradeoff the fidelity of the model against the measurement uncertainty associated with the physical measurements in the network.* Measurement uncertainty included sensor noise, network delays and clock uncertainty.

In the third case, a model structure adaptation approach was proposed for models that are compositionally assembled from several component models. *Combinatorial options for the modified model structure were restricted to physically feasible choices by defining a syntax based on physical compatibility between component models.* We also used a metric to rank combinatorial choices based on sensitivity of the compositional model to each proposed structural modification. The second and third cases were presented in Chapter IV.

Finally, we demonstrated the use of semantic assertions to improve the perfor-

mance of a large scale decentralized control problem for a network of agents. *Semantic assertions were used to generate small peer groups of agents with decentralized control policies, local objectives and local constraints.* As shown in Chapter VII, the decentralized controller were able to achieve local objectives while satisfying global constraints more effectively when the delegation of control authority followed the semantic guidelines.

8.2 Model adaption, diagnosis and control using declarative representations of model topology

Models for systems that span multiple physical domains or models that represent a compositional system made up of several interconnected components can be often represented as a graph.

The graph representation explicitly describes the network of interconnections of the model without capturing the purpose or behavior of the model. The graph notation however can be used in and of itself to diagnose and adapt the model when needed.

Chapters V and VI demonstrated a diagnostic methodology to isolate model components (or vertices) in a graph that may need to be adapted. The computational complexity of model adaptation is significantly reduced by first isolating only those model components that warrant adaptation. In Chapter V, the diagnostic approach was applied to an electrical circuit. We also demonstrated a structure preserving state space realization for the electrical circuit that enabled automated transformations between the state space and graph forms of the model. *Using the diagnostic tools and the structure preserving realization, a parameter identification methodology was proposed which is computationally efficient and well suited for large compositional models.*

In Chapter VI, *the vertex isolation approach was expanded to include a wider class of graphs. Specifically, graphs with loops, repeating vertices and un-modeled disturbance effects were considered.* Three graph simplification strategies were proposed to suit the application domain of a semiconductor manufacturing plant. The proposed simplification strategies included methods to collapse cycles or loops into an abstract vertex, methods to remove vertices corresponding to repeating tools, and stochastic relaxation of un-modeled disturbances.

Lastly, the topology of a communication network connecting a set of electric vehicle (EV) charging stations was considered in Chapter VII and a case study was presented demonstrating the use of a declarative model for the network topology to improve the performance of a networked control system. The topology of interconnections between interacting agents in a consensus network plays a significant role in ensuring that a peer mediated control strategy approaches stable consensus. A declarative representation of the communication network connecting a set of EV charging stations enables the expression of the state of consensus in the network as a linear dynamic system. *By inspecting the conditions under which the consensus dynamics are stable, we were able to propose an algorithm to automatically construct a network topology for a group of charging stations.*

8.3 Future work

Semantic networks are well suited to domains where a system made up of many interacting components has to be modeled as a whole. Future extensions to the work presented in this dissertation include extensions to the theory as well as broadening the approach to new application areas.

8.3.1 Extensions to the theory

There are several improvements to the theory that we would like to consider in the future. Chapters III and IV demonstrated three case studies highlighting the advantages of introducing system level semantic definitions in a model. All three case studies offer plenty of scope for improvement.

In Chapter III we proposed using clock offsets and jitter to dynamically modify the design of a networked state estimator. The state estimator used for the case study was a simple implementation of a Luenberger observer. We would like to be able to provide a set of model synthesis tools for a general class of dynamic systems so that clock synchronization moves from being a consideration purely during implementation to being one of the factors influencing early design choices and mathematical modeling.

In Chapter IV we presented a combinatorial model adaptation algorithm designed to intelligently insert additional model components to update the structure of a model. One major challenge remaining for the algorithm is to improve the scalability of the approach, to see if and how it can be applied to dozens or hundreds of model components with different fidelities and purposes interacting across a wide-area network. Modifications to the proposed method may be necessary. With the advances in memory, computing speed, and multi-core processors, there are possibilities for parallelization of the adaptation algorithm. More work can be done on deriving the sensitivity of a model's outputs to its different subcomponents with improvements to the algorithmic differentiation algorithm. In addition, the brute force search method we use to exhaustively identify all the potential candidates before pruning with semantic constraints can be significantly improved by using concepts in formal model based reasoning [26].

The parameter isolation and adaptation approach proposed in Chapter V assumes perfect data quality for all the measurements transmitted from the physical process. Including probabilistic data quality measures obtained from the NIST 61850 test net-

work [5] will allow more realistic models for the probability for false or intermittent observations. Data quality is a significant factor to consider while calculating the network utilization cost. Additionally, the analogous state-space form to the compositional model used in the chapter is a special structure preserving realization suited for models of electrical networks. A more general approach is required for application to other systems. And finally, model adaptation is not restricted to identifying the correct model parameters. In the case of a compositional model, it may also be possible to identify changes in the model structure (such as a change in the configuration of branch lines). A future improvement would be to develop a combined strategy for parameter identification and topology adaptation.

In Chapter VI we experimented with using auxiliary variables to represent uncertainty in each measurement, a more thorough analysis of the impact of uncertain process models and measurement noise will add to formally understanding the convergence properties of the diagnosis algorithm in real operating conditions. On a similar note, we assume latent effects are uncorrelated while industrial practitioners claim they frequently are. Any knowledge about correlation between random factors can be used to improve the opinion pooling of auxiliary variables. A similar strategy may be used to accommodate unknown or non-parametric distributions for latent sources.

8.3.2 New application areas

In the course of our research, we have discovered that there are several domains beyond power systems and manufacturing, where the use of semantic modeling and declarative representations of model topology can be used to improve diagnosis, model adaptation and control. Listed below are a few potential application areas to explore.

i) Design automation for complex systems: When designing complex engineered systems such as automobiles and consumer electronics it is practically impossible to imagine the full impact of early design choices on the performance of the

finished product. For critical systems such as space craft and medical devices, rigorous testing and validation is applied at every stage of the design process, at the expense of time and cost. If a semantic network was to be used to describe the design workflow then it would be possible for designers to appreciate the system level consequences of each design choice (or change). Semantic assertions that relate design choices to system performance can be inserted based on previous design efforts, allowing a designer to benefit from the collective experience of past designs for similar products.

ii) Sensor fusion: At several points in our research we have been asked by reviewers to consider the case of sensor noise and measurement uncertainty. While it is true that system level inference is sometimes sensitive to sensor noise, there is also a case to be made for using a semantic information to directly address the problem of reducing measurement uncertainty. Information about the operating limits or modes of individual sensors, for example, could be used in conjunction with classical sensor fusion techniques when the outputs from multiple sensors are fused into a single measurement. In most cases, the performance of a sensor is dependent on several parameters, many of which are co-dependent. Using a semantic definition of sensor performance would enable designers to automate sensor selection and optimize their preferred sensor fusion technique more effectively.

iii) Workflow optimization: The work we presented in Chapter VI on diagnosing semiconductor process workflows can be extended to workflow optimization in other areas. One such area that shows promise is the workflow optimization of inpatient procedures in a hospital.

A patient interned in a hospital is generally subjected to several interventions before discharge, and there is potential for significant reduction of cost and

risk to the patient if the intervention steps can be optimized to reduce time and redundancy. Any optimization procedure is limited by the fact that each patient is unique and the interventions applied are frequently updated while the patient is in the hospital. It is possible to imagine a graph representation linking cause and effect relationships, (surgery precedes post-op care, diagnostic updates causally follow lab tests, etc.), and a set of semantic definitions, (surgery times are related to the severity of a condition, lab tests have a known cycle time, etc.). Once a workflow is represented as a semantic network, it can be optimized by analyzing the impact of each intervention on the overall inpatient experience.

iv) Collaborative control: The study of collaborative control for networked multi-agent systems is a mature field of research with over a decade of results. Most research in the area is centered around consensus algorithms and perturbation studies when a set of interacting agents are modeled using algebraically coupled differential equations. With the advent of cheap high performance computing, there have been recent efforts to simulate the behavior of collaborative networked agents using model based reasoning. By modeling a network of agents as a semantic network, it is possible to study the emergent behavior of a system without explicitly solving a large system of equations. The methods presented in this dissertation can be modified to improve the computational complexity and accuracy of reasoning methods applied to collaboratively controlled muti-agent systems.

There are several other potential extensions and application areas that have come up during discussions with colleagues, reviewers and experts. Having spent five years on the work presented here, we hope that the future of *Semantic networks for hybrid processes* is both bright and exciting. Thank you for reading this dissertation.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. BAR fault tolerance for cooperative services. In *Proc. Association for Computing Machinery Symposium on Operating Systems Principles*, 2005.

[2] X. Allamigeon. Strongly connected components of directed hypergraphs. *arXiv:1112.1444*, 2011.

[3] P. Almstrom, M. Rabi, and M. Johansson. Networked state estimation over a gilbert-elliot type channel. In *Proc. IEEE Conference on Decision and Control*, 2009.

[4] J. Amelot, D.M. Anand, T. Nelson, G. Stenbakken, Y. Li-Baboud, and J. Moyne. Towards timely intelligence in the power grids. In *Proc. Precise Time and Time Interval Systems and Applications Meeting*, 2012.

[5] J. Amelot, J. Fletcher, D. Anand, C. Vasseur, Y. Li-Baboud, and J. Moyne. An ieee 1588 time synchronization testbed for assessing power distribution requirements. In *Proc.IEEE International Symposium on Precision Clock Synchronization*, 2010.

[6] D. M. Anand, D. M. Tilbury, and J. Moyne. Running simulation models in parallel with physical systems for improved estimation performance: Semantic models facilitate updating model state, parameters, and structure. In *Proc. ASME Dynamic Systems and Control Conference*, 2011.

[7] D.M. Anand, J. G. Fletcher, Y. Li-Baboud, J. Amelot, and J. Moyne. Using clock accuracy to guide model synthesis in distributed systems: An application in power grid control. In *Proc. IEEE International Symposium on Precision Clock Synchronization*, 2010.

[8] D.M. Anand, J.G. Fletcher, Y. Li-Baboud, and J. Moyne. A practical implementation of distributed system control over an asynchronous ethernet network using time stamped data. In *Proc. IEEE International Conference on Automation Science and Engineering*, 2010.

[9] D.M. Anand, J.R. Moyne, and D.M. Tilbury. Performance evaluation of wireless networks for factory automation applications. In *Proc. IEEE International Conference on Automation Science and Engineering*, 2009.

[10] D.M. Anand, D. Sharma, Y. Li-Baboud, and J. Moyne. EDA performance and clock synchronization over a wireless network: Analysis, experimentation and application to semiconductor manufacturing. In *Proc. IEEE International Symposium on Precision Clock Synchronization*, 2009.

[11] Theodore Anderson. *An Introduction to Multivariate Analysis.* John Wiley, 1958.

[12] P.J. Antsaklis. Neural networks for control systems. *IEEE Trans. Neural Networks*, 1:242–244, 1990.

[13] M. Asada. Wafer yield prediction by the mahalanobis-taguchi system. In *IEEE International Workshop on Statistical Methodology*, 2001.

[14] V.S. Asirvadam and M.J.O. Elamin. Wireless system identification for linear network. In *Proc. IEEE International Colloquium on Signal Processing and Applications*, 2009.

[15] A.V. Balakrishnan and V. Peterka. Identification in automatic control systems. *Automatica*, 5:817–829, 1969.

[16] R. Baly and H. Hajj. Wafer classification using support vector machines. *IEEE Trans. Semiconductor Manufacturing*, 25:373–383, 2012.

[17] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software.* Wiley-Interscience, 2001.

[18] M.E. Baran and F.F. Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power Delivery*, 4:1401–1407, 1989.

[19] K Behrendt and K Fodero. The perfect time: An examination of time-synchronization techniques. In *Proc. Western Protective Relay Conference*, 2006.

[20] Roderick Bloem, Harold Gabow, and Fabio Somenzi. An algorithm for strongly connected component analysis in n log n symbolic steps. In *Formal Methods in Computer-Aided Design.* Springer, 2000.

[21] B. Bollobas. *Graph theory: An introductory course.* Springer Verlag, NY, 1979.

[22] E. Bompard, R. Napoli, and Fei Xue. Analysis of structural vulnerabilities in power transmission grids. *International Journal of Critical Infrastructure Protection*, 2:5–12, 2009.

[23] John M. Boyer and Wendy J. Myrvold. On the cutting edge: Simplified O(n) planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8:241–273, 2004.

[24] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *Journal of the Association for Computing Machinery*, 32:824–840, 1985.

[25] E. Bruls. Quality and reliability impact of defect data analysis. *IEEE Trans. Semiconductor Manufacturing*, 8:121–129, 1995.

[26] Peter Bunus and Karin Lunde. Supporting model-based diagnostics with equation-based object oriented languages. In *Proc. 2nd International Workshop on Equation-Based Object-Oriented Languages and Tools*, 2008.

[27] Chengyu Cao and N. Hovakimyan. L1 adaptive controller for nonlinear systems in the presence of unmodelled dynamics: Part II. In *Proc. ACC*, 2008.

[28] Ying Chen, Peter B Luh, Che Guan, Yige Zhao, Laurent D Michel, Matthew A Coolbeth, Peter B Friedland, and Stephen J Rourke. Short-term load forecasting: Similar day-based wavelet neural networks. *IEEE Trans. Power Systems*, 25:322–330, 2010.

[29] Chen-Fu Chien, Wen-Chih Wang, and Jen-Chieh Cheng. Data mining for yield enhancement in semiconductor manufacturing and an empirical study. *Expert Systems with Applications*, 33:192–198, 2007.

[30] Han-Lim Choi, L. Brunet, and J.P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans. on Robotics*, 25:912–926, 2009.

[31] A. Clement, H. Li, J. Napper, J.-P. Martin, L. Alvisi, and M. Dahlin. BAR primer. In *Proc. IEEE International Conference on Dependable Systems and Networks*, 2008.

[32] Henry Cox. On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Trans. Automatic Control*, 9:5–12, 1964.

[33] J.A. Cunningham. The use and evaluation of yield models in integrated circuit manufacturing. *IEEE Trans. Semiconductor Manufacturing*, 3:60–71, 1990.

[34] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 1987.

[35] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[36] J. deKleer and J. Kurien. Fundamentals of model-based diagnosis. In *Proc. IFAC SafeProcess*, 2003.

[37] E. Del Castillo and A. Hurwitz. Run-to-run process control: Literature review and extensions. *Journal of Quality Technology*, 29:184–196, 1997.

[38] DHS. National power grid simulation capability: Needs and issues. Technical report, U.S. Department of Homeland Security Science and Technology Directorate, 2008.

[39] R. Diestel. Graph theory. *Graduate texts in Mathematics*, 2005.

[40] DTE Energy. DTE electric company rate book for electric service. Technical report, The Detroit Edison Company, 2013.

[41] T.F. Edgar, S.W. Butler, W.J. Campbell, C. Pfeiffer, C. Bode, S.B. Hwang, KS Balakrishnan, and J. Hahn. Automatic control in microelectronics manufacturing: Practices, challenges, and possibilities. *Automatica*, 36:1567–1603, 2000.

[42] John C. Eidson. *Measurement, Control, and Communication Using IEEE 1588*. Springer London, 2006.

[43] Hilding Elmqvist, Sven Erik Mattsson, and Martin Otter. Modelica - the new object-oriented modeling language. In *Proc. 12th European Simulation Multiconference*, 1998.

[44] EPRI. Report to NIST on the smart grid interoperability standards roadmap. Technical report, Electric Power Research Institute, 2009.

[45] C. Farmer, P. Hines, J. Dowds, and S. Blumsack. Modeling the impact of increasing phev loads on the distribution infrastructure. In *Proc. HICSS*, 2010.

[46] Alexander Feldman, Jurryt Pietersma, and Arjan van Gemund. All roads lead to fault diagnosis: Model-based reasoning with LYDIA. In *Proc. Belgium-Netherlands Conference on Artificial Intelligence*, 2006.

[47] Lihong Feng. Review of model order reduction methods for numerical simulation of nonlinear circuits. *Applied Mathematics and Computation*, 167:576–591, 2005.

[48] Ronald Aylmer Fisher and Statistiker Genetiker. *Statistical methods for research workers*, volume 14. Oliver and Boyd Edinburgh, 1970.

[49] Hans Follmer. On entropy and information gain in random fields. *Probability Theory and Related Fields*, 26:207–217, 1973.

[50] Simon French. Aggregating expert judgement. *RACSAM*, 105:181–206, 2011.

[51] Peter Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, 2004.

[52] Q. Fu, A. Solanki, L.F. Montoya, A. Nasiri, V. Bhavaraju, T. Abdallah, and D. Yu. Generation capacity design for a microgrid for measurable power quality indexes. In *Proc. IEEE Innovative Smart Grid Technologies Conference*, 2012.

[53] M.D. Galus and G. Andersson. Power system considerations of plug-in hybrid electric vehicles based on a multi energy carrier model. In *Proc. IEEE Power and Energy Systems: General Meeting*, 2009.

[54] Lingwen Gan, U. Topcu, and S. Low. Optimal decentralized protocol for electric vehicle charging. In *Proc IEEE Conference on Decision and Control*, 2011.

[55] R.M. Gardner, J. Bieker, and S. Elwell. Solving tough semiconductor manufacturing problems using data mining. In *IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop*, 2000.

[56] M. Gerdin and J. Sjoberg. Nonlinear stochastic differential-algebraic equations with application to particle filtering. In *IEEE Conference on Decision and Control*, 2006.

[57] J.D. Glover, M.S. Sarma, and T. J. Overbye. *Power Systems Analysis and Design*. CL-Engineering, 2007.

[58] Alim PC Goncalves, André R Fioravanti, and José C Geromel. Markov jump linear systems and filtering through network transmitted measurements. *Signal Processing*, 90:2842–2850, 2010.

[59] Q. Gong, S. Midlam-Mohler, V. Marano, and G. Rizzoni. Distribution of PEV charging resources to balance transformer life and customer satisfaction. In *Proc. IEEE-IEVC*, 2012.

[60] B.E. Goodlin, D.S. Boning, H.H. Sawin, and B.M. Wise. Simultaneous fault detection and classification for semiconductor manufacturing tools. *Journal of the Electrochemical Society*, 150:G778–G784, 2003.

[61] A. Goucern. Multi-domain modelling and simulation. *IEEE Review*, 45:85–87, 1999.

[62] Eric Gould. Modeling it both ways - hybrid diagnostic modeling and its application to hierarchical system designs. In *Proc. Annual Systems Readiness Technology Conference*, 2004.

[63] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. SIAM, 2008.

[64] Thomas R. Gruber. Ontolingua: A mechanism to support portable ontologies, 1992.

[65] W. Guo, B. Bai, and H.H. Sawin. Mixing-layer kinetics model for plasma etching and the cellular realization in three-dimensional profile simulator. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, 27:388–403, 2009.

[66] Vijay Gupta, Babak Hassibi, and Richard M Murray. Optimal LQG control across packet-dropping links. *Systems & Control Letters*, 56:439–446, 2007.

[67] R Haber and H Unbehauen. Structure identification of nonlinear dynamic systems- a survey on input/output approaches. *Automatica*, 26:651–677, 1990.

[68] W. Harrison, D. Tilbury, and C. Yuan. From hardware-in-the-loop to hybrid process simulation: An ontology for the implementation phase of manufacturing systems. *IEEE Transactions on Automation Science and Engineering*, 9:96–109, 2012.

[69] Q.P. He and J. Wang. Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE Trans. Semiconductor Manufacturing*, 20:345–354, 2007.

[70] R. Hermans, M. Almassalkhi, and I.A. Hiskens. Incentive-based coordinated charging control of plug-in electric vehicles at the distribution-transformer level. In *Proc. American Control Conference*, 2012.

[71] D. Hyland and D. Bernstein. The optimal projection equations for fixed-order dynamic compensation. *IEEE Trans. on Automatic Control*, 29:1034–1037, 1984.

[72] IEC TC-57. IEC 61850 communication networks and systems in substations. Technical report, International Electrotechnical Commission, 2004.

[73] IEEE Sensor Technology Committee. Standard for a precision clock synchronization protocol for networked measurement and control systems. Technical report, IEEE, 2002.

[74] IEEE Substation Committee. Standard communication delivery time performance requirements for electric power substation automation. Technical report, IEEE, 2005.

[75] IEEE WG-802.1Q. 802.1Q: Virtual LANS. Technical report, IEEE, 2006.

[76] Orhan C Imer, Serdar Yüksel, and Tamer Başar. Optimal control of LTI systems over unreliable communication links. *Automatica*, 2006.

[77] Joe-Air Jiang, Ying-Hong Lin, Jun-Zhe Yang, Tong-Ming Too, and Chih-Wen Liu. An adaptive PMU based fault detection/location technique for transmission lines. pmu implementation and performance evaluation. *IEEE Trans. Power Delivery*, 2000.

[78] Søren Johansen and Katarina Juselius. Maximum likelihood estimation and inference on cointegrationwith applications to the demand for money. *Oxford Bulletin of Economics and statistics*, 1990.

[79] Dean Karnopp, Donald L. Margolis, and Ronald C. Rosenberg. *System Dynamics: Modeling and Simulation of Mechatronic Systems*. Wiley., 2006.

[80] Dean C. Karnopp, Donald L. Margolis, and Ronald C. Rosenberg. *System Dynamics: Modeling & Simulation of Mechatronic Systems*. Wiley, 2006.

[81] Tohru Katayama. *Subspace Methods for System Identification*. Springer-Verlag London, 2005.

[82] P.A. Kawka and A.G. Alleyne. Stability and performance of packet-based feedback control over a markov channel. In *Proc. American Control Conference*, 2006.

[83] DG Kneller, FE Cohen, R Langridge, et al. Improvements in protein secondary structure prediction by an enhanced neural network. *Journal of molecular biology*, 1990.

[84] P. Korba, M. Larsson, and C. Rehtanz. Detection of oscillations in power systems using kalman filtering techniques. In *Proc. IEEE Conference on Control Applications*, 2003.

[85] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel. Reconfigurable manufacturing systems. *CIRP Annals-Manufacturing Technology*, 1999.

[86] P.R. Kristoff and D.P. Nunn. The process specification system for MMST. *IEEE Trans. on Semiconductor Manufacturing*, 1995.

[87] N. Kumar, K. Kennedy, K. Gildersleeve, R. Abelson, C. M. Mastrangelo, and D. C. Montgomery. A review of yield modelling techniques for semiconductor manufacturing. *International Journal of Production Research*, 2006.

[88] Way Kuo and Taeho Kim. An overview of manufacturing yield and reliability modeling for semiconductor products. *Proceedings of the IEEE*, 1999.

[89] R.H. Lasseter, J.H. Eto, B. Schenkman, J. Stevens, H. Vollkommer, D. Klapp, E. Linton, H. Hurtado, and J. Roy. CERTS microgrid laboratory test bed. *IEEE Transactions on Power Delivery*, 2011.

[90] D.M. Laverty, D.J. Morrow, R. Best, and P.A. Crossley. Telecommunications for smart grid: Backhaul solutions for the distribution network. In *Proc. IEEE Power and Energy and Society*, 2010.

[91] H.Y. Li and B. Yunus. Assessment of switched communication network availability for state estimation of distribution networks with generation. *IEEE Transactions on Power Delivery*, 2007.

[92] Ya-Shian Li-Baboud, X. Zhu, D.M. Anand, S. Hussaini, and J.R. Moyne. Semiconductor manufacturing equipment data acquisition simulation for timing performance analysis. In *Proc. IEEE International Symposium on Precision Clock Synchronization*, 2008.

[93] Daniel Liberzon and João P Hespanha. Stabilization of nonlinear systems with limited information feedback. *IEEE Trans. Automatic Control*, 2005.

[94] Faa-Jeng Lin, Rong-Jong Wai, and Chun-Ming Hong. Hybrid supervisory control using recurrent fuzzy neural network for tracking periodic inputs. *IEEE Trans. Neural Networks*, 2001.

[95] David Linden and Thomas B Reddy. Handbook of batteries. *McGraw-Hill*, 2002.

[96] Qiang Ling and Michael D Lemmon. Optimal dropout compensation in networked control systems. In *Proc. IEEE Conference on Decision and Control*, 2003.

[97] G.P. Liu, V. Kadirkamanathan, and S.A. Billings. Variable neural networks for adaptive control of nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 1999.

[98] G. P. Liu et al. Design and stability analysis of networked control systems with random communication time delay using the modified MPC. *International Journal of Control*, 2006.

[99] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1987.

[100] D. Luenberger. *Introduction to dynamic systems : Theory, models, and applications*. Wiley, 1979.

[101] Karin Lunde, RÅdiger Lunde, and Burkhard Munker. Model-based failure analysis with rodon. In *Proc. European Conference on Artificial Intelligence*, 2006.

[102] Zhongjing Ma, D. Callaway, and I. Hiskens. Decentralized charging control for large populations of plug-in electric vehicles: Application of the Nash certainty equivalence principle. In *Proc. IEEE Conference on Control Applications*, 2010.

[103] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proc. Association for Computing Machinery International Workshop on Wireless Sensor Networks and Applications*, 2002.

[104] K.E. Martin. Exploring the IEEE standard C37.118-2005 synchrophasors for power systems. *IEEE Trans. on Power Delivery*, 2008.

[105] A.S. Masoum, S. Deilami, P.S. Moses, M.A.S. Masoum, and A. Abu-Siada. Smart load management of plug-in electric vehicles in distribution and residential networks with charging stations for peak shaving and loss minimisation considering voltage regulation. *IET Generation, Transmission, Distribution*, 2011.

[106] T. McKelvey. SSID - MATLAB toolbox for multivariable state- space model identification. Technical report, Dept. of EE, Linkoping University, 1994.

[107] A. Milevsky and J. Walrod. Development and test of IEEE 1588 precision timing protocol for ocean observatory networks. In *OCEANS*, 2008.

[108] L. A. Montestruque and P. J. Antsaklis. On the model-based control of networked systems. *Automatica*, 2003.

[109] J.M. Mooij and H.J. Kappen. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Trans. on Information Theory*, 2007.

[110] Lia Toledo Moreira Mota and Alexandre Assis Mota. Load modeling at electric power distribution substations using dynamic load parameters estimation. *International Journal of Electrical Power & Energy Systems*, 2004.

[111] J. Moyne and B. Schulze. Yield management enhanced advanced process control system part I: Description and case study of feedback for optimized multiprocess control. *IEEE Trans. Semiconductor Manufacturing*, 23:221–235, 2010.

[112] J. R. Moyne and D. M. Tilbury. The emergence of industrial control networks for manufacturing control, diagnostics, and safety data. *Proceedings of the IEEE*, 95:29–47, 2007.

[113] James Moyne, Enrique delCastillo, and Arnon Hurwitz, editors. *Run-to-Run Control in Semiconductor Manufacturing*. CRC Press, 2001.

[114] I.R. Navarro, M. Larsson, and G. Olsson. Object-oriented modeling and simulation of power systems using modelica. In *IEEE Power Engineering Society Meeting*, 2000.

[115] T. Neagoe, V. Cristea, and L. Banica. NTP versus PTP in computer networks clock synchronization. In *Proc. IEEE International Symposium on Industrial Electronics*, 2006.

[116] Johan Nilsson, Bo Bernhardsson, and Björn Wittenmark. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34:57–64, 1998.

[117] Johan Nilsson et al. *Real-time control systems with delays*. PhD thesis, Ph. D. dissertation, Department of Automatic Control, Lund Institute of Technology, 1998.

[118] F. Pasqualetti, A. Bicchi, and F. Bullo. A graph-theoretical characterization of power network vulnerabilities. In *Proc. American Control Conference*, 2011.

[119] R.J. Patton, C.J. Lopez-Toribio, and F.J. Uppal. Artificial intelligence approaches to fault diagnosis. In *IEE Colloquium on Condition Monitoring*, 1999.

[120] Ron Patton, Paul Frank, and Robert Clark. *Issues of fault diagnosis for dynamic systems*. Springer, London, 2000.

[121] J. Pearl. *Causality: models, reasoning, and inference*. Cambridge Univiversity Press, 2000.

[122] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[123] Andreas Peikert, Josef Thoma, and Steven Brown. A rapid modeling technique for measurable improvements in factory performance. In *Proc. Winter simulation Conference*, 1998.

[124] John Pierre, Dan Trudnowski, Matt Donnelly, Ning Zhou, Francis Tuffner, and Luke Dosiek. Overview of system identification for power systems from measured responses. In *System Identification*, 2012.

[125] M. Prodanovic and T.C. Green. High-quality power generation through distributed control of a power park microgrid. *IEEE Trans. on Industrial Electronics*, 53:1471–1482, 2006.

[126] Chenkun Qi, Han-Xiong Li, Xianchao Zhao, Shaoyuan Li, and Feng Gao. Hammerstein modeling with structure identification for multi-input multi-output nonlinear industrial processes. *Industrial & Engineering Chemistry Research*, 50:11153–11169, 2011.

[127] Herbert E Rauch, CT Striebel, and F Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3:1445–1450, 2012.

[128] Mike Read, Gary Workman, and D.M. Anand. Performance specifications for industrial networks. Technical report, USCAR Plant Controllers Committee, 2009.

[129] A.D. Richards and H.H. Sawin. Atomic chlorine concentration measurements in a plasma etching reactor: A simple predictive model. *Journal of Applied Physics*, 62:799–807, 1987.

[130] Richard H Richens. Interlingual machine translation. *The Computer Journal*, 1:144–147, 1958.

[131] Martin Sachenbacher, Peter Struss, and Claes M. Carlen. A prototype for model-based on board diagnosis of automotive systems. In *AI Communications*, 2000.

[132] Borhan M Sanandaji, Tyrone L Vincent, and Michael B Wakin. Exact topology identification of large-scale interconnected dynamical systems from compressive observations. In *Proc. American Control Conference*, 2011.

[133] F.C. Schweppe and D.B. Rom. Power system static-state estimation, part i, ii, iii. *IEEE Transactions on Power Apparatus and Systems*, 1970.

[134] C. Shan, P. Tianhong, L. Zhengming, and J. Shi-Shang. Statistical key variable analysis and model-based control for improvement performance in a deep reactive ion etching process. *Journal of Semiconductors*, 6:10–25, 2012.

[135] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M.I. Jordan, and S.S. Sastry. Kalman filtering with intermittent observations. *IEEE Trans. on Automatic Control*, 49:1453–1464, 2004.

[136] Robert Duane Skaggs. *Identifying Vertices in Graphs and Digraphs*. PhD thesis, University of South Africa, 2007.

[137] S Craig Smith and Peter Seiler. Estimation with lossy measurements: jump estimators for jump systems. *IEEE Trans. Automatic Control*, 2:25–39, 2003.

[138] E. Sortomme, M.M. Hindi, S.D.J. MacPherson, and S.S. Venkata. Coordinated charging of plug-in hybrid electric vehicles to minimize distribution system losses. *IEEE Transactions on Smart Grid*, 2:198–205, 2011.

[139] John F. Sowa. *Principles of Semantic Networks*. Morgan Kaufmann, 1991.

[140] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. In *Data & Knowledge Engineering*, 1998.

[141] J.A.K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9:293–300, 1999.

[142] Koichiro Tamura, Daniel Peterson, Nicholas Peterson, Glen Stecher, Masatoshi Nei, and Sudhir Kumar. Mega5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. *Molecular biology and evolution*, 28:2731–2739, 2011.

[143] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.

[144] J.H. Taylor and B.H. Wilson. A frequency-domain model-order-deduction algorithm for nonlinear systems. In *Proc. IEEE Conference on Control Applications*, 1995.

[145] FE Thau. Observing the state of non-linear dynamic systems. *International Journal of Control*, 17:471–479, 1973.

[146] K.W. Tobin, T.P. Karnowski, and F. Lakhani. Technology considerations for future semiconductor data management systems. *Semiconductor Fabtech*, 12, 2005.

[147] Antti Valmari. Compositional state space generation. In *Advances in Petri Nets*, Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1993.

[148] P. Van Overschee and B. De Moor. *Subspace Identifcation for Linear Systems*. Dordrecht: Kluwer, 1996.

[149] Venkat Venkatasubramanian, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Computers and Chemical Engineering*, 27:293–311, 2003.

[150] Yair Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12:1–41, 2000.

[151] Bernard Widrow. Rate of adaptation in control systems. *ARS Journal*, 32:1378–1385, 1962.

[152] B. H. Wilson and J. L. Stein. An algorithm for obtaining proper models of distributed and discrete systems. *Journal of Dynamic Systems, Measurement, and Control*, 117:534–540, 1995.

[153] B. H. Wilson and J.H. Taylor. A frequency domain model-order-deduction algorithm for linear systems. *ASME Journal of Dynamic Systems, Measurement, and Control*, 120:185–192, 1998.

[154] R. E. Wilson. PMUs. *IEEE Potentials*, 13:26–28, 1994.

[155] Dapeng Wu and R. Negi. Effective capacity: a wireless link model for support of quality of service. *IEEE Trans. Wireless Communications*, 2:630–643, 2003.

[156] Ge Xia and Yong Zhang. On the small cycle transversal of planar graphs. *Theoretical Computer Science*, 412:3501–3509, 2011.

[157] Yonggang Xu and Joao Pedro Hespanha. Estimation under uncontrolled and controlled communications in networked control systems. In *In proc.IEEE Conference on Decision and Control*, 2005.

[158] M. Yamawaki. Embedded dram process technology. In *Symposium on Semiconductors and Integrated Circuits Technology*, 1998.

[159] Fuwen Yang, Zidong Wang, YS Hung, and Mahbub Gani. H-infinity control for networked systems with random communication delays. *IEEE Trans. Automatic Control*, 51:511–518, 2006.

[160] J.K. Yook, D.M. Tilbury, and N.R. Soparkar. A design methodology for distributed control systems to optimize performance in the presence of time delays. In *Proc. American Control Conference*, 2000.

[161] J.K. Yook, D.M. Tilbury, and N.R. Soparkar. Trading computation for bandwidth: reducing communication in distributed control systems using state estimators. *IEEE Trans. on Control Systems Technology*, 10:503–518, 2002.

[162] Wenwu Yu and Jinde Cao. Adaptive QS time-varying synchronization and parameters identification of uncertain delayed neural networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 16, 2006.

[163] K. Zhang, I.W. Tsang, and J.T. Kwok. Maximum margin clustering made practical. *IEEE Trans. Neural Networks*, 20:583–596, 2009.