FAULT DETECTION AND DIAGNOSIS METHODS FOR RESIDENTIAL AIR CONDITIONING SYSTEMS USING CLOUD-BASED DATA

A Dissertation

by

AUSTIN PAUL ROGERS

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2019

Major Subject: Mechanical Engineering

ABSTRACT

Buildings account for nearly 40% of total energy consumption and nearly 75% of electrical energy consumption in the United States, and a significant portion of this energy consumption is due to the heating and cooling systems. Both commercial and residential heating, ventilation, and air conditioning (HVAC) systems are prone to faults that degrade performance and increase energy consumption. Furthermore, these systems are robust to faults in that they will operate with faults present for an extended period of time and will often continue to maintain a comfortable indoor environment. While considerable work has been devoted to developing fault detection and diagnosis (FDD) strategies for large and small commercial systems, relatively little has been done specifically for residential systems.

This research presents novel FDD methods developed specifically for residential air conditioning systems. By using a novel set of virtual sensing methods, the proposed methodology eliminates the need for installing sensors on the outdoor unit. This is a significant advantage for residential 'split' air conditioning systems because installing sensors on both the indoor and outdoor units increases the complexity and cost of the data acquisition system.

In addition to the proposed set of virtual sensors, this research provides solutions to two other problems that arise when implementing FDD methods on field-operating systems. (1) While most FDD methods use static models and rely on steady state analysis, field-operating systems often will not achieve steady state operation. This research provides a method for predicting the equilibrium operating point for many air conditioning parameters while the system is still in a transient response. This enables the equilibrium point to be determined before steady state operation has been achieved, and thus a static analysis may be performed without the system reaching steady state. (2) Existing change-point detection methods that could be used for *detecting* faults are impractical to implement on a large scale because they may require *a priori* knowledge, extensive tuning, or high computational loads. This research proposes a change-point detection algorithm for the purpose of fault detection which requires minimal assumptions, tuning, and computation. This change-point detection algorithm is suitable for deployment across many different systems simultaneously.

In addition to the solutions outlined above for performing FDD using installed sensors, this research also proposes methods for performing fault detection and diagnose using only thermostat data. While a full strategy for thermostat data is not presented, crucial preprocessing methods that more complete methods will be built on are presented in detail. Nearly all of the data analyzed for each method described in this study uses event-based data uploaded in real-time to a cloud-based database and then queried and analyzed to perform FDD.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Dr. Bryan Rasmussen (advisor), Dr. David Claridge, and Dr. Prabhakar Pagilla of the Department of Mechanical Engineering, and Dr. Xia Ben Hu of the Department of Computer Science & Engineering.

Funding Sources

Graduate study was supported by the William E. Dark '54 Graduate Fellowship from the Department of Mechanical Engineering at Texas A&M University, and by Trane, a subsidiary of Ingersoll Rand. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Trane.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Background and Motivation

Buildings accounted for 39% of total energy consumption and 74% of electrical energy consumption in the United States in 2017 [1]. Approximately 48% of the total energy consumption and 32% of the electrical energy consumption is due to Heating, Ventilation, and Air Conditioning (HVAC) systems [2, 3] (see Table 1.1 and Figure 1.1). However, these systems (especially vapor compression-based systems) are prone to faults that may go unnoticed by the building occupants and operators for an extended period of time and that result in a substantial reduction in energy efficiency. Automated Fault Detection and Diagnosis (FDD) for HVAC systems provide a means to maintain the design efficiency over time as faults are detected, diagnosed, and corrected with little delay.

The majority of FDD work related to building systems is developed for application to large commercial HVAC systems [4, 5, 6] including central chillers, cooling towers, and air-handling units (AHUs). A smaller portion of work has been dedicated to smaller commercial systems

Table 1.1: An Overview of Electrical and Total Energy Consumption (trillion Btu) in the U.S. Residential and Commercial Sectors.

	Electricity Consumption[*]						Total Energy Consumption[*]					
	Residential		Commercial		Combined		Residential		Commercial		Combined	
Heating[†]	638	14.8%	85	2.0%	723	8.4%	3945	43.2%	1756	25.2%	5701	35.5%
Cooling[†]	730	16.9%	633	14.9%	1363	15.9%	731	8.0%	656	9.4%	1387	8.6%
Ventilation[†]			668	15.8%	668	7.8%			668	9.6%	668	4.2%
Total HVAC[†]	1368	31.6%	1386	32.7%	2754	32.2%	4676	51.3%	3080	44.2%	7756	48.2%
All End-Uses[‡]	4323		4241		8564		9114		6963		16077	

[*] Residential data is from the EIA 2015 Residential Energy Consumption Survey [3]. Commercial data is from the EIA 2012 Commercial Buildings Energy Consumption Survey [2].
[†] All percentages are given with respect to the energy consumption for All End-Uses (the last row).
[‡] The energy consumption for All End-Uses includes HVAC, lighting, water heating, refrigeration, etc. and represents the *site* energy consumption: electricity retail sales and primary energy consumption. Electrical system energy losses are not included.

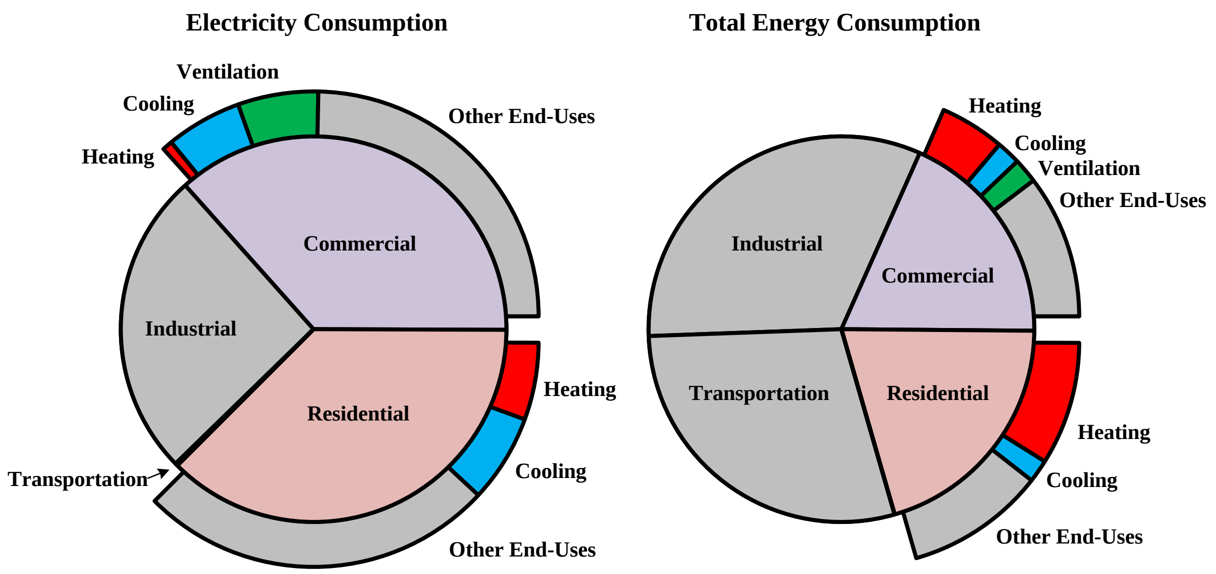


Figure 1.1: A break down of U.S. total energy and electrical energy consumption by sector (commercial, residential, industrial, and transportation) [1], and a further breakdown of consumption within the residential and commercial sectors [2],[3].

including packaged air conditioners or roof-top units (RTUs). Relatively little work has been developed for residential systems that are largely comprised of centralized split air conditioning systems. This may be surprising considering that the residential sector consumes more energy than the commercial sector and that the energy dedicated specifically to cooling is greater in the residential sector (Table 1.1 and Figure 1.1). However, the energy consumption in the commercial sector is spread across significantly fewer systems such that the energy consumption *per system* is greater in the commercial sector; there are about 5.6 million commercial buildings [2] and 118.2 million residential housing units [3] in the United States. Advanced features such as FDD are therefore more cost effective in the commercial sector because the HVAC systems are larger, have a higher up-front cost, and have higher operating costs.

A more comprehensive view of the benefits provided by FDD is necessary in order to justify the costs in the residential sector. Chapter 3 in this dissertation presents a literature review of FDD methods applicable to residential air conditioning systems and highlights several benefits that are not commonly considered. While the reduction in energy consumption and maintenance costs has been studied [7, 8, 9], other benefits such as the improved occupant comfort and the reduction in peak electrical demand have not been studied extensively. To more fully capture the motivation behind FDD for residential air conditioning systems, the following three sections review studies on service records that capture prevalence of faults and the impact that these faults have on system performance (Section 1.1.1), review the potential reduction in operating costs and service costs (Section 1.1.2), and present the impact that faulty air conditioning operation has on occupant comfort (Section 1.1.3).

1.1.1 The Prevalence and Impact of Faults

Approximately 20 years ago, several studies [10, 11, 12] were published on the prevalence and impact of faulty air conditioning behavior using data obtained from many field-operating systems. While parts of these studies may be outdated (e.g. the results regarding over-sizing may be outdated because sizing standards have changed), they are still of great value and many results are still relevant in today's market. Fault detection and diagnosis (FDD) methods have not changed significantly over the past 20 years in the residential market, and therefore the findings regarding airflow and charge faults will still be relevant today. While more recent studies have measured the impact of faults in a controlled laboratory setting [13, 14, 15, 16], there is great value in obtaining the prevalence and impact of faults on systems operating in the field.

The field results revealed that, on average:

- Residential air conditioners operate at least 17% below their design performance [11].
- 68% of residential systems required repair [12].
- The most common faults are related to refrigerant charge and airflow.
- 32% of residential systems had greater than 7.5% overcharge [12].
- 21% of systems have inadequate airflow [12].
- Approximately 40% had greater than 17% undercharge [12].

- A 16% increase in system efficiency is possible by correcting only charge and airflow faults [10].

1.1.2 Reducing Operating and Service Costs

Considering the substantial amount of energy consumed by air conditioning systems in the United States (Table 1.1 and Figure 1.1) and the prevalence of faults (Section 1.1.1), one would expect that significant energy savings are possible by implementing FDD methods. Indeed, 17% of the energy dedicated to cooling in the residential sector [11] were saved is equivalent to 5.4% of the electrical energy consumption in the residential sector and 1.1% of electrical energy consumption in the United States. Considering the large scale reduction in energy provides a very compelling argument for implementing FDD. However, the cost effectiveness must be considered.

The authors of [7, 8, 9] consider the potential cost savings that FDD provides with respect to commercial rooftop units. In analyzing the service cost savings, the authors include factors such as the savings associated with preventing a compressor failure, savings associated with eliminating unnecessary service (e.g. replacing filters too often), and savings associated the reducing the time required to diagnose a fault. The authors estimate that implementing an FDD system will save \$108 per ton per year of the baseline \$147 per ton per year: a saving in service costs of 73.5%. However, this study does not generalize well to the residential market. For example, a maintenance plan involving regular preventative maintenance will be far less common for residential systems. For many homeowners and occupants, regular maintenance involves no more than replacing the indoor air filter every 3 or 4 months. While FDD will reduce service costs for residential systems, these savings will generally only be seen years later when an emergency situation arises. Soft faults that reduce cooling capacity and efficiency by 15-20% are unlikely to be serviced and will therefore only result in an increase in operating costs until a hard fault occurs.

The authors of [8] observed a wide range (\$16 to \$180 per ton per year) in potential operating cost savings depending on the type of commercial building and the location. To put this in context, the EIA reports that residential air conditioning accounts for 730 trillion Btu across 118.2 million households in the residential market: an average of 6.18 MBtu or 1810 kWh per year [3]. At

$0.12/kW, the average household spends only $217 annually on air conditioning. Given that the average home is about 2,000 square feet [3], the average air conditioner size is likely 3 - 3.5 tons, and the average operating costs are between $62 and $72 per ton per year. This indicates that the average operating cost savings that air conditioning FDD provides in the residential market is likely closer to the lower estimate that [8] observed. Of course, this conclusion has only been obtained using inexact estimates, and the potential cost savings will be much higher in warmer climates.

This section has explored a study of the potential cost savings that FDD provides for smaller commercial units and has discussed how the cost savings in the residential market compare with the estimates provided in the study. This section has concluded that there are significantly fewer potential cost savings in the residential market and therefore financial considerations on their own are unlikely to justify the cost of an FDD system. The following section explores the improvement in occupant comfort that air conditioning FDD may provide, thus providing further motivation for the development of these tools.

1.1.3 Improving Occupant Comfort

A residential homeowner faced with the decision of whether or not to pay for the additional costs associated with an FDD system is unlikely to see the same reduction in service costs and operating costs reported in [7, 8, 9]. However, there are other important benefits provided by FDD that are not typically captured in the published literature. The first and foremost of these is the improved occupant comfort that FDD provides. Without an FDD system, faults are most likely to be discovered during the most extreme days of the year; the presence of the fault will result in uncomfortable indoor conditions on these extreme days. With effective FDD methods in place, degraded performance may be detected and diagnosed before it is noticed by the occupant. For many homeowners and occupants, the potential for more reliable occupant will be a more compelling argument that the potential cost savings.

Previous studies using field-installed systems [10, 11, 12] reported that faults are common in the residential sector such that residential systems operated about 17% below their design performance on average. However, the impact that this performance reduction has on occupant comfort

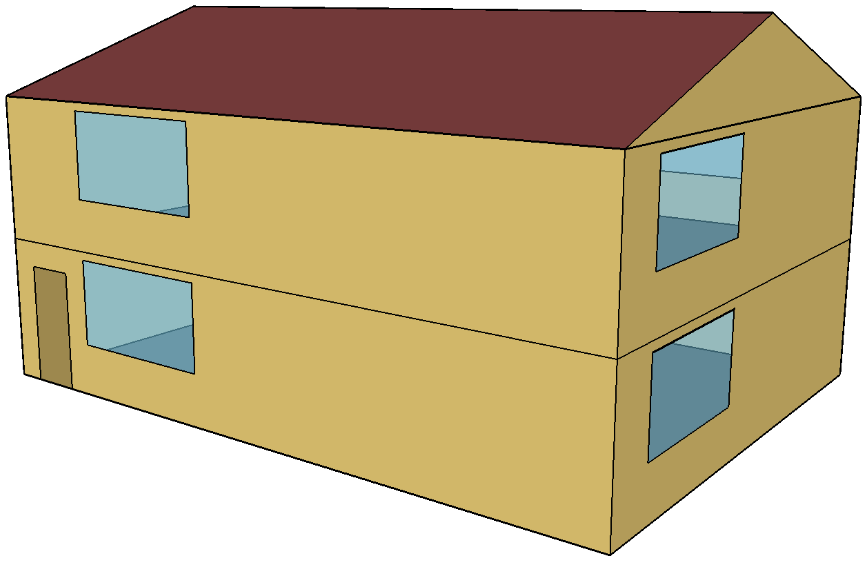


Figure 1.2: A depiction of the EnergyPlus residential single family reference model developed by [17] in order to analyze the 2012 International Energy Conservation Code (EICC). This model is used here to quantify the impact of performance degradation on occupant comfort.

must be quantified. In order to quantify this impact, residential reference EnergyPlus models [17] that adhere to the 2012 International Energy Conservation Code (IECC) are used. EnergyPlus is used to size the HVAC systems appropriately for a single-family home (Figure 1.2) in a certain location, and then a sizing factor is used to reduce the system size below the nominal value thus simulating degraded performance.

Figures 1.3 and 1.4 show the impact that degraded air conditioning performance has on occupant comfort in the cities of Houston, TX and Phoenix, AZ respectively. The occupant comfort is quantified in terms of two metrics: (1) the maximum indoor temperature and (2) the number of hours during which the indoor temperature is greater than 0.5°F above the setpoint. Even with the nominal cooling capacity (no degradation), a home in Houston will be reach approximately 3°F above the setpoint; the home nearly reaches 78°F when the setpoint is 75°F and nearly reaches 80°F when the setpoint is 77°F. However, the Houston home exceeds the setpoint for only a few hours during the year when the nominal cooling capacity is used. The conditions in Phoenix are more extreme, and the home reaches nearly 4°F with the nominal cooling capacity and spends

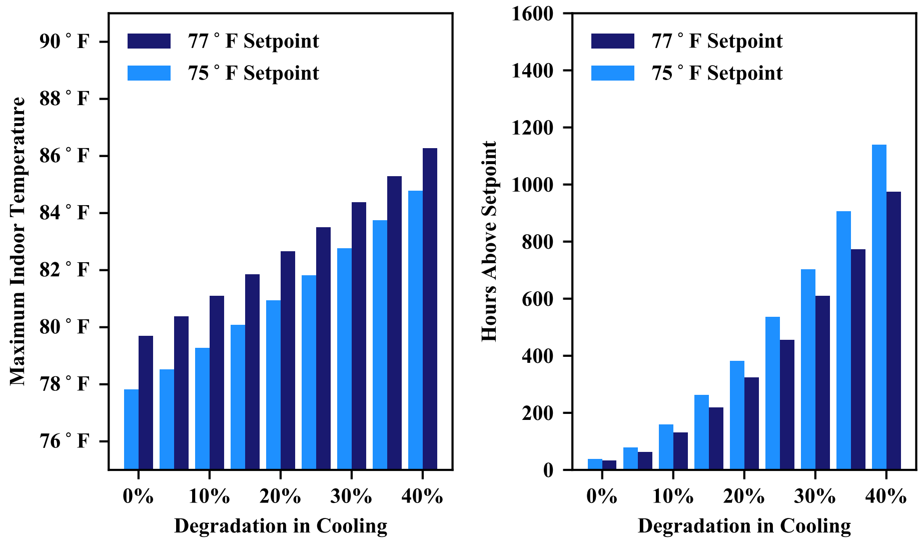


Figure 1.3: An analysis of the impact that degraded performance has on occupant comfort in Houston, TX. A residential single family EnergyPlus model [17] with a slab-on-grade foundation is used with varying sizing factors. Two metrics are used to evaluate the occupant comfort: (1) the maximum annual indoor temperature and (2) the number of hours annually that the indoor temperature is greater than 0.5°F above the setpoint.

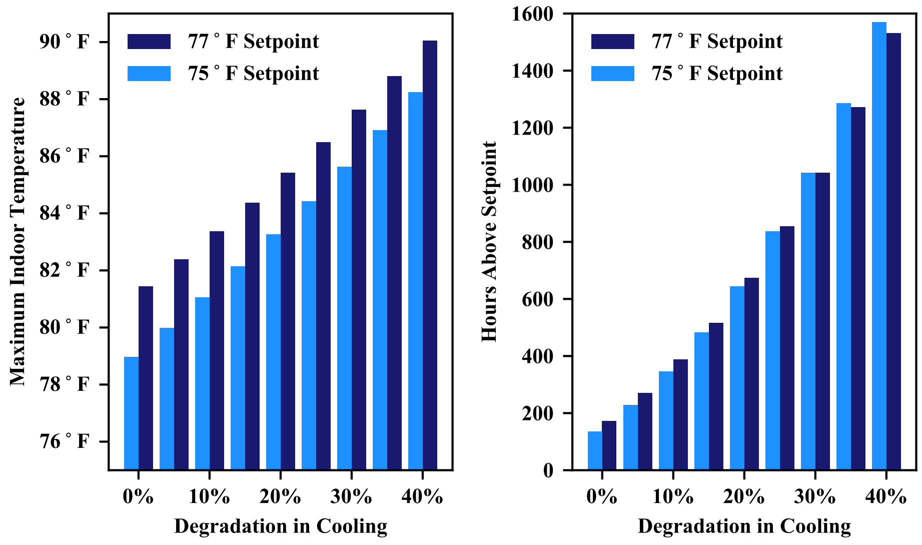


Figure 1.4: An analysis of the impact that degraded performance has on occupant comfort in Phoenix, AZ. A residential single family EnergyPlus model [17] with a slab-on-grade foundation is used with varying sizing factors. Two metrics are used to evaluate the occupant comfort: (1) the maximum annual indoor temperature and (2) the number of hours annually that the indoor temperature is greater than 0.5°F above the setpoint.

about 100 hours annually above the setpoint.

Figures 1.3 and 1.4 show that the level of discomfort rises fairly quickly as an air conditioning system degrades in these warm climates. The maximum indoor temperature rises somewhat linearly with the degradation in cooling capacity, but the number of hours above the setpoint grows exponentially. If capacity has degraded 20% and the setpoint is kept at 75°F, then the indoor temperature reaches levels above the setpoint for approximately 400 hours in Houston and 600 hours in Phoenix. For many homeowners and occupants, the impact of faults on comfort will provide a stronger incentive to implement effective FDD methods than the potential cost savings will.

The observed degradation in comfort associated with a reduction in cooling capacity is fairly intuitive, and many occupants are already familiar with this behavior. On the hottest summer days, even an appropriately sized system may not be capable of maintaining a reasonable setpoint. However, if the system cannot maintain the setpoint on a daily basis, then the system is likely under-sized or has developed a fault. This behavior is shown with thermostat data for a real home in Figure 1.5. The indoor temperature rises between 2.5 and 3.5 °F above the setpoint each day such that the indoor conditions are consistently uncomfortable.

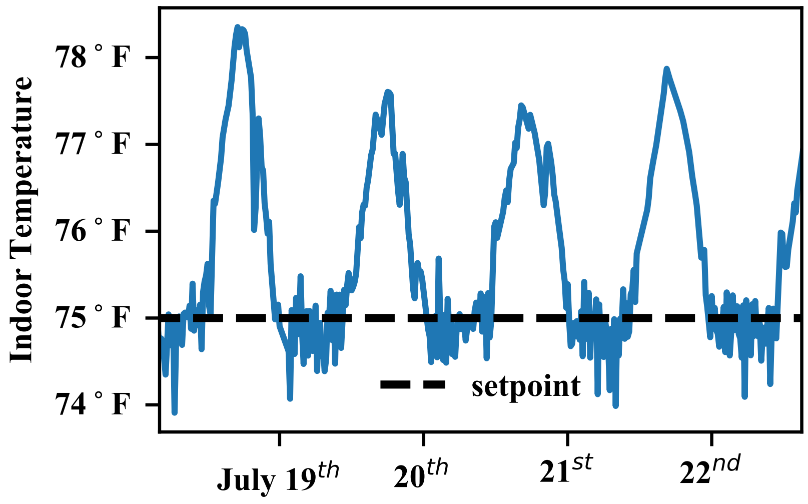


Figure 1.5: An example of the thermostat data for a real home in which the air conditioning system is unable to maintain the setpoint on a daily basis.

The cooling capacity could be estimated in spring or early summer using automated FDD methods, and the occupant comfort during the hottest days of the year could then be predicted. Faults could then be addressed during cooler weather such that occupant comfort is affected only minimally by the required service.

1.2 State-of-the-Art

Air conditioning fault detection and diagnosis reduces operating and service costs and improves the occupant comfort. There are several other benefits that were not explored in the previous section including the reduction of peak electric demand, and improved services throughout the value chain (e.g. installation and commissioning, service, sales, and system design). In response to these potential benefits, considerable research efforts have been expended on developing cost-effective and reliable FDD systems for air conditioning systems. While the majority of these methods have been developed for commercial systems [4, 5, 6], there have been some methods developed specifically for residential systems, and much of the work for smaller commercial systems is applicable to residential split systems.

1.2.1 The Commercialized State of Air Conditioning FDD

FDD products for larger commercial buildings are currently available in the form of software-as-a-service products which generally rely heavily on the existing data streams available through the building automation system (BAS) [18]. FDD products for smaller commercial and residential buildings are available as tool kits designed to assist in analyzing a system during a service; the sensors are installed temporarily during the service and then removed after the service is complete. Permanently-installed FDD systems are not typical for smaller air conditioning systems.

1.2.2 The State of Air Conditioning FDD in Published Literature

In order to achieve further commercialization, a significant amount of research effort has been expended for the purpose of improving the cost effectiveness and reliability of fault detection and diagnosis (FDD) methods for air conditioners. One of the earliest proposed methods was the The Statistical Rule-Based (SRB) method [19, 20]. The SRB used a set of seven residuals derived from measurements and low-order polynomial fault-free models as an FDD feature set. A Bayes classifier was used to detect faults and then a set of qualitative rules was used to identify the pattern that the features follow and to diagnose the fault.

The authors of [21] proposed an FDD method similar to the SRB. The authors suggested several improvements including (a) the combination of neural network and polynomial models for fault-free modeling and (b) simplified fault detection and diagnosis classifiers which require fewer assumptions and *a priori* knowledge. In [22, 23], the same authors propose a set of physics-based virtual sensors in order to decouple faults and therefore perform fault detection and diagnosis in a single step. The authors asserted that the decoupling fault diagnosis methods are advantageous to the earlier rules-based diagnosis methods because they were better-suited to handle multiple simultaneous faults.

The decoupling FDD methods have arguably achieved the greatest level of development in published literature because these methods are the only ones among those reviewed that have been implemented on field-operating systems. The methods were originally implemented on a 'field emulation site' at Purdue University and then tested on several field-operating rooftop units in California [7]. After further development, a similar set of virtual sensors were implemented on field-installed rooftop units in Washington state [24].

There have been several other proposed air conditioning FDD methods. The Simple Rule-Based Method [25] was proposed as a method that did not require fault-free models. Features were derived from raw measurements only, and features were chosen based on the criteria that they would be sensitive to faults while insensitive to operating conditions. Another set of methods is comprised of those that leverage Non-Invasive Load Monitoring (NILM) [26, 27]. These NILM-based methods measure the system power with a high sampling rate and then detect and diagnose faults using both the transient start-up data and steady state operating data.The NILM-based methods are among the few that rely heavily on transient data; the majority of methods use a steady-state filter to remove transient data and then apply static models. Most of the methods introduced here rely on low-order polynomial regression models and the underlying physical relationships. Several methods have proposed the use of more advanced machine learning methods such as support vector machines [28, 29] and clustering [30] for the purpose of fault detection and diagnosis. As with many other methods, these data-driven methods perform well with laboratory data but have yet to be applied to field-operating systems.

1.3 Overview of the Dissertation

In reviewing the state-of-the-art for air conditioning fault detection and diagnosis (FDD) methods, the need for low-cost and reliable systems that are well-suited for residential systems becomes apparent. Two broad areas in which to develop residential FDD have been identified. If FDD is to be performed using installed sensors, then the cost of the installed sensor set must be reduced; reducing the cost may involve reducing the number of sensors, using lower-cost sensors, or simplifying the installation of the sensors. Residential split air conditioning systems present a unique challenge in that installing sensors on both the indoor and the outdoor unit is more complex than only installing sensors on one of the units. In order to reduce and simplify the required sensors and possibly install sensor in only one location, the desired level of fault diagnosis must be considered.

A second area of development for residential FDD methods is in the use of cloud-based thermostat data to perform anomaly detection or change point detection and thereby detect faults. In the past, researchers have not had access to thermostat data on a large scale, and only recently have analyses been published using a substantial amount of data [31, 32]. While thermostat data is well-suited to detect hard faults in which there is a total loss of cooling, detecting soft faults or degraded performance is far more challenging, and much research will be performed in this area as data becomes more available to researchers.

This dissertation presents recent advances in the field of fault detection and diagnosis for residential air conditioning systems using both (a) installed sensors and (b) thermostat data. Chapter 3 presents a detailed review of published literature in this field and Chapters 4 through 7 present specific methods developed as part of the record of study presented in this dissertation. Chapters 4 and 5 present methods more specific to FDD methods in which sensors are installed, Chapter 6 presents a method specific to thermostat data, and Chapter 7 presents a general change-point detection algorithm and applies it to both sensor-based diagnostic data and thermostat data. The following chapter, Chapter 2, provides a summary of each of the individual contributions and more clearly establishes the over-arching research theme that connects them. Finally, Chapter 8 proposes several opportunities to continue developing the fault detection and diagnosis methods for residential air conditioning systems.

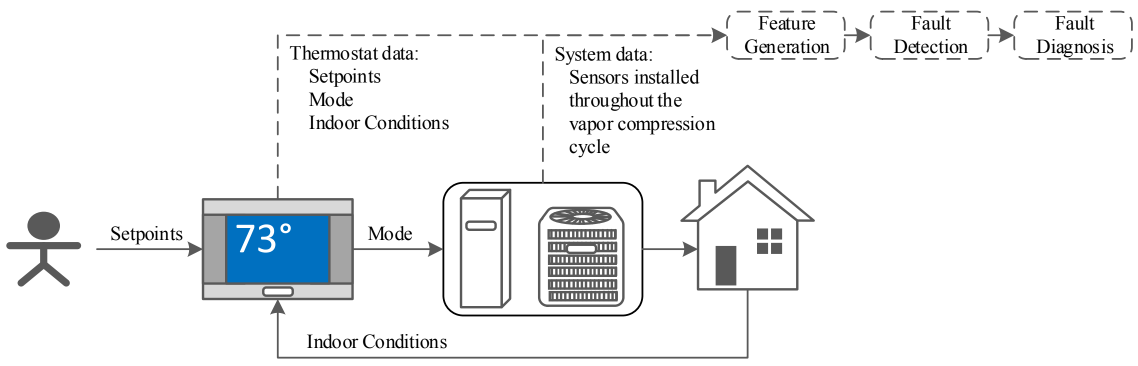
2. SUMMARY OF CONTRIBUTIONS

2.1 Fault Detection and Diagnosis for Residential Air Conditioning Systems

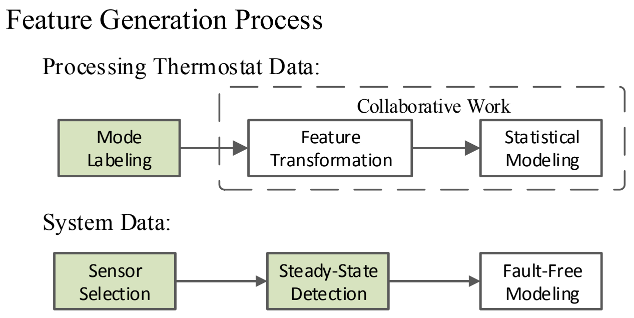
This dissertation furthers the effectiveness of fault detection and diagnosis methods for residential air conditioning systems. A comprehensive literature review of fault detection and diagnosis (FDD) methods for vapor-compression-based air conditioning systems [33] identified opportunities and challenges specific to the residential market and then divided the opportunities into two primary areas of research: (1) traditional fault detection and diagnosis methods in which sensors are installed throughout the vapor compression system, and (2) thermostat-based fault detection methods in which thermostat data is used to identify anomalous systems or systems that show a degradation in performance. The primary distinction between these two areas is the source of data used (operating data from installed sensors or thermostat data), and processing these different data sources requires different methods.

Thermostat data includes the setpoints that the user selects, the indoor conditions measured at the thermostat, and the system mode that the thermostat selects. The advantage to using this data for FDD is that this data already exists within the air conditioning system. However, the FDD capabilities using thermostat data are limited because this data contains relatively little information about how the air conditioning system is operating. In order to gain further insight to the operating behavior, sensors need to be installed throughout the vapor compression system.

Figure 2.1a provides a representation of how fault detection and diagnosis (FDD) may be built into a typical residential air conditioning system using either thermostat data or operating data. This representation shows how the user provides the setpoints to the thermostat, the thermostat then measures the indoor conditions of the home and sets the system mode appropriately. Without additional sensors installed, the air conditioning system is essentially a block box which influences the home (e.g. heats or cools) according to the mode that is selected. An FDD system requires either the existing thermostat data or data from additional installed sensors.



(a) A representation of how FDD methods may be built into existing air conditioning control architecture. FDD methods may use thermostat data, which is already present in the system; and FDD methods may use system data, which requires installing additional sensors through the vapor compression cycle.



(b) A representation of the proposed process to perform feature generation using both thermostat data and system data. This dissertation focuses on the shaded processes, and collaborative efforts to develop more complete methods for thermostat data will be presented in a future record of study.

Figure 2.1: Representations for (a) how fault detection and diagnosis (FDD) may be built into existing air conditioning architecture, and (b) the proposed process to generate features for the purpose of FDD.

Figure 2.1b outlines the basic feature generation methods required to perform FDD, and this outline highlights that the data processing for thermostat data is different from data processing for system data. Typical data processing for system data includes the selection of sensors necessary to obtain the desired features, a steady-state detector in order to remove transient data, the application of static fault-free models, and then fault detection and diagnosis [33]. In order to process thermostat data, this dissertation proposes a feature generation method referred to as the 'mode labeling' algorithm. In a collaborative effort, the features generated using the mode labeling algorithm are then transformed in order to be analyzed using statistical models. The statistical models are generated using the population of thermostat data available such that the FDD process takes the form of outlier detection. The feature transformation and statistical modeling methods will be presented at a later time [34, 35].

This dissertation focuses the shaded processes in Figure 2.1b and on fault detection and fault diagnosis methods. This includes proposing:

(1) the mode labeling algorithm for thermostat data,

(2) a unique set of features and the sensors required,

(3) a novel approach to steady-state detection,

(4) a fault detection method requring minimal tuning and *a priori* knowledge, and

(5) fault diagnosis methods using the proposed features.

The remainder of this chapter summarizes each of these contributions in turn.

2.2 The Mode Labeling Algorithm for Processing Thermostat Data

Figure 2.1 outlined the fault detection and diagnosis problem when thermostat data is used. The generated features, statistical models, and fault detection and diagnosis methods are all built on what is referred to as the mode labeling algorithm. While a more complete process will be presented at a later time [34, 35], this dissertation includes a detailed presentation of the underlying mode labeling algorithm.

The thermostat database available for this project has the indoor heating and cooling setpoints, the indoor temperature, and the system mode recorded. For cycling systems (i.e. not variable refrigerant flow (VRF) systems), the thermostat's mode of operation indicates which stage the system is operating in or whether the system is off. The system will cycle on and off as needed in order to maintain the setpoints. In order to more clearly capture the system behavior, [36] proposes a preprocessing method to transform these modes of operation to more clearly capture the effort required to (a) maintain a constant setpoint, or (b) achieve a new setpoint.

2.2.1 A proposed method to preprocess thermostat data by transforming operating modes

The proposed preprocessing method transforms the raw thermostat mode of operation (on or off) to modes of operation inspired by the thermostat's purpose of controlling the indoor temperature. When the setpoint remains constant, the thermostat *regulates* the indoor temperature such that the average indoor temperature is approximately equal to the setpoint. When the cooling setpoint lowers, the thermostat then turns the air conditioning system on in order to *track* the new setpoint. Similar tracking behavior occurs when the heating setpoint rises. When neither the heating nor the cooling setpoint is activated (e.g. the indoor temperature is between setpoints or the system is manually turned off), then the HVAC system does not actively control the home temperature, and the home behavior could be considered a *free response*.

Transforming the thermostat modes from the simple recorded on/off modes to the proposed modes of control (regulating, tracking, and free response modes) is an effective method for preprocessing thermostat data for the purpose of fault detection and diagnosis (FDD). The transformed modes clearly capture the cooling/heating effort required to maintain a constant indoor temperature separately from the effort required to change the temperature. The process by which this mode transformation is performed is outlined in Figure 2.2. As shown in this decision tree, the mode labeling algorithm relies on two key parameters: the Midpoint Deviation d and the Cycle Count N.

A brief explanation of the algorithm's two key parameters will be provided here, and [36] explains the logic in greater detail. At a high level, the Midpoint Deviation d is simply a measure of how similar an 'on' or 'off' period is to the periods preceding and following it. The Midpoint Deviation is defined as the greatest distance that the midpoint temperature (i.e. maximum/2 + minimum/2) during the current period deviates from the range of temperatures either preceding or following the current period. In other words, the Midpoint Deviation is the maximum that the distance is either *above the preceding/following maximum* or *below the preceding/following minimum*. If this Midpoint Deviation deviates too far from the surrounding range (passes a threshold), then the current cycle will not be considered a regulating period.

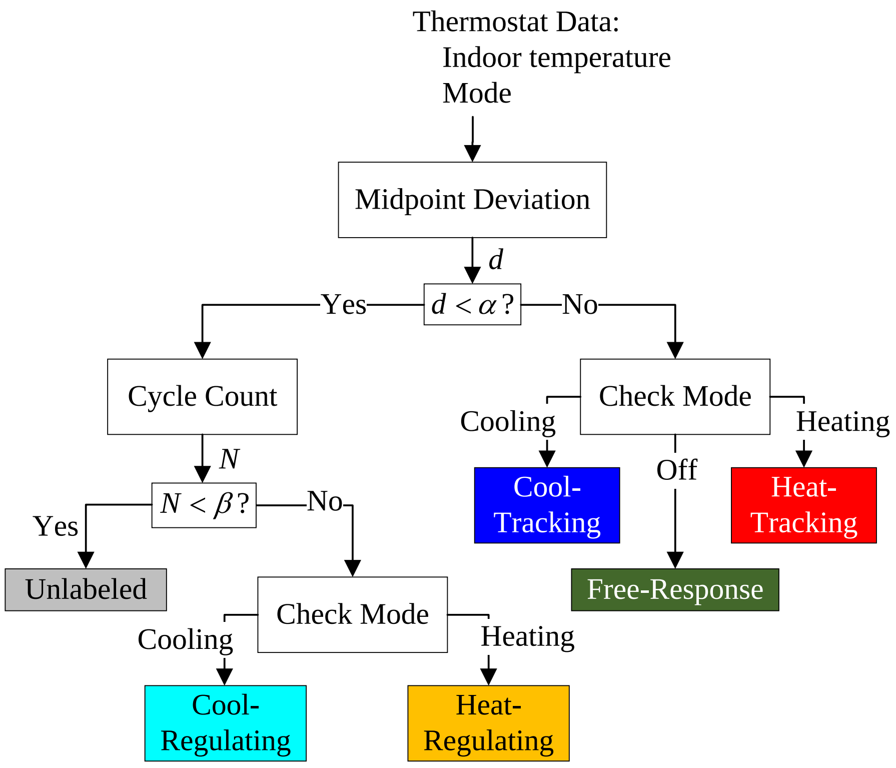


Figure 2.2: An overview of the mode labeling algorithm presented in Chapter 6.

The Cycle Count parameter is the number of consecutive cycles that do not deviate substantially from the surrounding periods. Placing a threshold on the Cycle Count ensures that the regulating modes are persistent; each regulating mode must contain a certain number of on and off periods. The core of the mode labeling algorithm may be expressed in terms of the Midpoint Deviation and Cycle Count parameters as shown in Figure 2.2. However, additional conditions may be added to further refine the algorithm.

2.2.2 Proof-of-concept that the proposed preprocessing method highlights faulty behavior

The proposed mode labeling algorithm was applied to real thermostat data, and Figure 2.3 includes several examples of indoor temperature time series with the modes labeled. The algorithm effectively identifies when the HVAC system is regulating at a constant indoor temperature, when the system is trying to track a setpoint, and when the setpoints are not active such that the home is in a free response.

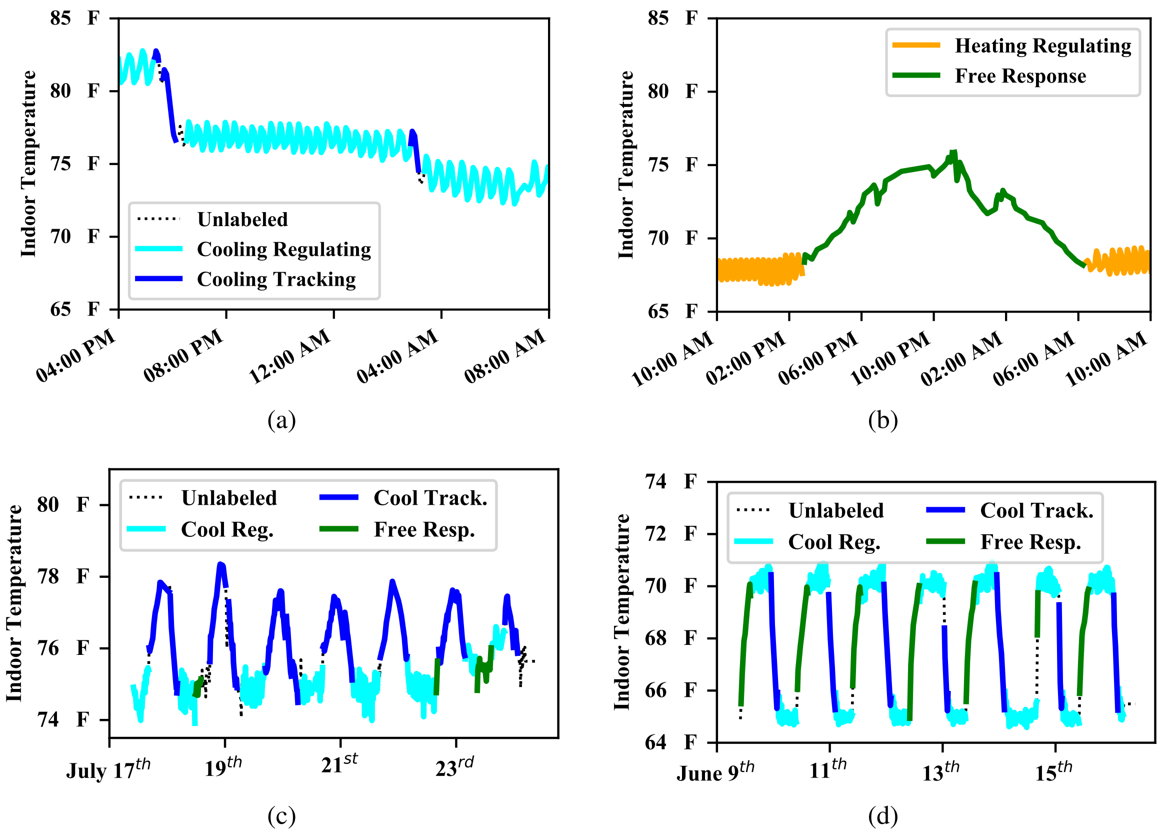


Figure 2.3: Examples of the mode labeling algorithm applied to real thermostat data. Presented in Chapter 6.

The modes of operation obtained using the proposed algorithm provide clearer insight to the relationships between key features such as the duty cycle and the outdoor temperature. For example, without preprocessing the data, the duty cycle may appear higher than expected during a given time period if the cooling setpoint were lowered during that period. Preprocessing the data using the mode labeling algorithm allows similar modes of control to be compared directly to one another. The duty cycle may be compared for all regulating modes, and the time required to change the indoor temperature may be compared for all tracking modes.

Analysis of the thermostat operation after preprocessing should be performed on two important, and fundamentally different, scales. Firstly, the operation from a single system may be analyzed such that changes in operation may be identified as the current time period is compared to previous time periods. This analysis could potentially capture the gradual degradation of an air conditioning system, the sudden onset of a fault, or the corrected behavior after a service. However, many observed changes may be due to factors unrelated to the condition of the HVAC system including seasonal changes, a change in occupancy behavior, and changes in the home fenestration or plug loads.

The second scale on which the processed thermostat data should be analyzed is on a multi-system scale. The operation observed in similar types of systems or similar types of homes should be compared to one another in order to identify outliers. Outlier detection methods for preprocessed thermostat data using uni- and multi-variate statistical methods as well as deep learning methods are currently under development.

2.3 Selecting Features and Identifying the Required Sensors

Feature selection and sensor selection are closely related in the field of fault detection and diagnosis (FDD). Sensors may be selected to obtain the desired features, and features may be selected based on the sensor limitations. One of the earliest air conditioning FDD methods, the statistical rule-based (SRB) method [37], proposed that common faults could be detected and diagnosed using low-cost temperature sensors and then proposed the following seven features: (1) evaporating temperature, (2) suction line superheat, (3) condensing temperature, (4) liquid line sub-cooling, (5)

Table 2.1: Common Faults and their Associated Decoupling Features Proposed in [23].

	Fault	Feature
1	Compressor Flow Loss	Compressor discharge temperature
2	Non-condensables	Temperature difference between the condensing temperature and the condensing saturation temperature (requires a pressure sensor)
3	Condenser Fouling	Condenser airflow rate
4	Liquid Line Restriction	Pressure drop in the liquid line
5	Evaporator/Filter Fouling	Evaporator airflow rate
6	Incorrect Charge	Weighted difference between condenser subcooling and evaporator superheat

compressor discharge temperature, (6) air temperature rise across the condenser, and (7) air temperature drop across the evaporator. Several later methods used this same feature set or a similar feature set [25, 21, 38]. Faults could be detected when one of these features deviates significantly from its expected value (obtained from a fault-free model), but fault diagnosis was less straightforward. The proposed diagnosis methods consisted of rule-based methods in which the pattern of errors (or residuals) between the measured value and the expected value were compared with the pattern that results from various faults. One of the challenges with these methods is that they cannot diagnosis multiple simultaneous faults, and [23] proposed a set of features that decouple faults in order to address this challenge.

In the decoupling methods, there is one feature for each fault and the associated feature is designed to be sensitive only to that fault. For example, the evaporator airflow rate is used as a feature to indicate evaporate fouling faults. Table 2.1 lists the feature associated with each fault. Each of these features was derived from temperature measurements (and a pressure measurement for non-condensables).

While the decoupling features simplify the fault diagnosis process and enable the diagnosis of multiple simultaneous faults, the features still require many sensors on both the indoor and the outdoor unit. This is not such a major concern when considering FDD for a packaged rooftop

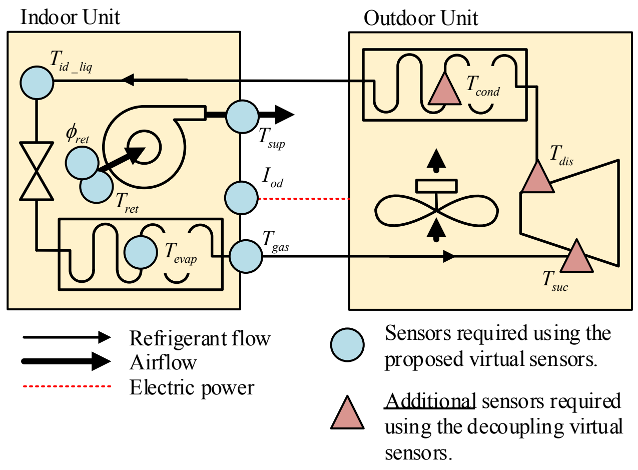


Figure 2.4: The sensors required to estimate the indoor airflow, cooling capacity, system efficiency, and refrigerant mass flow rate using the propose virtual sensors, and the additional sensors required to estimated these same features using the decoupling virtual sensors from [23].

unit, but installing sensors on both the indoor and outdoor unit of a split system complicates the data acquisition requirements. This dissertation proposes a novel set of features derived from virtual sensors to address this issue, thus providing a more practical approach to residential air conditioning FDD [39].

The proposed set of features includes system-level parameters: (1) indoor airflow rate, (2) cooling capacity, (3) system efficiency, and (4) refrigerant mass flow rate. These features are derived from low-cost measurements using virtual sensor methods similar to those used in [23, 40]. However, each of these may be estimated using only sensors installed on the indoor unit; no sensors need to be installed on the outdoor unit, and therein lies the advantage of this feature set. However, the original virtual sensors methods [23] required outdoor unit sensors to obtain these same features. Only by using elements from the virtual airflow sensor presented in [40] can the outdoor sensor be eliminated. Figure 2.7 presents a schematic of a split air conditioning system and labels the sensors required to estimate the proposed features as well as the addition outdoor-unit sensors that would be required using previously published methods.

The above four features inherently decouple indoor airflow faults (e.g. evaporator/filter fouling and blocked ductwork) from other faults by including the indoor airflow rate as a feature. However, other faults will be coupled with the remaining features. Rule-based methods (similar to those used in the SRB method may be used to diagnose condenser fouling, liquid line restriction, incorrect charge, and compressor flow loss. In this way, the proposed system-level feature set provides a balance between the original rule-based fault diagnosis method and the decoupling diagnosis method that uses virtual sensors. These features will be further expounded upon when discussing their fault diagnosis capabilities in Section 2.6.

2.4 Steady-State Detection

The majority of air conditioning fault detection and diagnosis (FDD) methods use static models and thus require a steady state filter to remove the transient data [33]. However, air conditioning systems often will not achieve steady state due to drifting driving conditions and may not run long enough to even achieve a meaningful pseudo steady state. Therefore, in many circumstances, a traditional steady state filter will fail to find any steady state data from a field-operating system. This dissertation proposes a method to predict the equilibrium operating point using the start up transients [41]. Using this method, static models may be appropriately applied using the predicted equilibrium point, but steady state does not need to be achieved. Because steady may not be achieved, the proposed method is not a true steady state filter. However, the method performs the same function as a steady state filter and considering it a steady state filter may be helpful.

2.4.1 Prediction of equilibrium point using a first-order model

Many air conditioning parameters will exhibit a first-order dynamic response because the thermal mass in the system will dominate higher order dynamics. In order to predict the equilibrium point of a first order system, an exponential function may be fit to the transient response. A regression method using an integral equation [42] is leveraged in order to perform regression using an exponential curve with a non-zero assymptote. The general exponential function is given below.

$$y(t) = y_f - (y_f - y_0)e^{-t/\tau} \tag{2.1}$$

Performing regression using this general exponential function may be transformed to a linear regression problem by first integrating both sides of Equation 2.1. This results in Equation 2.2, which may be solved using linear regression.

$$\begin{aligned} S(t) =& y_f(t - t_1) + \tau(y_1 - y) \\ \text{where } S(t) =& \int_{t_1}^{t} y(u)du \end{aligned} \tag{2.2}$$

where (t_1, y_1) is the first data point available. When performing linear regression using Equation 2.2, the error in the cumulative integral of y is minimized rather than the error in y itself. The above formulation provides a simple and reliable means to predict the final value y_f of a first-order system. The set of virtual sensors presented in [39] exhibit a first order response, and this prediction method was used to validate the virtual sensors' sensitivity to faults.

2.4.2 Analysis of the equilibrium prediction's effectiveness

The proposed prediction method uses the exponential regression method from [42] in order to predict the equilibrium operating point of a first-order dynamic system so that static models may be applied using transient data. In order to understand the effectiveness of this prediction method, thousands of exponential time series were randomly generated with different characteristics, and the prediction was performed on each time series. This Monte Carlo approach provides insight to the average accuracy with which the exponential regression predicts the final value y_f.

When applied to an exponential function with Gaussian noise, the accuracy of the prediction depends on two parameters: the length of the time series and the magnitude of the noise. Figure 2.5 presents the accuracy in y_f as a function of these two parameters. The length of the time series is normalized relative to the actual time constant, and the standard deviation of the noise is normalized to the rise $y_f - y_0$. The results in Figure 2.5 are generated using 1000 simulations at each combination of parameters.

Figure 2.5 may be used as an engineer's design tool. If the noise cannot be reduced, then an engineer can use this analysis to determine the length of data required to achieve a desired accuracy.

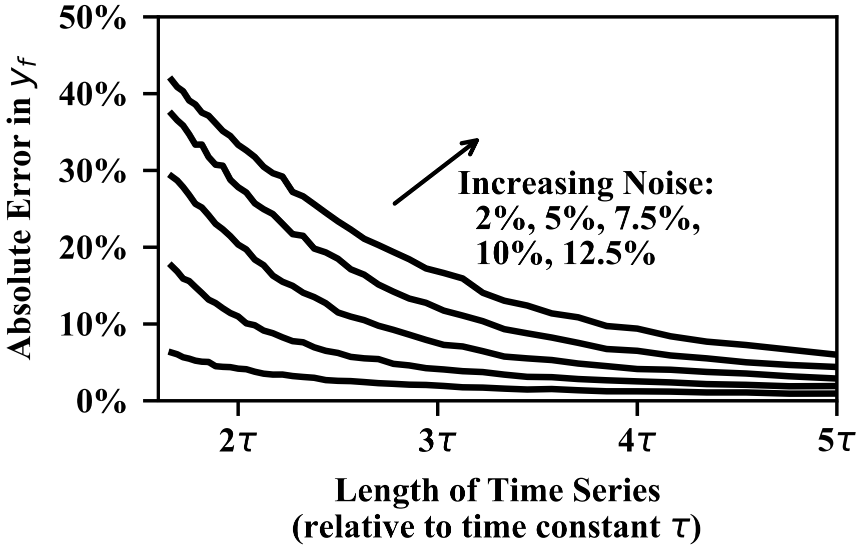


Figure 2.5: The error in the final value y_f prediction as a function of the time series length and the noise. The magnitude of the noise is relative to the rise ($y_f - y_0$). Presented in Chapter 5.

An engineer can also evaluate the advantage noise reduction would provide. While Figure 2.5 is a useful in understanding the exponential regression's accuracy, it is not a complete analysis. The 'failure rate' of the regression (defined as the percentage of simulations which results in a *negative* time constant such that the best-fit exponential function grows to infinity) and the accuracy of the estimated time constant τ must be considered in addition to the final value accuracy [41], but this full analysis is only included in Chapter 5.

2.4.3 Application of the equilibrium prediction to air conditioning data

Having obtained an understanding of the accuracy of the prediction method, it must be applied to air conditioning data. Multiple metrics are required in order to understand the full value that this method provides. The value in the prediction method comes from reducing the run time required to apply static models to air conditioning data for the purpose of fault detection and diagnosis (FDD). Therefore, metrics used to evaluate its value include (1) the accuracy of the prediction, (2) the time constant of the dynamic response, (3) the resulting correlation between the predicted equilibrium point and the fault level, and (4) the time saved in estimating the equilibrium point using the prediction. Together, these four metrics provide insight to the method's value.

The use of the time constant of the dynamic response as a metric to evaluate value may not be intuitive and some explanation is warranted. The prediction would be of very little value when applied to a system with a very fast dynamic response. Such a system will quickly achieve steady state, and the prediction is unnecessary. The prediction is most valuable when applied to a system with a slow dynamic response because the equilibrium can be predicted well before the system ever achieves steady state.

Perhaps the most important metric in evaluating the value of the equilibrium prediction for the purpose of FDD is the correlation between the predicted equilibrium and the fault level. As the length of the time series used to make the prediction grows shorter, the accuracy of the prediction will likely decrease. However, the fault prediction may still be highly correlated with with the level of fault such that FDD may still be performed. The coefficient of determination (R^2 value) is an appropriate measure of correlation and shows that the correlation does decrease slightly when less data is used to make a prediction, but there is still a high level of correlation.

2.5 Fault Detection

Several approaches have been used as fault detection classifiers for air conditioning systems; a Bayes classifier [37, 20], the Normalized Distance Fault Detection Classifier [21], and the Generalized Likelihood Ratio (GLR) [28, 29] are just a few examples. The primary drawback of each of these methods is that they require *a priori* knowledge of the nominal operating behavior. In other words, assuming Gaussian distributions, they require the mean and standard deviation of the features at the given operating conditions provided that there is no fault present. This dissertation proposes a novel change-point detection algorithm that can be used as a fault detection classifier, that does not require the nominal operating distribution, and that is scalable for deployment across large datasets. The scalability of the algorithm is particularly important for the application of thermostat data, and this section will develop the algorithm for application to both operating data and thermostat data.

Reviewing the existing body of change point detection algorithms reveals three common drawbacks. The algorithm may be computationally expensive (e.g. requiring the recursive solving of optimization problems), the algorithm may require extensive tuning for the specific application in order to be effective, or the algorithm may require key assumptions and *a priori* knowledge that is impractical for a field implementation. An appropriate change point detection algorithm must therefore meet three criteria corresponding to the above-stated drawbacks. In order to be deployed across thousands of systems simultaneously, the change point detection algorithm must:

1. be computationally efficient,
2. require minimal tuning, and
3. not require unreasonable assumptions or *a priori* knowledge.

2.5.1 A proposed change point detection algorithm well-suited for large data sets

The proposed algorithm is based on the t test that is commonly used for hypothesis testing when sample sizes are small. The t-statistic for two samples X and Y with means $\bar{X}$ and $\bar{Y}$ is calculated in terms of a pooled variance s_p^2 as:

$$t = \frac{\bar{X} - \bar{Y}}{s_p \sqrt{1/n_X + 1/n_Y}} \tag{2.3}$$

The t test is applied to a time series by dividing the series into two samples with sample X occurring before sample Y. In order the most likely instant at which a change occurred, the time series should be divided at every possible point. A change is most likely to have occurred at the instant where the t statistic is maximized. However, this maximum t-statistic (MTS) will have a different probability density function than the Student's t distribution that is typically used with the t-test. The proposed methodology is to determine the MTS probability density function using a Monte Carlo analysis and then select appropriate thresholds from the density function [43].

The advantage of using the t-statistic as the foundation of the change point detection algorithm is that the t-statistic inherently accounts for the quantity of data (whether that be a large or small quantity) both before and after the point at which there may be a change. Therefore, the resulting algorithm does not require that the distribution before the change is known, nor does the algorithm require that the distribution after the change is known. The algorithm must only be applied to a variable which is *nominally constant* and which has approximately *Gaussian noise*.

The MTS algorithm (Maximum t-Statistic algorithm) satisfies the three requirements given above. First, the algorithm may be vectorized and implemented in a single instruction rather than sequentially analyzing each possible division in the time series. Second, the only user-defined variables of the algorithm are the threshold (which may be chosen uniformly for all applications) and the size of the time series. Third, the t-statistic inherently accounts for the amount of data available such that the variance and mean of the distributions do not need to be assumed.

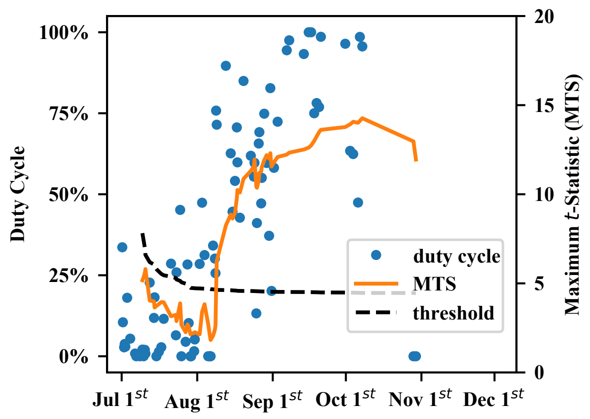
2.5.2 Analysis of the proposed algorithm's effectiveness

Intuitively, one would expect that the algorithm's effectiveness depends on both the signal-to-noise ratio and the size of the time series analyzed. As the signal-to-noise ratio (ratio of the change in mean to the standard deviation) increases, the algorithm should detect the change more reliably and with little delay. Similarly, the algorithm effectiveness will increase as the size of the time series increases.

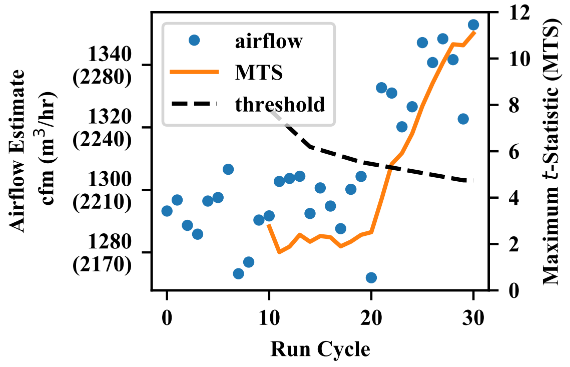
The relationship between these three variables was explored using a Monte Carlo analysis and the results may be used as a design tool in selecting the appropriate window size for the expected signal-to-noise ratio. If the change in mean is only expected to be about 1σ, then a window size of at least 100 should be used to detect the change. For a change of 1σ and a window size of 100, the change will be detected approximately 96% of the time. However, when the algorithm does successfully detect the change, it may only do so after a considerable delay: often requiring 25 or more data points after the change in order to detect the change. When the change in mean is 3σ or more, as little as 25 data points may be used to reliably detect the change.

2.5.3 Application of the proposed algorithm to air conditioning data

The proposed algorithm was applied to both thermostat data and diagnostic data generated using the airflow virtual sensor from [39]. While this implementation revealed a variety of interesting



(a) Duty cycle for 6-hour periods when the average outdoor temperature is between 8 and 10 °F above the indoor temperature.



(b) Estimate in airflow for several run periods. A change in filter restriction is manually introduced after the 21st run.

Figure 2.6: Examples of the Maximum t-Statistic algorithm applied to (a) thermostat data, and (b) the airflow virtual sensor. Presented in Chapter 7.

examples using the thermostat, the ground-truth for the thermostat systems is unknown; for example, the duty cycle may be observed to increase as shown in Figure 2.6a. However, this change may be due to an introduced fault, a change in occupant behavior, or simply a natural seasonal variation. Comparing the detected change behavior across multiple systems would provide further insight to which changes are typical and which are anomalous. Therefore, even in detecting degradation in individual systems, the development of statistical and machine learning methods for identifying anomalous behavior across multiple systems is an essential element for processing thermostat data.

In contrast to the sparse information provided by the thermostat data, the virtual sensor-based diagnostic methods presented in [39] provide a data set in which a change can be manually introduced and the change detection algorithm may be tested against a ground truth. Figure 2.6b provides an example in which a change in indoor airflow has been manually introduced, and the change detection algorithm quickly detects the introduced change when applied to the airflow virtual sensor estimate. This case study demonstrates the value of both the change detection algorithm and the airflow virtual sensor; the signal-to-noise ratio is such that a 3% change in airflow is detected after 2 data points and with only 21 data points before the change.

2.6 Fault Diagnosis

In discussing the set of proposed features, Section 2.3 briefly reviewed past diagnosis methods including the rule-based diagnosis classifiers [37, 25, 21, 38] and the use of decoupling features for fault diagnosis [23, 24]. These past methods require ten or more sensors, and require sensors installed on both the indoor and the outdoor units for residential split systems. The proposed system-level features (indoor airflow rate, cooling capacity, system efficiency, and refrigerant mass flow rate) reduces the number of sensors and eliminates the need for sensors installed on the outdoor unit while still providing some level of fault diagnosis.

2.6.1 The Virtual Sensors: Estimating System Parameters without Sensors on the Outdoor Unit

The original decoupling methods [23] obtained the refrigerant mass flow using a compressor energy balance, used the mass flow to obtain the refrigerant-side cooling capacity, and then obtained the airflow using an evaporator energy balance [23]. The method proposed herein first obtains the airflow, uses the airflow to obtain the air-side cooling capacity, and then obtains the refrigerant mass flow using an evaporator energy balance. The airflow is estimated by implementing the airflow virtual sensor from [40, 44] in a proactive manner; essentially, the airflow is estimated by applying a reliable and predictable heating source (e.g. gas heat or resistive electric heat) and then the system may be operated in cooling mode and the cooling diagnostics may proceed.

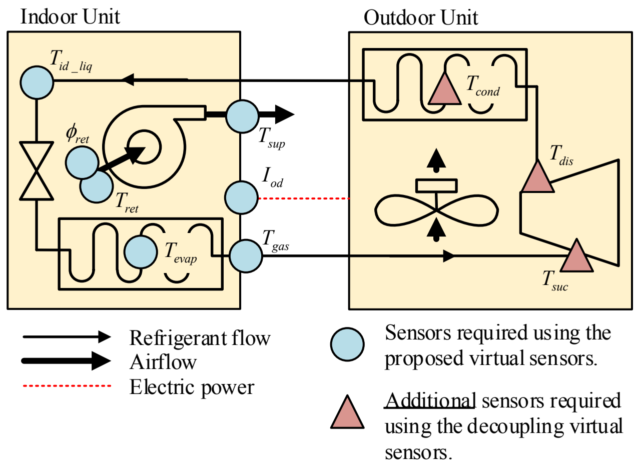
The original decoupling methods use a refrigerant-side cooling capacity estimate, and are referred to as a refrigerant sensing method (RSM). The proposed methods use an air-side cooling capacity estimate, and are referred to as an air sensing method (ASM). The sensor requirements for these two methods is summarized in Figure 2.7. By performing a compressor energy balance, the RSM requires three sensors on the outdoor unit whereas the ASM requires only indoor unit sensors.

2.6.2 An analysis of the uncertainty and sensitivity for two sets of virtual sensors

The proposed air sensing method (ASM) requires three less sensors than the refrigerant sensing method (RSM) and does not require any sensors on the outdoor unit. However, these advantages result in a trade-off in the accuracy with which the ASM can estimate the system parameters. The uncertainty of the output parameters and the sensitivity of the output parameters to the input parameters was analyzed for both methods in order to identify the sources of uncertainty. The resulting uncertainty in output parameters is presented in Table 2.2, and the underlying assumptions regarding the uncertainty in the input parameters can be found in [39].

Table 2.2: Uncertainty in System Parameters for the Refrigerant Sensing and the Air Sensing Methods. Presented in Chapter 4.

	Uncertainty	
Output	RSM	ASM
Airflow	17.3%	10.5%
Cooling Capacity	11.4%	16.2%
System Efficiency	13.6%	17.8%
Refrigerant Mass Flow	11.4%	16.2%



(a)

Figure 2.7: Sensor requirements for the air-side and refrigerant-side sensing methods.

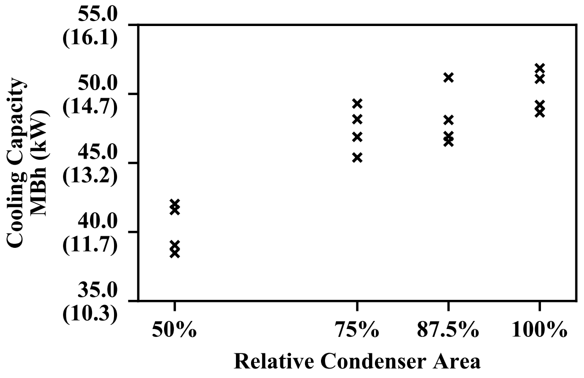
While the air sensing method (ASM) has less uncertainty in the airflow, it has greater uncertainty in the other outputs. Analyzing the sensitivity of these parameters to the inputs reveals that both the ASM and the RSM are highly sensitive to the return and supply air temperatures in cooling mode. While the RSM only uses these temperatures to estimate the airflow, the ASM uses these temperatures to estimate the capacity, efficiency, and mass flow.

2.6.3 Implementation and testing of the proposed set of virtual sensors on a field system

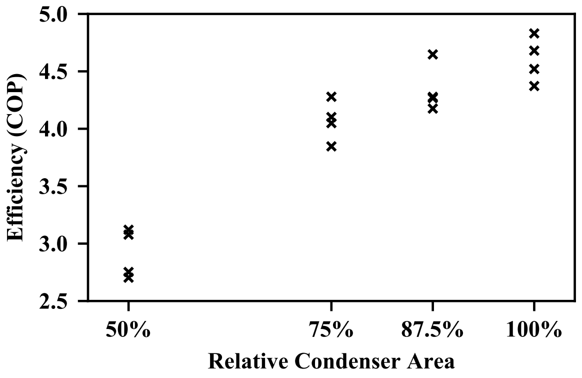
The air sensing method was implemented on a field-installed air conditioning system, and indoor and airflow faults were manually introduced in order to test this method's sensitivity to faults. These tests proved that, even with the increased uncertainty, the system parameters were still sufficiently sensitive to the introduced faults. Therefore, regardless of *accuracy* in the virtual sensors, they are sufficiently *precise* for the purpose of detecting degradation and other changes in behavior.

The estimated airflow was very close to the system nominal airflow and the estimated cooling capacity, efficiency, and mass flow were all reasonable values. However, further validation is required in order to understand the accuracy of the estimates. If the virtual sensors can be calibrated to produce a low absolute error, then they would be valuable for commissioning.

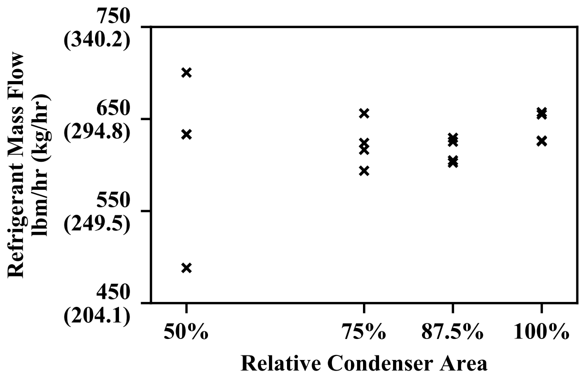
The proposed feature set (airflow, cooling capacity, efficiency, mass flow) sufficiently decouple the indoor airflow faults from other faults. Furthermore, the efficiency was shown to be more sensitive to the outdoor airflow faults than the cooling capacity, and additional fault diagnosis may be possible by considering the magnitude of effect that faults have on the cooling capacity, efficiency, and mass flow. Figure 2.8 presents the virtual sensor estimates for cooling capacity, system efficiency, and refrigerant mass flow rate for several levels of condenser airflow fault; the airflow virtual sensor is not influenced by the condenser airflow fault and would therefore remain unchanged with this fault level. Figure 2.8 shows that the system efficiency is more sensitive to condenser airflow faults than the cooling capacity is, and the refrigerant mass flow is the least sensitive to this fault. Further study is necessary to explore the level of fault diagnosis possible when considering the sensitivity of the features to different faults.



(a)



(b)



(c)

Figure 2.8: The virtual sensor estimates for (a) cooling capacity, (b) system efficiency, and (c) refrigerant mass flow rate for several levels of condenser airflow fault. Presented in Chapter 4.

3. PAPER A: A REVIEW OF FAULT DETECTION AND DIAGNOSIS METHODS FOR RESIDENTIAL AIR CONDITIONING SYSTEMS[1]

3.1 Synopsis

Fault detection and diagnosis (FDD) for air conditioning systems has been an active area of research for over two decades, but the vast majority of methods have been developed for commercial buildings. While much of this work does have application to the residential market, the residential market has unique challenges and opportunities that should be considered separate from the commercial heating, ventilation, and air conditioning (HVAC) and industrial refrigeration systems. This paper reviews and evaluates state-of-the-art methods for performing FDD for air conditioning systems. Opportunities for development in the field of applying these methods to the residential market include (a) considering the level of fault *diagnosis* that is most cost-effective in the residential market, and (b) simplifying the set of required sensors for FDD. This paper also reviews the emerging field of fault detection of residential air conditioning systems using cloud-based thermostat data. Large-scale analyses of thermostat data have only recently begun to be published, and this is a field that is anticipated to see considerable growth.

3.2 Introduction

The most commonly recognized benefits associated with successfully detecting and diagnosing air conditioning faults are (1) reducing energy consumption and (2) reducing maintenance costs. However, these benefits alone have failed to justify the cost of implementing fault detection and diagnosis (FDD) methods in the residential air conditioning sector. The majority of this paper is dedicated to establishing the state-of-the-art in FDD methods for residential air conditioning systems and identifying opportunities to reduce the cost of these methods. However, this introductory discussion seeks to establish a more comprehensive understanding of the benefits that effective FDD provides.

[1]Rogers, A.P., and F. Guo, and B.P. Rasmussen. A Review of Fault Detection and Diagnosis Methods for Residential Air Conditioning Systems. **In Review**.

3.2.1 Motivation for Fault Diagnosis

The home occupant is the primary beneficiary of reducing the energy consumption, and the home owner is the primary beneficiary of reducing the maintenance costs. However, FDD provides other important benefits to the occupant and the owner. The occupant may benefit more from the reliable comfort that FDD can provide than from the reduced electricity costs, and the home owner may benefit more from the satisfaction of the occupants than from the reduced maintenance costs. Furthermore, the home owner and occupant are only two entities in a large air conditioning value chain; many of the other entities benefit from FDD as well.

Figure 3.1 provides a simple representation of the air conditioning value chain and the benefits provided by FDD to the various entities. Reduced air conditioning loads will lead to reduced peak demand and will therefore significantly reduce the demands on the electricity generation, transmission, and distribution entities. Effective FDD methods would also improve the commissioning process of air conditioning systems and provide service technicians a means to verify the effective-

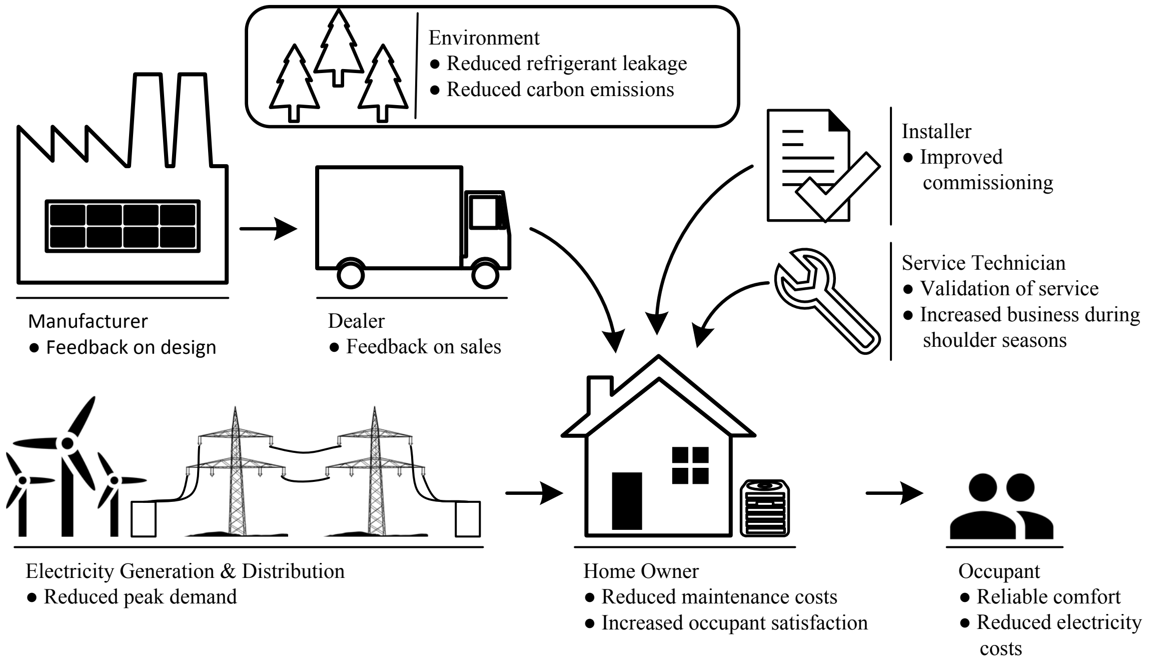

Environment
● Reduced refrigerant leakage
● Reduced carbon emissions
Installer
● Improved commissioning
Service Technician
● Validation of service
● Increased business during shoulder seasons
Manufacturer
● Feedback on design
Dealer
● Feedback on sales
Electricity Generation & Distribution
● Reduced peak demand
Home Owner
● Reduced maintenance costs
● Increased occupant satisfaction
Occupant
● Reliable comfort
● Reduced electricity costs

Figure 3.1: Examples of the benefits that automated fault detection and diagnosis (FDD) can provide throughout the air conditioning value chain.

ness of their service. Detecting faults before the home occupant notices the effects also means that faults may be addressed during the shoulder seasons thus reducing the strain on the air conditioning service industry during the summer and further reducing the repair costs for the home owner. Effective FDD methods could provide the manufacturer and dealer with feedback on the design and sales of systems in order to identify where improvement may be made and identify which systems have a history of reliability. Finally, improving air conditioning operation will provide significant benefit to the environment by reducing refrigerant leakage and by reducing carbon emissions at the power plant.

Understanding the benefit of FDD throughout the value chain is essential to achieving widespread adaptation of FDD in the residential sector. This is because someone needs to pay for the higher costs of the FDD system, and multiple parties may share the cost if these benefits are realized. For example, electric grid operators could provide an incentive in the form of a cash rebate for customers who install the FDD system, and manufacturers could offer an FDD-enabled system at a discounted price to the dealer. The dealer, installer, and service company could pay for access to the FDD data in order to receive feedback for their services.

The following studies have highlighted that there is potential for significant improvement in the areas listed in Figure 3.1.

- **Reduced maintenance costs.** In [12], service records showed that 65% of installed residential systems and 71% of installed commercial systems required repairs. Decreased performance leads to an increased work load for the compressor, and the authors of [45] reported that compressor repairs are the single most expensive repair and that compressor faults accounted for 24% of all repair costs in a database of 6,000 separate fault cases.

- **Reliable comfort; Reduced electricity costs.** The authors of [11] reported that residential air conditioners operate at least 17% below their rated efficiency. In [10], the prevalence of common faults was was explored including incorrect refrigerant charge, duct leakage, and low indoor air flow; the authors reported that, on average, a 16% increase in efficiency could be achieved simply by addressing problems with charge and airflow.

- **Improved commissioning.** Refrigerant charge is incorrect by more that 5% in more than 60% of residential air conditioners [11], and 47% of residential systems are oversized with respect to the recommended sizing calculation [10].

- **Reduced peak demand.** Degraded efficiency increases peak demand levels in addition to the overall energy consumption. In Texas, summer peak demand levels are about 25% higher than winter peak levels [46], and this is primarily due to air conditioning and chiller loads.

- **Reduced carbon emissions, Reduced electricity costs.** Combined, air conditioning and space heating account for 30% of electricity expenditures and 40% of total energy expenditures in the U.S. residential sector [3]. Given that the residential sector accounts for 39% of U.S. electricity expenditures [3, 1] and 20% of total U.S. energy expenditures [47], *residential air conditioning and space heating account for 11.7% of U.S. electricity expenditures and 8.0% of total U.S energy expenditures.*

3.2.2 Previous Reviews and Surveys

Having established a broader understanding of the benefits that fault detection and diagnosis (FDD) may provide in the residential air conditioning market, this paper will proceed with establishing the state-of-the-art for FDD in the residential sector. The potential for significant energy savings and reduced maintenance costs has spurred a considerable amount of work in the past 20 years aimed at diagnosing faults in HVAC systems. Several high quality reviews have documented these efforts and provide an appropriate starting point for understanding the broad scope of this work. This section will introduce several of the most helpful reviews, and these reviews are presented in an order such that the scope is gradually narrowed from general FDD methods (without a specific application) to FDD methods specifically for vapor compression based air conditioners. The present work is aimed at further narrowing the scope to focus specifically on the residential sector.

General FDD methods are reviewed in three parts in [48, 49, 50]. The three-part review classified FDD methods according to whether they use *a priori* knowledge of the process (quantitative and qualitative model-based methods) or not (process history-based methods). The authors readily acknowledged that this distinction can be confusing because almost all physics-based models require process data for identifying parameters (e.g. tuning a gray-box model) and many data-driven models are quantitative (e.g. polynomial regression and neural networks). In addition to this classification, a general mathematical framework was presented in [49] which highlighted how a full FDD algorithm includes several steps; raw measurements are analyzed to generate useful features and these features are used to diagnose specific faults. This three-part review was helpful for understanding the general FDD approach, but the review did not provide detail for any specific application.

The authors of [4, 5] reviewed FDD methods for building systems. The first part of this review provided useful background for FDD methods with the specific application of buildings as a motivation. The second part provided a comprehensive list of FDD publications for chillers, air conditioners, and AHUs. The authors of [6] provided an update to the review found in [4, 5] by including a comprehensive list of publications since 2004. This updated review broke down how various methods are distributed throughout the body of literature. The authors reported that 45% of publications used exclusively black box modeling methods and 42% of publications dealt specifically with air handling units (AHUs); see Figure 3.2. The two-part review [4, 5] and the update [6] were helpful for understanding how FDD work is distributed across the various HVAC

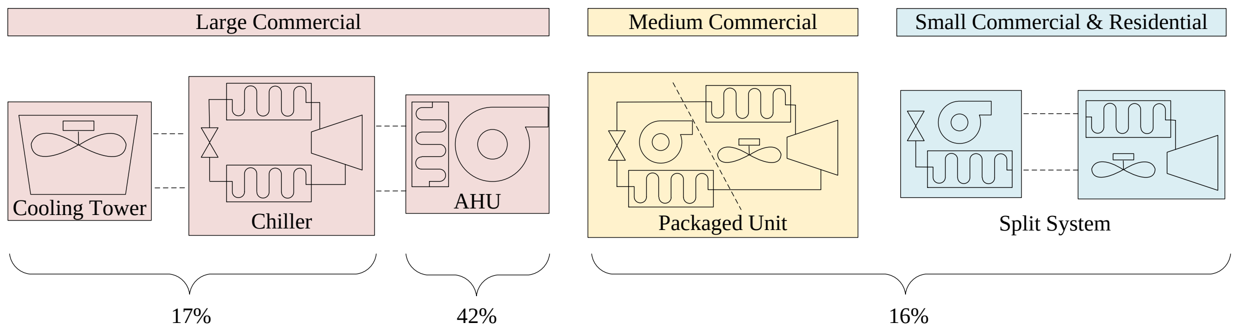


Figure 3.2: An approximate breakdown of fault detection and diagnosis (FDD) literature for building systems by system type [6]. Other categories (25%) include overall building, water heaters, refrigerators, lighting, etc.

systems, and for understanding the general FDD methods for buildings. However, by considering FDD work for all building systems these reviews did not provide substantial detail on methods specific to air conditioners.

Fourteen currently available FDD products for buildings were compared in [18]. The authors classified the products according to several categories including the intended market, the sensors used, the detectable faults, and the methods and algorithms. The authors reported that FDD is being used in nearly all larger commercial buildings and that these FDD products analyze the building automation system data rather than providing additional sensor information. The majority of methods and algorithms used are rule-based, but some process history-based techniques are emerging. The rule-based methods typically incorporate expert systems or decision trees and some physical understanding of the underlying processes. This review allowed the reader to understand the commercialized state of FDD for building systems. However, the majority of the reviewed commercialized methods were for large buildings using chillers and AHUs, and the authors highlighted the need for commercial products designed for small buildings.

The author of [51] provided a tutorial-style review of FDD methods specifically for vapor compression systems. The discussion described the important faults in vapor compression systems, the use of performance monitoring for fault detection, the process of steady-state detection, and fault detection and diagnosis classifiers. One of the greatest challenges identified was the balance between sensor cost and the simplicity of the algorithm. While temperature sensors are cheap, analyzing the temperature measurements to generate useful features is complex. The history and progress of commercially available products was also touched on; at the time [51] was published, commercially available FDD tools for vapor compression equipment were generally comprised of tools that could be installed during routine maintenance rather than tools designed for continual monitoring.

3.2.3 Overview

The previous section introduced several review papers that are relevant in the field of FDD for air conditioning. [48, 49, 50] reviewed general FDD methods, [4, 5, 6] reviewed FDD methods for building systems, [18] reviewed commercially available FDD products for large buildings, and [51] reviewed FDD methods specifically for vapor compression systems. The present review focuses on fault detection and diagnosis for residential air conditioning systems. Section 3.3 reviews the typical process for performing FDD for vapor compression systems (with an emphasis on air conditioners) and then evaluates several air conditioning-specific methods in terms of the sensor requirements and which faults are diagnosed. Section 3.4 reviews the emerging field of fault detection using smart thermostat data. In conclusion, Section 3.5 discusses several opportunities for both (a) traditional FDD methods in which sensors are installed for FDD, and (b) improving the fault detection capabilities using thermostat data.

3.3 Detection and Diagnosis of Air Conditioning Faults

3.3.1 Common Soft Faults in Air Conditioners

The vast majority of the FDD work related to air conditioners has been focused on soft faults as opposed to hard faults. Soft faults result in degraded performance without sacrificing occupant comfort whereas hard faults result in uncomfortable occupant conditions [45]. For the purposes of FDD, the distinction between hard and soft faults is that hard faults may be detected by analyzing the indoor conditions, whereas some insight to the system operation is necessary to detect soft faults. The research community has had a greater interest in soft faults because soft faults are more difficult to detect, and soft faults will often remain for an extended time because they are not detected by the occupant. Table 3.1 provides a list of both air-side and refrigerant-side soft faults that are commonly studied in the literature. Note that the faults listed in Table 3.1 are all mechanical faults, but many air conditioning service calls are due to the failure of an electric component such as a failed contactor, a wiring short, or a failed motor [45]. However, these electrical faults typically result in a hard fault and are immediately noticed and reported by the occupant. The vast majority of soft faults are related to mechanical systems such as the compressor, heat exchangers, and duct work.

A portion of the faults listed in Table 3.1 are considered service faults because they can only be

Air-Side	Refrigerant-Side
Low Evaporator Airflow – clogged air filter – evaporator fouling – closed supply registers – damaged ductwork – blower degradation – improper blower installation* – improper duct sizing* – duct leakage* Low Condenser Airflow – condenser fouling – fan degradation – improper fan installation*	Liquid Line Restriction – clogged filter/drier – malfunctioning expansion device Overcharge* Noncondensables* Undercharge – refrigerant leakage – improper service* Compressor Valve Leak Evaporator Fouling† Condenser Fouling†

* Common service faults.
† For fault diagnosis, refrigerant-side fouling is nearly identical to low airflow.

Table 3.1: Examples of Soft, Mechanical Faults in Air Conditioning Systems

introduced during the installation of the system or during a service [22]. For example, the presence of noncondensables in the system or an overcharged system indicates that a technician charged the system incorrectly. However, an undercharged system may be due to a leak in the system (an operational fault) or due to an improper charge procedure (a service fault).

3.3.2 Quantifying the Effect that Faults Have on Operation

The faults listed in Table 3.1 will affect the system in two important ways. The first is the effect that the faults have on system performance as quantified by system parameters such as capacity and efficiency. The effect on performance provides motivation for FDD and may be used to identify which faults are most detrimental and to prioritize FDD efforts. Studies that quantify this effect on performance typically take one of three approaches. First of all, several studies have analyzed maintenance service records or actively collected data during routine maintenance [12, 45, 10]. This approach has the additional benefit of providing insight to the prevalence of faults in the field, and Section 3.2.1 cited these studies to provide motivation for air conditioning FDD. Other studies either artificially impose faults in a controlled laboratory experiment [45, 52, 14, 16, 15] or impose faults in simulation [53, 54] to understand the impact that faults have on performance.

In addition to the performance degradation, air conditioning faults will also have an effect on the system operation as quantified by operating parameters such as the air and refrigerant temperatures throughout the vapor compression cycle. For example, reducing the evaporator airflow will decrease the evaporator air exit temperature and reducing the refrigerant charge will decrease the level of subcooling. While faults may be *detected* by monitoring the system performance, quantifying the effects on operation is critical to performing fault *diagnosis*. A large portion of the research in air conditioning FDD has been dedicated to understanding the effect that faults can be expected to have on system operation. The full FDD process described in the next section includes selecting appropriate measurements and then transforming these measurements into features that may be used to classify operation as fault-free or faulty and to diagnose the faulty operation.

3.3.3 The Fault Detection and Diagnosis Process

Any given fault detection and diagnosis (FDD) system is likely a combination of several methods. The mathematical framework for FDD presented in [49] describes how (1) features must be generated from raw measurements, then (2) decisions are made based on these features, and finally (3) a fault is diagnosed based on these decisions. Feature generation on its own may involve a combination of qualitative rules, physics-based models, and data-driven models. After features are generated, entirely different methods may be used to detect and diagnose faults. Previous reviews have designated an FDD system as process history-based or quantitative model-based, but in reality most FDD systems consist of several different methods.

This review will describe the general process that is commonly used for air conditioning FDD. This general process consists of (1) feature selection using the available measurements (and choosing the right measurements), (2) detecting steady-state operations and filtering data accordingly, (3) modeling the steady-state fault-free behavior of the system at the current operating conditions, (4) classifying the current operations as faulty or not, and (5) diagnosing the fault. Figure 3.3

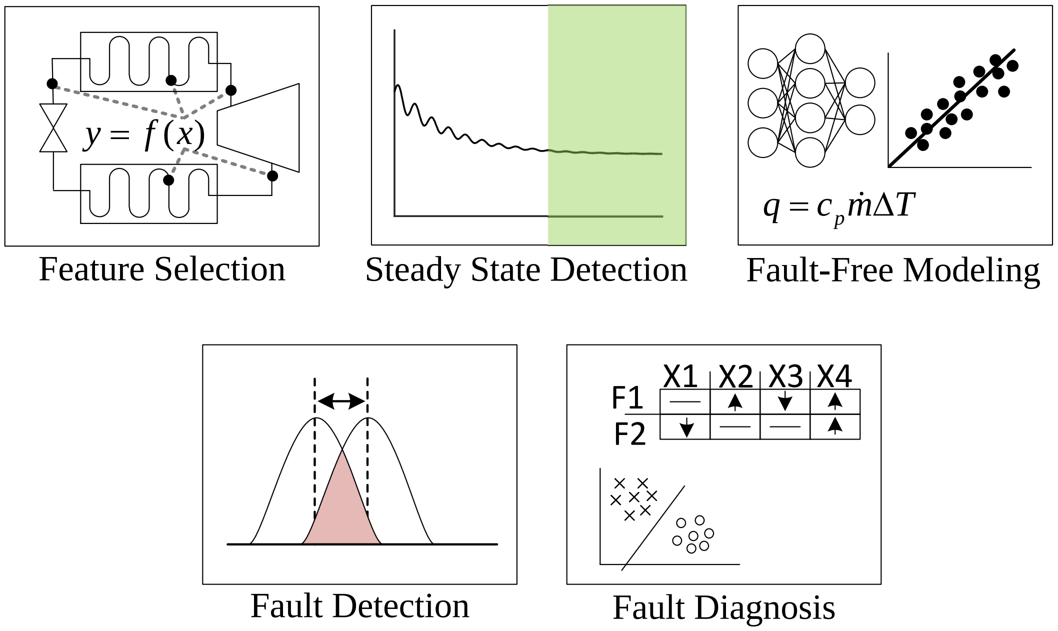


Figure 3.3: An overview of common fault detection and diagnosis (FDD) processes for air conditioning systems.

shows this process visually with simple representations of how these steps are often performed.

3.3.3.1 Feature Selection

Features are the variables that are analyzed in order to detect and diagnose faults. They may be directly measured or derived from measurements and models. For example, the refrigerant superheat is derived from measurements of the temperature and pressure exiting the evaporator. While superheat is a useful feature that is relatively straightforward to estimate using temperature sensors, many features that would be useful are difficult and expensive to obtain. One option is to directly measure a desired feature. For example, the refrigerant charge would be easy to analyze if the total refrigerant mass were measured directly. However, this is not a practical approach and considerable effort has been devoted to estimating refrigerant charge using other measurements.

In most FDD methods, a variable such as superheat is not used directly as a feature but rather the difference between the estimated superheat and the modeled fault-free superheat is used as a feature. This difference is known as a *residual*. Determining a residual requires both an estimate during current operating conditions and an estimate under fault-free operating conditions. This latter estimate is typically determined using a fault-free steady state model, and therefore Section 3.3.3.2 reviews the process of steady state detection and Section 3.3.3.3 reviews the fault-free modeling methods.

The authors of [37] proposed one of the earliest air conditioning FDD methods. The set of features chosen in [37] has influenced much of the research in following years, and these seven features include *residuals* for (1) evaporating temperature (T_{evap}), (2) suction line superheat (T_{sh}), (3) condensing temperature (T_{cond}), (4) liquid line sub-cooling (T_{sc}), (5) compressor discharge temperature (T_{dis}), (6) air temperature rise across condenser (ΔT_{ca}), and (7) air temperature drop across evaporator (ΔT_{ea}). The method proposed in [37] was further evaluated in [20] and these 7 features were used for the evaluation. These same features were used in [21] and [55] as well. These 7 measurements (with the addition of the temperature difference across the liquid-line filter/drier – T_{ll}) were referred to as "system state variables" in [23], and other meaningful features were derived from the system state variables. The features ultimately chosen in [23] were designed to decouple the various faults such that each feature would correspond with a specific fault.

The "Sensitivity Ratio Method" proposed in [25] is an attempt to further reduce the number of features. [25] used four residual ratios as features; for example, one of the features is the ratio of the evaporating temperature residual to the condensing temperature residual. The "Simple Rule-Based Method" proposed in [25] used feature selection to avoid the need for a fault-free model. This was done by choosing measurements that are sensitive to faults but insensitive to operating conditions (return air and ambient air conditions). Four features with this characteristic are presented in [25]. While the premise of the methods in [25] seems promising, the methods were not further developed, and the state-of-the-art rather shifted to the decoupling methods in [22, 23].

The authors of [22, 23] presented a physics-based FDD method that focused on choosing features that decouple faults in order to enable diagnosis of multiple simultaneous faults. For example, [23] used refrigerant mass directly as a feature by deriving a gray-box model of the refrigerant in the system. This gray-box model is termed a "virtual sensor" and was developed further in [56]. The decoupling physics-based features have found success over earlier methods, and this success is, at least in part, due to the simple fault detection and diagnosis classifiers that this decoupling method enables.

There is a strong contrast between the original seven features presented in [37] and the physics-based virtual sensors presented in [23]. In essence, the virtual sensors require more information about the system (e.g. estimated pressure losses and a model of the compressor) but the resulting classifiers are simple and intuitive. This contrast will become more evident as the associated FDD classifiers are described.

The potential features are largely limited by the available sensor measurements, and therefore sensor selection is an important precursor to feature selection. The methods described to this point placed sensors intuitively with an understanding of underlying thermodynamics. The authors of [57] used a genetic algorithm to balance the number of sensors with the diagnostic capability. This work on sensor selection was a follow-up to work previously presented in [29, 28]. This earlier research focused on the detection and diagnosis classifiers rather than on feature selection and many sensors were used. Given assumptions about the sensor costs, the genetic algorithm-based optimization in [57] reduced the number of required sensors from 48 down to 8 while retaining the accuracy reported in [29, 28]. While the optimization method may be of value, the reported case study is somewhat trivial; the sensor set was reduced from 48 to 8, but most proposed air conditioning FDD methods already propose between 7 and and 10 sensors.

Diagnosing a variety of faults requires many features and many sensors, and this drives up the price of an FDD system. The author of [58] recognized the need for low-cost FDD methods for smaller buildings and proposed a basic FDD system to meet this need. The proposed solution is to use two temperature measurements and noninvasive load monitoring (NILM) to monitor performance (efficiency and capacity) and perform some rudimentary diagnostics. NILM was first proposed in [59, 60] for the purpose of analyzing the operations of various household appliances from a single sensor, and [61, 62, 63, 26, 27] have furthered these methods for air conditioners. This set of NILM-based methods proposed to diagnose specific faults such as refrigerant charge, liquid line restrictions, and compressor leakage by analyzing the transients at a high sampling rate

(120 Hz), and the NILM-based method in [58] is significantly less sophisticated.

3.3.3.2 Steady-State Detection

The majority of FDD methods include models developed for steady state operation. The vapor compression thermodynamics develop on a much faster time scale than overall building dynamics, and therefore the system may be considered in pseudo steady state when the system parameters have settled such that the remaining transient behavior is due to drift in the external driving factors. In order to use a steady-state model, the data must be processed with a steady-state filter, and there are generally two types of steady-state filters: slope-based and variance-based filters.

The authors of [64] proposed a steady-state detector that has been referred to as the exponentially weighted variance method. The method essentially computes the running variance and exponentially weights the values such that more recent values have a larger impact on the overall weighted variance. This steady-state detector may be tuned by adjusting the threshold that determines when steady-state has been reached and by adjusting the exponential weighting. While the original application for this method was AHUs, it was adopted for roof-top air conditioning systems in [20] and [25].

The exponentially weighted variance method was found to be very similar to a simple moving window variance method in [21]. The authors combined these variance-based methods with a moving window slope method in order to improve the reliability of the filter; placing a threshold on the slope of a measured value in a moving window will detect when a measured value is oscillating around a constant mean, and placing a threshold on the variance within some moving window will detect when these oscillations fade. A similar approach was used in [57]; the exponentially weighted variance method is combined with a moving window time derivative.

The proposed steady-state filters require that the past variance or slope must fall below a threshold. The selected window-size (or the forgetting factor in exponentially weighted filters) and the threshold are important tuning parameters, and the authors of [55] described methods for selecting these tuning parameters systematically. The thresholds were determined by applying the 3σ rule to steady-state data from a large sample of experiments, and the window size was determined using the transient data from these experiments.

The features used to detect steady state must be considered in addition to the type of filter. The measurements or features used to determine steady-state are referred to as *steady-state indicators*. Additional steady-state indicators increase the required computation, but fewer steady-state indicators may lead to falsely detecting steady state. The authors of [20] observed that the discharge temperature is the slowest of system states and therefore only used this measurement as a steady state indicator. However, only the liquid line subcooling was used as a steady state indicators in [25].

The authors of [55] found that superheat and subcooling suffice as steady state indicators. However, if the indoor temperature is changing, the superheat and subcooling may misidentify steady state conditions. In this situation, the evaporating temperature and the air temperature drop across the evaporator may be used as indicators. For robustness, the authors of [55] recommended using all features as indicators, and all features were likewise used as steady state indicators in [65].

The experiments used in [20] and [55] indicate that about seven to ten minutes are required in order obtain and then detect steady state. When the cooling load is low, a cycling air conditioner may not run for more than five minutes before shutting off and thus the steady state won't be analyzed during low load conditions. However, applying FDD during low load conditions is important in order to analyze an air conditioning system during shoulder seasons before comfort is affected in the summer and when maintenance costs are low. Therefore, the system must either be run longer than would normally be necessary in order to obtain steady state data or the transients must be analyzed instead of the steady state.

3.3.3.3 Fault-Free Modeling

The residuals that are so commonly used as features require a model of the system operating in steady-state and without faults. These models typically receive operating conditions such as the return air and ambient air conditions in order to predict systems states. An assumption is often made that the only external influences on a fault-free air conditioner are the temperature of the air entering the condenser and the temperature and humidity of the air entering the evaporator [37].

The authors of [20] explored the use of both lookup tables and low-order polynomial regression models in order to estimate fault-free values. While lookup tables can represent highly nonlinear relationships with sufficient training data, the authors found that tables did not provide any benefit over the low-order polynomial models. One benefit of using low-order polynomial models is that some degree of extrapolation can be reasonably performed.

While [20] compared polynomial models with look up tables, [66, 38, 67] compared polynomial models with neural networks. The authors of [66] focused on improving steady state models for FDD in packaged air conditioning systems, and the authors compared four black-box modeling methods: polynomial regression, general regression neural networks (GRNN), radial-basis functions, and back-propagation neural networks. The authors concluded that a hybrid model is best in which a GRNN is used for interpolation and a low-order polynomial model is used for extrapolation. This hybrid model was then used in [21]. The authors of [38, 67] also explored both low-order polynomial models and back-propagation neural networks and likewise concluded that third-order polynomial models are better suited as fault-free models than the neural networks due to a lack of significant nonlinearity.

The authors of [38] also included an interesting discussion on the challenges of using models in the field. One challenge specific to residential systems is the variation between installations due to the separate indoor and outdoor units and the variable lengths in refrigerant tubing. The authors of [68] presented a self training model in order to overcome the challenge of implementing models on a field-installed system. The proposed model is intended to be developed after installation. This is done by developing the model slowly as different operating conditions become available. Preliminary fault free models are used with relaxed thresholds to make an initial evaluation after installation. If installation errors are not detected, then the models are developed further. These models continue to be developed as more data becomes available. When the operating conditions cover the desired range then the final fault-free models are developed and the self training ceases.

The models presented in [68] are the only solution presented in this review for developing models on field-installed systems. The majority of methods use an extensive set of laboratory experiments in order to characterize the fault free behavior. However, such experimental results will not be available for all systems, and even when available, these models may not capture the variation in split system installations. Installed systems will likely need to be benchmarked during the commissioning process, and methods such as those proposed in [68] should be developed further to customize the model for a specific installation.

3.3.3.4 Fault Detection Classifiers

Once the features have been generated using measurements and models, fault detection and diagnosis may proceed. The method by which a fault is detected or diagnosed is referred to as a *classifier*. While detection and diagnosis classifiers are often performed separately, they are sometimes performed in a single step. For example, the authors of [38] performed fault detection and fault diagnosis with a single classifier. Also, if features are decoupled such that each feature corresponds to a specific fault, then fault detection and diagnosis is performed simultaneously. In other words, detecting a fault using a decoupled feature is the same as diagnosing the fault associated with that feature. This is the premise of [22, 23].

The most common fault detection classifier consists of simple thresholds on the features. If a feature exceeds or falls below the threshold, then a fault has been detected. Some methods such as the “Sensitivity Ratio Method” and “Simple-Rule-Based Method” from [25] and the decoupling methods from [23] use this type of classifier. However, several more rigorous classifiers have been presented.

A Bayes classifier was presented in [37] for determining when one of the residuals has a significant value according to a given statistical confidence. However, this method required the full covariance matrix under both current and fault-free operations. When implementing this method, [20] simply assumed the same diagonal covariance matrix for both current and fault-free operations, and this simplification detracts from the rigor with which the method was originally presented. To overcome this simplification, the authors of [21] proposed the “Normalized Distance Fault Detection Classifier.” This detection classifier essentially generalized the 3σ rule-of-thumb to higher dimensions, and the unknown covariance matrix for the current operations is then not necessary.

The generalized likelihood ratio test (GLRT) was used as a detection classifier in [29, 28, 57]. Like other likelihood tests, the GLRT is used to determine how likely it is that the null hypothesis should be rejected. In this application, the GLRT is used to detect the abrupt introduction of a fault, and therefore the null hypothesis states that operations have not changed. The GLRT requires that the nominal operating distribution be known, and the authors of [29, 28, 57] therefore assumed that the mean and variance before the change point were known. While this assumption is not practical, rigorous change point detection methods such as the GLRT are certaily of value for fault detection. However, the change point detection methods must account for uncertainty in the nominal operating distribution.

3.3.3.5 Fault Diagnosis Classifiers

Two general approaches to air conditioning fault diagnosis will be considered: rule-based and machine learning. With a qualitative rule-based approach, the user identifies the feature patterns for a set of faults and applies statistical methods to classify the current operations. With a machine learning approach, the patterns are generated automatically using training data.

The comprehensive review provided in [5] gave credit to [69] for laying a foundation for future diagnostic work in qualitative rule-based approaches. [69] determined if a given feature would increase, decrease, or remain unchanged under a given fault condition. A score is then generated in order to reflect the probability with which the current conditions match the fault conditions.

The authors of [37] applied these rule-based methods to packaged air-conditioners. Similar to the accompanying detection classifier, the rule-based diagnosis classifier in [37] also required the full probability distributions for both fault-free and current operations. The authors of [20] and [21] found this to be impractical, and [20] simplified this method by using the same diagonal covariance matrix for both fault-free and current operations.

The "Distance Ratio Fault Diagnosis Classifier" presented in [21] circumvented the need for a covariance matrix. In the Distance Ratio Fault Diagnosis Classifier, a point is defined for each fault type in the feature space. The distance from the current operating point to each fault point is then calculated. A fault is diagnosed when the ratio of the shortest distance to the next shortest distance falls below a preset threshold.

The fault diagnosis classifier presented in [38] is similar to that of [37], but the authors of [38] accounted for the possibility that some features won't change (increase or decrease) during certain fault conditions. This was necessary because the rules presented by [37] were developed for a packaged air conditioner with a fixed-orifice expansion device. A variable expansion device such as a TXV essentially introduces an internal control loop in which the refrigerant flow into the evaporator is controlled in response to the superheat. This internal control loop results in a more fault-tolerant system and makes fault diagnosis more difficult. With a TXV, many of the features do not increase or decrease with certain faults. For the rules in [37], each feature increased or decreased for each fault considered. However [38] included many situations in which there is no change. The authors of [38] observed that a system becomes even more fault tolerant with the use of a variable speed compressor. Similar to [20], [38] also simplified the statistics-based diagnosis classifier by ignoring cross-correlation.

While the diagnosis classifiers described to this point used rule-based methods supported by statistics, the authors of [29] and [28] explored several machine learning methods for fault diagnosis. Rather than observing patterns in the features under various fault conditions, the authors trained algorithms using experimental data of a chiller under various operating and fault conditions [70]. The explored methods are Principle Component Analysis (PCA), Partial Least Squares (PLS), and Support Vector Machines (SVM). Among these, SVMs showed the best performance.

The authors of [30] proposed a fault diagnosis classifier using clustering methods. Traditional clustering methods can only handle faults of discrete magnitudes (i.e. the severity of the fault must remain constant), but a "vector clustering" techniques is proposed in [30] to overcome this limitation and account for gradual faults. This method essentially clusters data according to how the features are developing over time.

While all of the fault diagnosis classifiers described here have been successfully implemented using laboratory experiments, none have been implemented on a field-operating system. However, the decoupling methods presented in [22, 23] and then used in [7, 24] have been implemented on field operating systems. These methods perform fault detection and diagnosis simultaneously because each feature corresponds with a specific fault.

3.3.3.6 Transient-Based Analysis

The FDD process that has been outlined includes feature selection, steady state detection, fault-free modeling, and fault detection and diagnosis classifiers. Each of the methods described thus far have relied on a steady state analysis. However, several researchers have found advantages in analyzing transient periods.

The authors of [27] used non-invasive load monitoring (NILM) to analyze the transient compressor and fan power and to diagnose improper refrigerant charge, condenser fouling, and liquid line restriction. The use of transients allowed [27] to diagnose many common faults with the use of just one sensor. Also, the methods were capable of diagnosing faults such as slugging and a flooded start. The authors proposed that the NILM sensor be used with temperature sensors to improve the confidence of FDD.

While the majority of decoupling methods presented in [23] rely on steady state operations, the authors used the transient liquid line temperatures to determine the liquid line pressure drop and isolate a clogged filter/drier fault. The authors observed that, during the startup transients, the liquid line consists of a two-phase flow, and the subcooling will therefore be near zero. While the liquid line contains two-phase flow, the refrigerant pressure on either side of the filter/drier is directly related to the refrigerant temperature. The pressure drop across the filter/drier may therefore be determined using the temperatures while the subcooling is near zero during the start-up transients.

The use of transient-based fault detection for vapor compression system faults was also promoted in [71]. The authors noted that dynamic methods could detect fan failures quicker than static methods could, and that dynamic methods were essential to detect a very slow refrigerant leak early on, especially if a receiver is present. The challenge with implementing the proposed methods is that they require an accurate model of the transient behavior, and a transient model is more difficult to obtain that a steady-state model.

Most of the transient-based methods have not progressed to field implementations. As with other proposed methods, they work well using laboratory data but a field implementation is required to validate the performance. However, the liquid-line restriction indicator presented in [23] was implemented on a field-installed unit owned by the research lab and on other field-installed rooftop units [7].

3.3.4 Evaluation of Methods

FDD methods are always evaluated to some degree when they are presented, and this evaluation generally includes an evaluation of the method's accuracy such as the rate of false positives and false negatives. A method's sensitivity to faults may also be evaluated. For example, The authors of [20] reported the level of fault severity that first caused an alarm as well as the level of severity at which an alarm was caused in all tests performed. The authors emphasized the utility of the evaluation methods as a general tool for all FDD methods. The authors of [72, 73] likewise developed a method for evaluating any proposed FDD method. The method rated an FDD method against the diagnostic capabilities of a routine maintenance visit. The authors found that methods which include some level of fault assessment (i.e. determining if service is cost effective) are most valuable.

FDD methods are almost always shown to work well when they are proposed. However, the method whereby they were evaluated is an important aspect when considering deploying them in the field. Almost all methods were developed using data collected during careful laboratory experiments in which faults were artificially introduced (e.g. obstructing a portion of the heat exchanger). Research groups either perform experiments to generate their own data, or use the data previously published by another group. After the initial development, some methods were tested with data collected from units operating in the field [7, 24]. During many of these tests, faults were artificially introduced to these field units similar to how they would be introduced in a laboratory.

Paper	Packaged/Split	Validation	Sensors[1]					Diagnosed Faults[2]						
			T	RH	P	V	I	UC	OC	LR	CL	CF	EF	NC
[37]	packaged	lab data[3]	9	1				✗		✗	✗	✗	✗	
[25][4]	packaged	lab data[5]	7	1				✗	✗	✗	✗	✗	✗	✗
[25][6]	packaged	lab data[5]	7					✗	✗	✗	✗	✗	✗	✗
[21]	packaged	lab data[7]	9	1				✗		✗	✗	✗	✗	
[23]	both	field data[8]	10	1	2			✗	✗	✗	✗	✗	✗	✗
[27]	packaged	lab data[9]				1	1	✗	✗		✗	✗	✗	
[74]	unspecified[10]	unspecified[10]	8					✗	✗					
[38]	split	lab data[5]	9	1				✗	✗	✗	✗	✗	✗	
[24]	packaged	field data[11]	8					✗	✗	✗		✗		
[75]	split	lab data[12]	7		2			✗	✗	✗		✗	✗	
[40]	packaged	lab data[5]	4	2									✗	

[1] (T) temperature, (RH) humidity or wet-bulb temperature, (P) pressure, (V) voltage, and (I) current.
[2] (UC) undercharge, (OC) overcharge, (LR) liquid-line restriction, (CL) compressor valve leakage, (CF) condenser airflow/fouling, (EF) evaporator airflow/fouling, (NC) noncondensables.
[3] Some initial testing was performed in [37], but the method was tested more fully in [20] using their own data obtained in [45].
[4] *Sensitivity Ratio Method.*
[5] Developed using their own laboratory testbed.
[6] *Simple Rule-based Method.*
[7] Testing was performed using data from [20].
[8] Initial testing was performed with data from [20]. [7] then tests the decoupling method by introducing faults on a unit operating in the field. Finally, the method was applied to field data, though no validation using this field data was performed.
[9] Testing was performed specifically for this project using a testbed at another lab.
[10] The paper is unspecific; "The experimental data was collected from several devices of the same class, installed in different locations" [74].
[11] Tested by introducing artificial faults in a packaged unit operating in the field.
[12] Developed using their own laboratory testbed as well as their own existing laboratory data. The FDD tool was used once in the field to test ease of use.

Table 3.2: An evaluation of several FDD methods for air conditioning systems. Emphasis is placed on the type of system used for development and validation, the source of data used for validation, the number sensor used, and the types of faults diagnosed.

The present review provides a simple evaluation of FDD methods proposed specifically for air conditioning systems. The methods are evaluated according to (1) how they were validated, (2) how many sensors are required, and (3) which faults may be diagnosed. Table 3.2 summarizes this evaluation, and highlights how few of the methods have been applied to field units. In fact, only the decoupling method described in [23] and a later variant of it [24] were tested using data collected from a field unit.

Note that, with the exception of the NILM-based transient method from [27] and the virtual airflow sensor from [40], each of the methods use between seven and ten temperature measurements. Many methods measure the evaporator air inlet relative humidity and use this as a driving factor to generate fault-free models, but some methods train models with only the ambient air temperature and the evaporator air inlet temperature. The *Simple Rule-based Method* from [25] is the only method listed that does not use a fault-free model but rather seeks to combine raw measurements to create features that are not sensitive to operating conditions.

Few methods claim to be capable of diagnosing noncondensables. Even the methods presented in [25] require a technician's involvement in order to distinguish noncondensables from an overcharge fault; the technician would need to check the evaporator pressure against the expected refrigerant saturation pressure at the measured temperature. The decoupling methods presented in [23] do distinguish between noncondensable and overcharge faults, but the method is similar to the technician's test proposed in [25]; the discharge pressure is measured and compared with the saturation pressure associated with the condenser temperature.

The evaluation in Table 3.2 provides an overview of the state of FDD for air conditioning systems. This evaluation shows how most of the FDD systems focus on the same group of faults and have similar sensor requirements. This evaluation also highlights that most methods have been tested in a lab, but few have been implemented on a field-operating system. Reducing and simplifying the set of sensors required for FDD will be an important part of increasing the cost-effectiveness of these methods. However, this will likely require that the diagnostic capabilities be reduced somewhat.

3.4 Fault Detection Using Thermostat Data

All of the FDD methods described to this point require that sensors be installed throughout the system in order to gain insight to the operation of the vapor compression cycle. Larger commercial systems such as chillers and AHUs come installed with many of the required sensors for FDD because this information is used for the control of the system, but these sensors are not typically included in a residential system. Residential systems are controlled with a thermostat, which simply measures the conditions of the conditioned space. However, the advent of the smart and connected thermostats has greatly increased the availability of residential thermostat measurements, and this data could be used for the purpose of fault detection.

Smart thermostat data typically includes the indoor temperature and relative humidity, the system status (e.g. cooling, heating, off), and an estimate for the outdoor temperature obtained from a third party such as the National Oceanic and Atmospheric Administration (NOAA). Smart thermostat datasets are well-suited for the detection of hard faults because they adequately capture the indoor conditions. Recall that hard faults occur when the system is not able to maintain occupant comfort. Therefore, a hard fault may have occurred when the system is unable to maintain the setpoint. Even so, there are many factors not captured by the thermostat data that will have a significant effect on the cooling or heating load (e.g. occupancy, fenestration, and plug loads) such that uncomfortable conditions may be due to abnormally high loads rather than a system fault. The detection of soft faults using thermostat data is also possible because thermostat data does provide some insight to the system operation in the form of the system status. Analyzing the runtime may indicate that a system's performance has degraded such that it runs longer or harder than it did previously.

Thermostat-based fault detection is still a relatively new field of research, and has been partially hindered by the lack of publicly available smart thermostat data. Studies using relatively small datasets and simulations have proposed methods for improving smart thermostat control [76], identifying inefficient home construction [77], inefficient occupant behavior [78, 79], and air conditioning system faults [80]. However, Ecobee's "Donate Your Data" program has made larger scale data analysis available to many researchers, and results are only recently being published [31, 32]. The authors of [31] and [32] analyze the aggregate behavior of the available data with [31] focused on user behavior and [32] focused on runtime. Therefore, these initial studies do not attempt to identify faulty behavior in specific homes.

Among all of these studies, [80] is the closest in line with detecting soft faults in air conditioning systems. The authors proposed recursive least squares models to describe the thermostat behavior and then propose that a fault may be detected when the model parameters change. However, the authors of [80] developed this model-based method using simulations and the method may not generalize well to real thermostat data. Indeed, when analyzing the thermostat data for 7000 homes, the authors of [32] noted that there is large variation in system runtime from one home to the next *even at constant ambient conditions* which indicates that there are other very important factors that are not captured in the thermostat data.

Cloud-based thermostat data provides a much needed opportunity to bring air conditioning fault detection to the residential sector. However, the process by which this data may be analyzed to perform reliable fault detection remains an open research problem. Large-scale analysis results have only recently become available, and this field of research will see considerable growth over the next decade.

3.5 Conclusions

This paper has reviewed fault detection and diagnosis (FDD) methods for residential air conditioning systems. This has included many methods developed for commercial systems because the amount of work developed specifically for residential air conditioners is relatively small. The typical air conditioning FDD process (i.e. feature selection, steady state detection, fault-free modeling, and detection and diagnosis classifiers) has been described in detail and the methods have been evaluated in terms of the sensor requirements, the faults diagnosed, and the level of validation used (lab data or field data). State-of-the-art FDD methods typically require between seven and ten temperature sensors, and the set of faults that these methods diagnose is fairly standard: undercharge and overcharge, liquid-line restriction, compressor valve leakage, and condenser and evaporator airflow faults. The physics-based gray box method proposed in [23] and adapted in [24] is the only method reviewed that was evaluated using a field-installed system.

While Section 3.3 reviewed the more traditional FDD literature described above, Section 3.4 reviewed the state of fault detection methods for residential air conditioning systems using only thermostat data. The open-source data in this field has historically been limited, but large-scale analyses of smart thermostat data have just recently been published and considerable growth in this field should be expected.

There are several opportunities for improving residential air conditioning FDD using both installed sensors and thermostat data. Installing sensors throughout the vapor compression cycle provides detailed insight which may be used to diagnose specific faults. Thermostat data is already available for many thousands of systems, and properly utilized, this data could provide effective fault detection capabilities and perhaps some limited fault diagnosis. The following sections outline a small portion of the research opportunities in these fields.

3.5.1 Opportunities for FDD Using Installed Sensors

The cost of the proposed sensor package is the main obstacle to implementing FDD methods on residential systems, and the following opportunities focus on reducing this cost.

3.5.1.1 Applying AHU-Related FDD Methods

Section 3.2.2 noted how a large portion of FDD work for building systems has focused on air handling units (AHUs). As a result, several fields of research have been explored for AHUs but have not been adequately developed for air conditioning systems. Sensor fault detection and proactive diagnostics are two such examples.

Sensor faults include a constant or drifting bias or a complete failure in the sensors used to perform FDD. Proposed methods of performing sensor fault detection have been based on principle component analysis [81, 82, 83, 84, 85], joint angle analysis [86], wavelet analysis [87, 85], and neural networks [88, 89]. Effective sensor FDD in air conditioning systems would increase the reliability of the FDD system when low-cost sensors are used.

Proactive diagnostics describe a group of methods in which a system is controlled in a non-standard way in order to gain some diagnostic insight that would not be possible during normal operation [90, 91, 92]. The authors of [24] used proactive diagnostics and a rule-based method to diagnosis air-side sensor and actuator faults in the air handling portion of a rooftop unit. The state-of-the-art air conditioning FDD methods are all designed to be implemented during normal operation, but proactive diagnostics may improve the capabilities of the methods.

3.5.1.2 Determining the Optimal Level of Fault Diagnosis

Fault *diagnosis* is admittedly a more complex problem that fault *detection*. Most of the proposed FDD methods evaluated used many sensors in order to effectively diagnose each of the faults individually. Intuitively, one could conclude that fewer sensors would be required if less fault diagnosis were performed. For example, diagnosing between overcharge and noncondensables requires a pressure sensor [25, 23], and the pressure sensor may not be necessary if these faults are not diagnosed individually. The optimal level of fault diagnosis is a balance between the cost of the FDD sensor package and the capabilities of the FDD system.

3.5.1.3 Simplifying the Required Sensor Set

In order to bring FDD sensors to the residential market, the required sensor package must be simplified significantly. This involves not only reducing the number of sensors but also accounting for the type of sensors used and considering where these sensors must be installed. The residential air conditioning market introduces a unique challenge for FDD systems by separating the evaporator and condenser by a great distance through the use of a split system. State-of-the-art methods require sensors installed on both the indoor and the outdoor unit, but the cost would be reduced greatly if sensors were limited to just one of the units. Simplifying the required sensor set will also involve optimizing the fault diagnosis capabilities as described in the previous section.

3.5.2 Opportunities for Fault Detection using Thermostat Data

The primary challenge with leveraging thermostat data is that many important factors are missing from the dataset, and the following opportunities focus on maximizing the information available when analyzing thermostat data.

3.5.2.1 Expanding the Available Data

In its most basic form, thermostat data only includes the indoor conditions and the system status. This dataset is often supplemented by an outdoor temperature estimate, but there may be other useful ways in which to supplement the dataset. For example, additional climate parameters such as solar irradiation, home parameters such as the construction date and square footage, and system parameters such as the nominal capacity may provide researchers with information necessary to decrease modeling errors using smart thermostat data.

3.5.2.2 Comparing Operation Across Many Systems

The traditional FDD methods are always focused on the operation for a single system. However, the introduction of cloud-based thermostat data has made an entirely different approach possible; the operation of many different homes can now be compared to each other and an outlier identification can be used to find homes that are the least efficient with respect to the others. Leveraging the data from many homes is another source of information that could provide insight to specific homes via anomaly detection methods. The utility of this type of approach would be greatly improved with an expanded dataset such that similar homes (in terms of square footage, location, etc.) may be compared.

3.6 Acknowledgment

This material is based upon work supported by Trane, a subsidiary of Ingersoll Rand. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Trane.

4. PAPER B: UNCERTAINTY ANALYSIS AND FIELD IMPLEMENTATION OF A FAULT DETECTION AND DIAGNOSIS METHOD FOR RESIDENTIAL HVAC SYSTEMS[1]

4.1 Synopsis

The vast majority of fault detection and diagnosis (FDD) methods for air conditioning systems are developed for packaged commercial systems. This paper presents a method for obtaining key operating parameters for air conditioning systems (airflow rate, cooling capacity, system efficiency, and refrigerant mass flow) in a way that is well-suited for the residential sector. The method relies on an air-side capacity estimate and will be compared and contrasted with a more traditional method that relies on a refrigerant-side capacity estimate. These methods are compared in terms of their sensitivity to input parameters, their uncertainty in the outputs, and their sensor requirements. The proposed air-side sensing method requires fewer sensors and has a significant advantage for residential split systems because it requires no outdoor unit sensors. The air-side sensing method is then successfully implemented on a field operating residential system. The experimental results show that the proposed method is sensitive to artificially inserted airflow faults.

4.2 Introduction

Air conditioning and space heating account for 32% of total energy consumption [3] and 21% of electrical energy consumption in the U.S. residential sector [93]. The most significant sources of energy inefficiency for residential heating, ventilation and air conditioning (HVAC) include air leaking from ducts, too little or too much refrigerant in the system (undercharge and overcharge), having a system that is oversized with respect to the required load, and having ductwork that is undersized [53]. Several studies have quantified both the prevalence and the impact of these efficiencies, and only a few of these studies will be cited here. [10] studied installation problems in residential systems and found that about 75% of central air conditioners and heat pumps have incorrect refrigerant charge, and about 70% have inadequate airflow. [14] later studied the impact

[1]Rogers, A.P., and F. Guo, and B.P. Rasmussen. Uncertainty Analysis and Field Implementation of a Fault Detection and Diagnosis Method for Residential HVAC Systems. **In Review**.

that refrigerant charge has on system efficiency and found that refrigerant charge 25% below the nominal reduces seasonal efficiency by approximately 16%. [53] also studied installation problems and found that indoor airflow 36% below the nominal increases energy usage by 11 - 14%. These studies show that installed air conditioners often do not reach the efficiency that they are designed for.

While the design efficiency of both heating and cooling systems continues to improve, there is a great need for improving the commissioning and maintenance process for these systems in order to realize the full benefits of installing a more efficient system. Fault detection and diagnostics (FDD) have become an increasingly popular tool for improving the commissioning and maintenance of commercial HVAC systems [18]. However, the state-of-the-art FDD tools depend heavily on the data available through the building automation system (BAS), and this detailed data is not typically available in the residential sector. FDD for residential split systems has not achieved an appreciable level of market penetration because the proposed methods are not yet cost effective. Air conditioning FDD methods propose that the compressor, the condenser, and the evaporator all be instrumented thus requiring many sensors. This is especially problematic for split systems where the indoor and outdoor units may be separated by a great distance. The required sensing package for FDD must be reduced and simplified in order for FDD to succeed in the residential air conditioning market, and this paper proposes a method that both reduces the number of sensors and simplifies the installation of the sensors.

4.2.1 Review of Fault Detection and Diagnosis Literature for Vapor Compression Systems

A variety of methods have been explored for FDD in vapor compression HVAC systems [5, 6]. The scope of modeling methods is also broad; statistics-based [37], simple physics-based [23], and purely data-driven methods such as support vector machines [29] and clustering [30] have all been used. The majority of FDD methods are explored using data from lab experiments in which faults are introduced in a careful, controlled manner. The physics-based decoupling methods presented in [23] are among the few methods actually applied to a system operating in the field. Initial testing of the decoupling methods was performed with data from [45], and then the methods were tested in the field in [51]. Similar decoupling methods were applied to a field unit in [24]. The virtual sensors proposed in [23] used a compressor energy balance to obtain the refrigerant mass flow and then estimated the indoor airflow using a refrigerant-side capacity estimate.

4.2.2 Overview

This paper builds on these simple physics-based FDD solutions in order to address challenges specific to residential split systems. Minimizing the up-front costs of an FDD system is of particular importance in the residential sector because residential systems are typically smaller than their commercial counterparts, and therefore have a lower capital cost and less energy expenditures. At a basic level, the up-front cost of the FDD sensors is affected by three factors: quantity, type, and location. The relationship between the quantity of sensors and capital cost is straightforward and this is usually the first factor considered. Of course, the type of sensor is also important. While a refrigerant mass flow sensor would certainly be useful for the purposes of FDD, its cost is substantially higher than the cost of multiple temperature sensors combined. The third factor is the sensor location, and this is not typically considered when designing an FDD system. For residential split systems, the indoor and outdoor units are typically separated by a great distance, and the outdoor unit is placed in an outdoor environment; any sensor on the outdoor unit must be able to withstand the inevitably harsh conditions. Furthermore, installing sensors on both the indoor unit and the outdoor unit may require a separate data acquisition board on each unit. The sensor location is certainly an important factor for determining the cost of a residential FDD system, but this factor is not typically considered because most FDD methods are designed for packaged commercial systems.

This paper proposes a set of virtual sensors for estimating the refrigerant mass flow rate, airflow rate, cooling capacity, and system efficiency using sensors only located on the indoor unit. This set of virtual sensors is based on an air-side capacity estimate and will be referred to as the Air Sensing Method. This method is contrasted with a set of virtual sensors that obtain the same parameters using a refrigerant-side capacity estimate: referred to as the Refrigerant Sensing Method. The Refrigerant Sensing Method is similar to the approach used in [23] in that it uses a compressor energy balance to obtain the refrigerant mass flow rate and then estimates airflow using a refrigerant-side cooling capacity estimate. The Air Sensing Method obtains the airflow rate using a method similar to that presented in [40, 44], and uses the estimated airflow to estimate the refrigerant mass flow rate using an air-side capacity estimate. As these methods are presented in greater detail, the key difference between them will become clear; the Refrigerant Sensing Method requires a compressor energy balance and thus requires outdoor sensors while the Air Sensing Method requires a reliable heating source and eliminates the need for outdoor sensors.

4.2.3 A Brief Introduction to Virtual Sensor Block Diagrams

Before proceeding with the formulation of the various virtual sensors, some context should be provided for the way in which these virtual sensors will be presented. Virtual sensors used in air conditioning FDD methods are typically physics-based and employ many assumptions and approximations. The required measurements, estimates, and assumptions can become nebulous as the estimation process becomes layered in equations. This paper will use a block diagram representation for virtual sensors in order to clearly show the full scope of each method. This block diagram representation is used in Figures 4.2 - 4.6, and each block diagram uses the notation given in Figure 4.1.

Figure 4.1 provides a legend for the most common operations and shadings used in the virtual sensor block diagrams. This legend shows the notation used for the majority of operations which include addition, multiplication, and refrigerant property tables. The legend also shows that the virtual sensor inputs can be either measured or estimated.

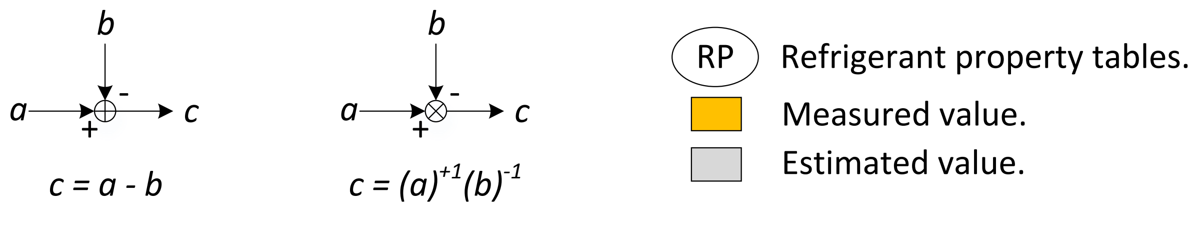


Figure 4.1: Legend for the virtual sensor block diagrams.

4.3 The Refrigerant Sensing Method

The cooling capacity is the amount of heat removed from the conditioned space. This is equivalent to the heat transfer that occurs at the indoor coil. This can be approached from either the refrigerant side or the air side. From the refrigerant side, the cooling capacity is given by Equation 4.1.

$$Q = \dot{m}_r \Delta h_{evap} \tag{4.1}$$

The original decoupling method [23] is based on a refrigerant-side capacity estimate. The decoupling method first determines the refrigerant mass flow using a compressor energy balance, the mass flow is then used to estimate cooling capacity from the refrigerant side, and the cooling capacity is then used to obtain the indoor airflow.

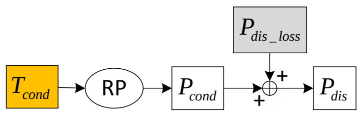
4.3.1 Refrigerant Mass Flow using a Compressor Energy Balance

Consider the refrigerant mass flow virtual sensor presented in [23]. A simple, physics-based energy balance of the compressor reveals that the refrigerant mass flow is the following.

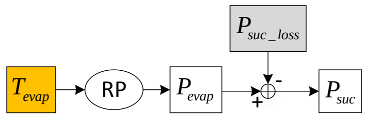
$$\dot{m}_r = \frac{\dot{W}_{comp} - Q_{loss}}{\Delta h_{comp}} \tag{4.2}$$

The compressor heat loss Q_{loss} can be approximated as $\alpha_{loss}\dot{W}_{comp}$. However, the proposed method for obtaining the compressor power is to use the manufacturer-provided 10-coefficient compressor map following AHRI Standard 540 [94]. Furthermore, the proposed method for obtaining the suction or discharge pressures using only temperature measurements is to measure the condensing and evaporating temperatures, convert to pressure, and then account for the pressure loss in the suction and discharge lines [95]. The full process of estimation is difficult to fully grasp using only equations, but each equation may be represented as a block diagram (Figure 4.2) and the block diagrams may be combined to show the full estimation process (Figure 4.3).

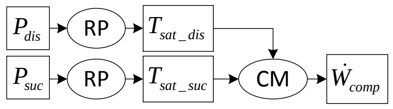
Figure 4.2 shows a block diagram for each equation required for the virtual mass flow sensor presented in [23] including the virtual pressure sensors. These sub-diagrams are combined in



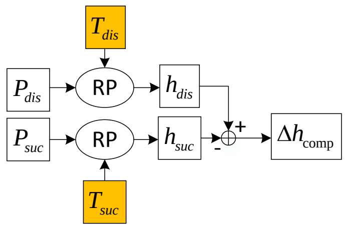
(a) $P_{dis} = P_{cond}(T_{cond}) + P_{dis\_loss}$



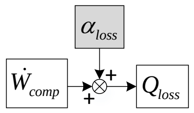
(b) $P_{suc} = P_{evap}(T_{evap}) - P_{suc\_loss}$



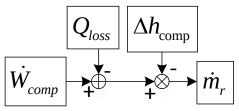
(c) $\dot{W}_{comp} = \mathrm{CM}(T_{sat\_suc}(P_{suc}), T_{sat\_dis}(P_{dis}))$



(d) $\Delta h_{comp} = h_{dis}(P_{dis}, T_{dis}) - h_{suc}(P_{suc}, T_{suc})$



(e) $Q_{loss} = \alpha_{loss}\dot{W}_{comp}$



(f) $\dot{m}_r = \frac{\dot{W}_{comp} - Q_{loss}}{\Delta h_{comp}}$

Figure 4.2: Sub-diagrams comprising the virtual mass flow sensor using the manufacturer's compressor map (CM) and virtual pressure sensors.

Figure 4.3. The full virtual sensor block diagram clearly shows that, in order to implement this virtual sensor, the evaporating, condensing, suction, and discharge temperatures must be measured. The pressure loss in the suction line and the pressure loss in the discharge line must be estimated as well as the compressor heat loss coefficient α_{loss}.

Breaking the estimation process down into the basic operations that comprise the virtual sensor provides a convenient means to analyze the sensitivity and uncertainty in the virtual sensor at a particular operating point. However, the sensitivity and uncertainty will be addressed in Section 4.5 after both the Refrigerant Sensing Method and the Air Sensing Method have been fully developed in order to directly compare these methods.

4.3.2 Modifying the Virtual Refrigerant Mass Flow Sensor

Figure 4.3 shows the virtual refrigerant mass flow sensor as presented in [23]. However, this virtual sensor requires the 10-coefficient compressor models provided by the manufacturer. These

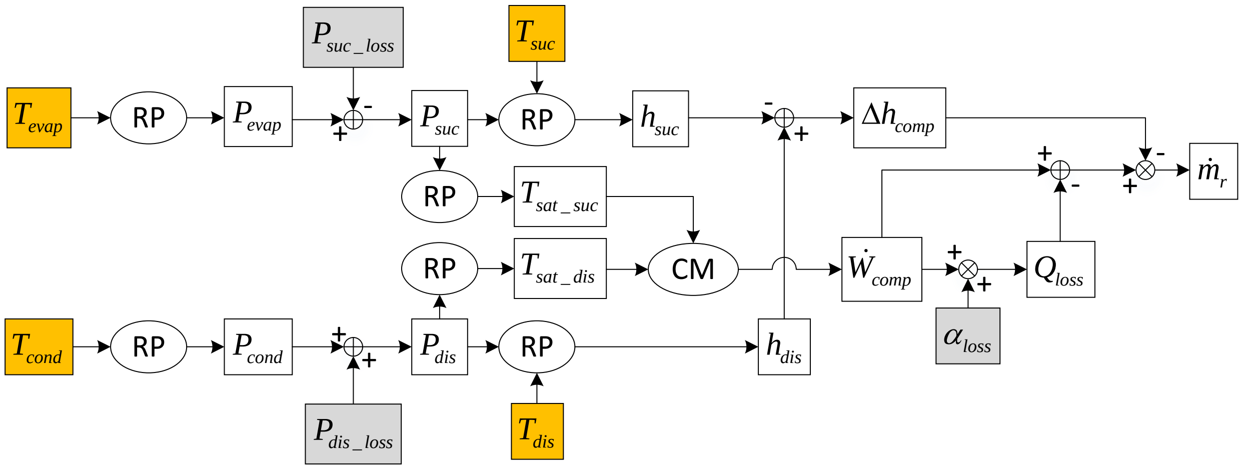


Figure 4.3: Full virtual mass flow sensor using compressor map (CM) to obtain the compressor power.

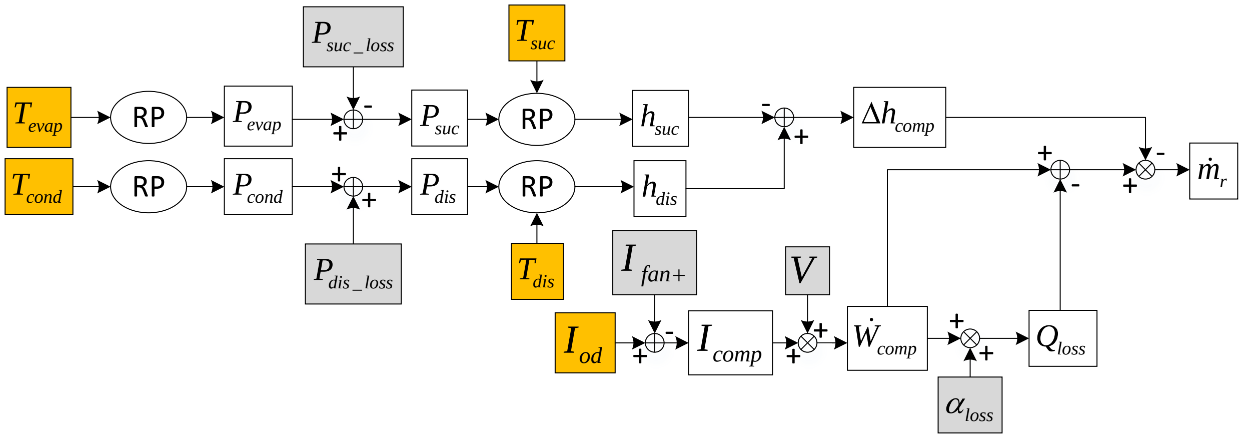


Figure 4.4: Full virtual mass flow sensor using a current measurement to obtain the compressor power.

models may not always be available, and this is especially true for residential systems which use smaller compressors. Indeed, the 10-coefficient compressor models were not available for the field-installed 3.5 ton (12.3 kW) packaged heat pump used for testing in this study, and the compressor power needed be estimated using a current measurement and an estimated voltage. However, the isolated compressor current would require a current transducer installed on the outdoor unit, and therefore, the current for the full outdoor unit was measured because this can be measured with a transducer installed inside the home. When the compressor is running, the outdoor unit power is predominantly due to the compressor. However, the compressor power can be estimated more accurately by subtracting the estimated current of the outdoor fan and other components I_{fan+} from the measured outdoor unit current. The fan power is typically an order of magnitude less than the compressor power such that the uncertainty in the fan power estimate will negligibly affect the accuracy of the compressor power estimate; this is verified in the sensitivity analysis presented in Section 4.5.

4.3.3 Refrigerant-Side Capacity, Airflow, and Efficiency

Having estimated the refrigerant mass flow using Figure 4.4, the cooling capacity may be estimated using the enthalpy rise of the refrigerant in the evaporator as shown in Equation 4.1. The evaporator enthalpy rise is the difference between the evaporator exit (vapor line) enthalpy and the expansion device inlet (indoor liquid line) enthalpy. Note that the expansion device may be treated as an isenthalpic device such that there is no change in enthalpy as the refrigerant expands. The refrigerant temperature should be measured at these two locations and the pressure at these two locations can be estimated using a measured coil temperature and the estimated pressure drops. This is shown in the upper shaded region of Figure 4.5. While the estimated pressure drop across the expansion device will likely have a high degree of uncertainty, the enthalpy of the saturated refrigerant at this point will not be sensitive to the pressure drop; i.e. the enthalpy of a saturated liquid is nearly independent of pressure.

Once the cooling capacity is obtained, the indoor airflow rate may be determined using the simple energy balance given in Equation 4.3.

$$\dot{V}_{air} \frac{1}{\rho_{air}} \frac{Q}{\Delta h_{air\_wet}} \tag{4.3}$$

However, estimating the wet air enthalpy drop Δh_{air_wet} is not a simple process. The return and supply air temperature should be measured in addition to the return air relative humidity. The supply air humidity ratio can be assumed to be saturated when the coil is wet, and equal to the return air humidity when the coil is dry. In order to obtain the humidity ratio, the saturation

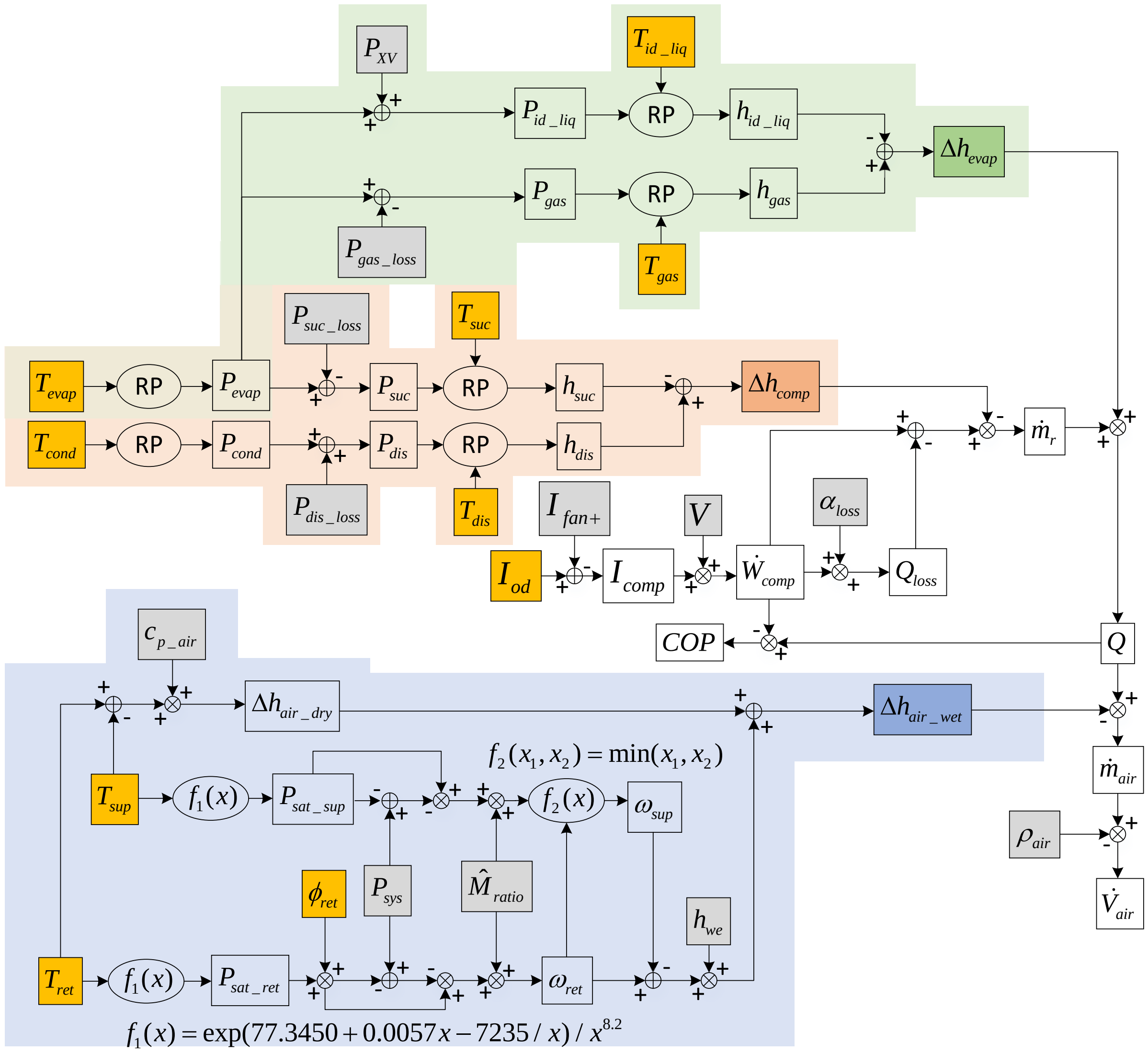


Figure 4.5: The Refrigerant Sensing Method

pressure must be estimated from the dry-bulb temperature. This full process is shown in the lower shaded region of Figure 4.5.

Figure 4.5 shows the full set of virtual sensors comprising the Refrigerant Sensing Method. The majority of work in these virtual sensors is dedicated to performing the enthalpy analysis of the compressor, evaporator refrigerant, and evaporator air. Having obtained Δh_{comp}, Δh_{evap}, and Δh_{air_wet}, Figure 4.5 highlights how straightforward several of the major system parameters can be estimated. The refrigerant mass flow is obtained using Equation 4.2, the capacity is obtained using Equation 4.1, the airflow is obtained using Equation 4.3, and the efficiency (as measured by the Coefficient of Performance or COP) is obtained using Equation 4.4.

$$COP = \frac{Q}{\dot{W}_{comp}} \tag{4.4}$$

The Refrigerant Sensing Method requires eight temperature sensors, one current sensor, and one relative humidity sensors. Three of the eight temperature sensors must be located on the outdoor unit. These sensor requirements will be compared with the Air Sensing Method after this second method is introduced.

4.4 The Air Sensing Method

As defined before, the cooling capacity is the amount of heat removed from the conditioned space. In the previous section, this was calculated using the rise in enthalpy of the refrigerant in the indoor coil. Equivalently, the cooling capacity can be expressed in terms of the air enthalpy drop.

$$Q = \dot{m}_{air} \Delta h_{air\_wet} \tag{4.5}$$

The enthalpy drop may be estimated using the method presented previously (Figure 4.5). However, the air mass flow rate is not immediately available. To obtain this, the Air Sensing Method uses the virtual supply airflow meter presented in [40, 44].

4.4.1 Virtual Airflow Sensor

The virtual airflow sensor was originally proposed for packaged HVAC systems. The sensor essentially reverses Equation 4.5 by first estimating the capacity and then using the capacity estimate to determine the airflow. Estimating the cooling capacity using a refrigerant-side energy balance results in the method presented in Figure 4.5. However, [40] proposed that the cooling capacity be obtained using a model based on the manufacturer's rating data. The estimated airflow would therefore be inaccurate in the presence of any fault other than low airflow because a fault-free cooling capacity model is used. Fortunately, [40] also proposed that the virtual airflow sensor could be used in heating mode with a heating capacity model based on rating data. Low airflow would therefore be coupled with heating mode faults but decoupled with cooling mode faults. This is a significant advantage when a reliable heating source such as a gas furnace or electric resistance heater is used. There are three advantages (listed below) of using the virtual airflow sensor in heating mode as opposed to cooling mode. Note that these advantages apply to using a gas furnace or an electric resistance heater; a heat pump operating in heating mode will have the same complexity and disadvantages as the cooling mode.

1. The rating data will predict the heating capacity more accurately than the cooling capacity. Thus, even when there are no refrigerant-side faults, using the heating mode results in less uncertainty. This is partly because the gas and resistive electric heat have fewer driving conditions that vapor compression systems: the capacity is independent of the outdoor air temperature.

2. Heating mode is more robust than the cooling mode. Furthermore, a heating mode fault will likely results in a complete loss of heating rather than degraded performance.

3. Air humidity levels do not need to be considered in heating mode. This greatly reduces the uncertainty in the airflow estimate.

The heating mode virtual sensor is summarized in Equation 4.6.

$$\dot{V}_{air} = \frac{1}{\rho_{air}} \frac{Q_h}{C_{p\_air}} (T_{ret} - T_{sup}) \tag{4.6}$$

This airflow virtual sensor will require that heating mode be used at times when it would not normally be. In other words, the heating mode would need to be used even on hot days in order to estimate the airflow rate (otherwise airflow won't be estimated in the summer). This method is therefore a proactive FDD method, and this method results in more energy consumption than passive methods. Proactive FDD methods have been proposed for sensor and actuator faults [90, 91], but not for the application described here. The virtual sensor was presented as a passive method in [40], but its adaptation as a proactive sensor is a key step for improving the cost-effectiveness of residential FDD. Implementation of the proactive strategy will be demonstrated in Section 4.6.

4.4.2 Air-Side Capacity, Efficiency, and Refrigerant Mass Flow

Having obtained an estimate of the airflow rate using the heating mode virtual sensor (Equation 4.6), the cooling capacity can be estimated using Equation 4.5. The system must therefore operate in cooling mode, and the air enthalpy drop is calculated as before. The system efficiency (COP) is then determined using Equation 4.4, and the refrigerant mass flow rate may be determined using Equation 4.7. The full set of virtual sensor is presented using a block diagram in Figure 4.6.

$$\dot{m}_r = \frac{Q}{\Delta h_{evap}} \tag{4.7}$$

Much of the estimation process using an air-side capacity (Figure 4.6) is similar to the estimation process using a refrigerant-side capacity (Figure 4.5). The methods use the same air enthalpy analysis, the same evaporator refrigerant enthalpy analysis, and the same estimation for compressor power. However, Figure 4.6 does not include a compressor enthalpy analysis and therefore eliminates three outdoor unit sensors: T_{cond}, T_{suc}, and T_{dis}. Using the heating-mode virtual airflow sensor therefore allows the airflow, capacity, efficiency, and refrigerant mass flow to be estimated using only indoor sensors. This is a significant advantage over previously proposed methods. In fact, the evaporator coil measurements (T_{evap}, T_{gas}, and T_{id_liq}) are used only for the refrigerant mass flow in Figure 4.6. The airflow, capacity, and efficiency can be determined using only four sensors: two air temperatures, the return air humidity, and the outdoor unit current.

4.5 Comparing the Refrigerant and Air Sensing Methods

The Refrigerant Sensing Method and the Air Sensing Method were compared in terms of their sensitivity to the inputs, the uncertainty in the outputs, and their sensor requirements. Note that the uncertainty and sensitivity will vary slightly as the operating point changes. The virtual sensor block diagrams presented in Figure 4.5 and Figure 4.6 are well suited for the uncertainty and sensitivity analysis. The block diagrams highlight the elementary operations that compose the virtual sensors, and the uncertainty ÏČ in each of these operations can be analyzed individually. For example, the uncertainty of the gas line enthalpy h_{gas} for both methods is given by Equation 4.8.

$$\sigma_{h_{gas}} = \sqrt{\left(\frac{\partial h_{gas}}{\partial P_{gas}}\right)\Bigg|^2_{P_{gas},T_{gas}} \sigma^2_{P_{gas}} + \left(\frac{\partial h_{gas}}{\partial T_{gas}}\right)\Bigg|^2_{P_{gas},T_{gas}} \sigma^2_{T_{gas}}} \tag{4.8}$$

For both methods, temperature sensors are assumed to have 2% uncertainty with respect to the ÂřF scale, and very large uncertainty values (50 or 60%) are used for the pressure loss estimates.

In addition to the propagation of uncertainty analysis, the sensitivity of the output variables to the input variables was analyzed. This sensitivity is defined as the partial derivative of the output with respect to the input. For example, the sensitivity of the refrigerant mass flow $\dot{m}_r$ to the suction

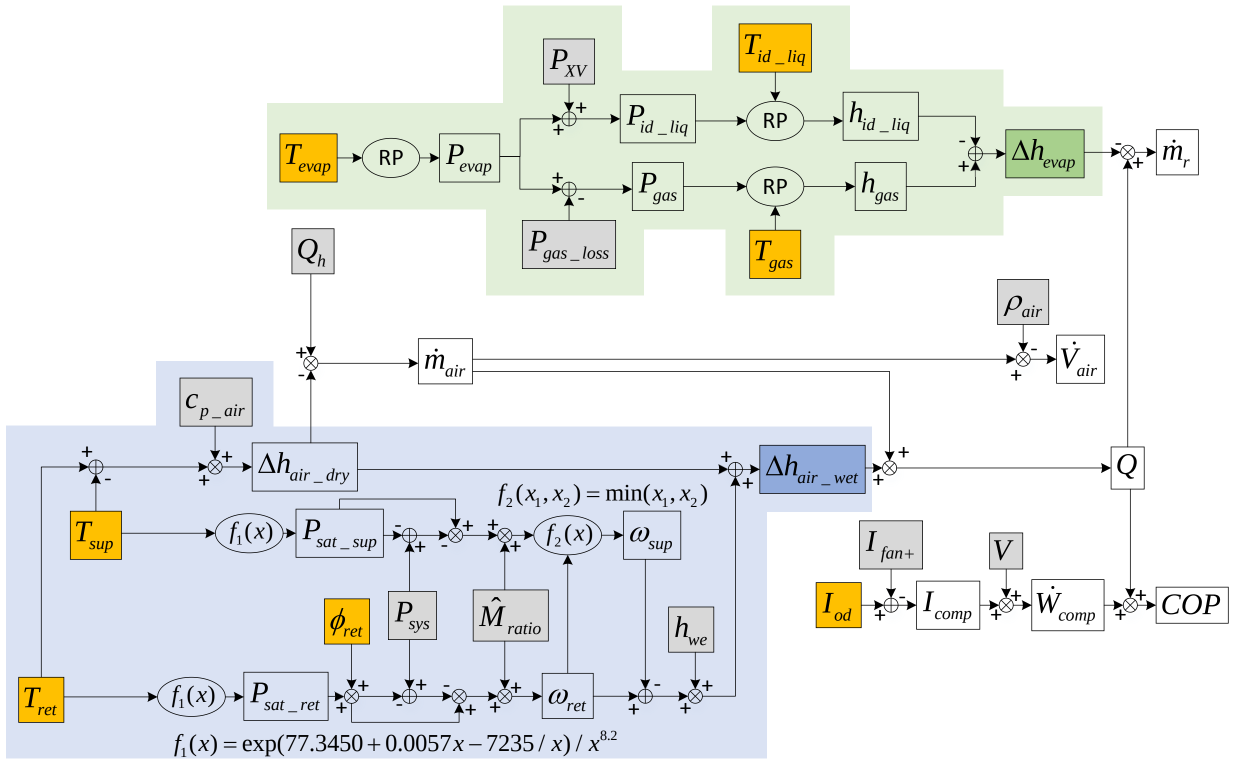


Figure 4.6: The Air Sensing Method

line pressure loss P_{suc_loss} in Figure 4.5 is given by Equation 4.9.

$$\frac{\partial \dot{m}_r}{\partial P_{suc\_loss}} = \left(\frac{\partial \dot{m}_r}{\partial \Delta h_{comp}}\right)\left(\frac{\partial \Delta h_{comp}}{\partial h_{suc}}\right)\left(\frac{\partial h_{suc}}{\partial P_{suc}}\right)\left(\frac{\partial P_{suc}}{\partial P_{suc\_loss}}\right) \tag{4.9}$$

This process is complicated only slightly when an input affects the output on two paths. For example, the sensitivity of the evaporator enthalpy rise to the evaporating temperature is given by Equation 4.10.

$$\frac{\partial \Delta h_{evap}}{\partial T_{evap}} = \left[\left(\frac{\partial \Delta h_{evap}}{\partial h_{gas}}\right)\left(\frac{\partial h_{gas}}{\partial P_{gas}}\right)\left(\frac{\partial P_{gas}}{\partial P_{evap}}\right) + \left(\frac{\partial \Delta h_{evap}}{\partial h_{id\_liq}}\right)\left(\frac{\partial h_{id\_liq}}{\partial P_{id\_liq}}\right)\left(\frac{\partial P_{id\_liq}}{\partial P_{evap}}\right)\right]\left(\frac{\partial P_{evap}}{\partial T_{evap}}\right) \tag{4.10}$$

The resulting partial derivative represents how the evaporator enthalpy rise is affected by the evaporating temperature at the current operating condition. However, interpreting this result on its own is difficult because the properties may have significantly different magnitudes. The sensitivity should be normalized relative to the magnitude such that the sensitivity represents the percent increase in the output provided a percent increase in the input (Equation 4.11).

$$\frac{\partial(\frac{A}{|A|})}{\partial(\frac{B}{|B|})} = \frac{\partial A}{\partial B}\frac{|B|}{|A|} \tag{4.11}$$

Table 4.1: Uncertainty in the Input Parameters and Sensitivity of the Outputs at an Example Operating Point for the Refrigerant Sensing Method and the Air Sensing Method

			Sensitivity of Outputs (% increase in output given a 1% increase in input[‡]; sensitivities > 1 are shaded)							
			Refrigerant Sensing Method				Air Sensing Method			
Input	Operating Point	Uncertainty[†] (%)	$\dot{V}_{air}$	Q	COP	$\dot{m}_r$	$\dot{V}_{air}$	Q	COP	$\dot{m}_r$
T_{ret}*	72 °F (22.2 °C)	2	-5.65					5.65	5.65	5.65
T_{sup}*	52 °F (11.1 °C)	2	3.45					-3.45	-3.45	-3.45
ϕ_{ret}*	0.7	5	-1.49					1.49	1.49	1.49
T_{evap}*	50 °F (10 °C)	2	-0.65	-0.65	-0.65	-0.51				0.14
T_{gas}*	50 °F (10 °C)	2	0.18	0.18	0.18					-0.18
T_{id_liq}*	75 °F (23.9 °C)	2	-0.37	-0.37	-0.37					0.37
I_{od}*	13.5 A	5	0	1.04	0.00	1.04			-1.04	
c_{p_air}	$0.24\frac{\text{Btu}}{\text{lbm°F}}$ $(1.01\frac{\text{kJ}}{\text{kg°C}})$	1	-0.56				-1	-0.44	-0.44	-0.44
ρ_{air}	$0.076\frac{\text{lbm}}{\text{ft}^3}$	3	-1				-1			
P_{sys}	14.7 psia (101.3 kPa)	2	0.45					-0.45	-0.45	-0.45
$\hat{M}_{ratio}$	0.622	1	-0.44					0.44	0.44	0.44
h_{we}	$1061\frac{\text{Btu}}{\text{lbm}}$ $(2501\frac{\text{kJ}}{\text{kg}})$	1	-0.44					0.44	0.44	0.44
I_{fan+}	0.5 A	50	-0.04	-0.04		-0.04			0.04	
V	240 V	5	1	1		1			-1	
P_{XV}	250 psia (1724 kPa)	50	0.00	0.00	0.00					-0.00
P_{gas_loss}	5 psia (34.5 kPa)	60	0.01	0.01	0.01					-0.01
P_{dis_loss}	10 psia (68.9 kPa)	60	0.02	0.02	0.02	0.02				
P_{suc_loss}	15 psia (103 kPa)	60	0.06	0.06	0.06	0.06				
α	0.05	60	-0.05	-0.05	-0.05	-0.05				
T_{cond}*	130 °F (54.4 °C)	2	1.75	1.75	1.75	1.75				
T_{dis}*	195 °F (90.6 °C)	2	-3.19	-3.19	-3.19	-3.19				
T_{suc}*	60 °F (15.6 °C)	2	0.81	0.81	0.81	0.81				
T_{ret} (heat)*	72 °F (22.2 °C)	2					2.52	2.52	2.52	2.52
T_{sup} (heat)*	100.6 °F (38.1 °C)	2					-3.52	-3.52	-3.52	-3.52
Q_h	35.7 MBh (10.5 kW)	5					1	1	1	1

* Measured value.

[†] Uncertainties in temperature measurements are given with respect to the °F scale.

[‡] Sensitivities to temperature inputs are given as a % increase given a 1% increase on the °F scale.

Table 4.2: Uncertainty in the Output Parameters at an Example Operating Point and the Sensor Requirements for the Refrigerant Sensing Method (RSM) and the Air Sensing Method (ASM)

				Sensor Requirements*							
		Uncertainty (%)		RSM				ASM			
Output	Operating Point	RSM	ASM	T_{id}	T_{od}	RH	I	T_{id}	T_{od}	RH	I
$\dot{V}_{air}$	1133 cfm (1925 $\frac{m^3}{hr}$)	17.3	10.5	5	3	1	1	2			
Q	44.3 MBh (13.0 kW)	11.4	16.2	3	3		1	2		1	
COP	4.16	13.6	17.8	3	3		1	2		1	1
$\dot{m}_r$	549$\frac{lbm}{hr}$ (249$\frac{kg}{hr}$)	11.4	16.2	1	3		1	5		1	1

*T_{id}: Temperature sensor located on the indoor unit.

T_{od}: Temperature sensor located on the outdoor unit.

RH: Relative humidity sensor.

I: Electric current sensor.

Table 4.1 includes the operating point of the input parameters (measurements and estimates), the estimated uncertainty in the input parameters, and the sensitivity of each output to each input for both methods. Table 4.2 includes the operating point of the output parameters, the uncertainty in these outputs for each method, and the sensor requirements for each output and for each method. The sensitivities presented in Table 4.1 are relative to the magnitude of both the input and output (Equation 4.11) such that they represent the percent increase in the output given a percent increase in the input. The first row shows that, if the return air temperature T_{ret} were to increase by 1% on the Fahrenheit scale, then the estimated airflow rate using the Refrigerant Sensing Method will decrease by 5.65%. In other words, if the return air temperature were to rise from 72 °F (22.2 °C) to 72.7 °F (22.6 °C) with all other inputs remaining unchanged, then the estimated airflow rate using the Refrigerant Sensing Method would decrease from 1133 cfm (1925 m^3/hr) to 1069 cfm (1816 m^3/hr).

While much can be learned from this comparison of methods, only a few points will be highlighted here.

- The outputs for both methods are not sensitive to the pressure losses. There are two reasons for this. Firstly, while the pressure loss across the expansion device is quite large, this pressure loss is used to obtain the pressure of a saturated liquid (indoor liquid line) which in turn is used to obtain the liquid enthalpy h_{id_liq}. Since the enthalpy of a saturated liquid is nearly independent of pressure, the high uncertainty in the expansion device's pressure drop does not affect the output uncertainty. Secondly, while the saturated liquid argument is not applicable to the other pressure losses (P_{gas_loss}, P_{dis_loss}, and P_{suc_loss}), these pressure losses are small relative to the magnitude of the pressure at these points. The 60% uncertainty in P_{gas_loss} results in an uncertainty of 3 psi (20 kPa), and the estimated gas line pressure is 152 psi (1048 kPa). Therefore, the 60% uncertainty results in less than 2% uncertainty in the gas line pressure.

- With the exception of the airflow, the Refrigerant Sensing Method has lower uncertainty than the Air Sensing Method. This is partly because both methods are sensitive to the air measurements (T_{sup}, T_{ret}, and ϕ_{ret}), and all estimates in the Air Sensing Method use these measurements. The Refrigerant Sensing Method only uses the air measurements to estimate the airflow.

- The Air Sensing Method has a significant advantage over the Refrigerant Sensing Method in terms of sensor requirements. In order to perform all four estimates, the Refrigerant Sensing Method requires ten sensors and the Air Sensing Method requires seven sensors. Furthermore, the three sensors that the Air Sensing Method removes are the three outdoor unit sensors such that this method does not require any sensors on the outdoor unit. The Air Sensing Method increases its advantage if all four measurements are not necessary; the airflow and capacity are arguably the most important estimates, and the Air Sensing Method uses only three sensors to obtain these. The Refrigerant Sensing Method requires seven sensors to estimate the capacity and all ten to estimate the airflow. However, if the refrigerant mass flow estimate is critical, then the Refrigerant Sensing Method has the advantage; it requires only 5 sensors to obtain the refrigerant mass flow and the resulting uncertainty is significantly lower than that of the Air Sensing Method.

4.6 Field Implementation of the Air Sensing Method

The virtual sensors presented in Figure 4.6 were implemented on a 3.5 ton (12.3 kW) heat pump operating in the field. The full implementation first requires that a known heating capacity be applied in order to measure airflow and then the cooling capacity is applied in order to measure capacity, efficiency, and mass flow. The following experimental results show the steady state estimates after a steady state filter is applied. Indoor airflow restriction is applied for the heating tests in order to show how the virtual airflow sensor is sensitive to airflow faults, and condenser restriction is applied for the cooling tests in order to show how the other virtual sensors are sensitive to this particular fault. As this unit is operating in field, other faults (e.g. compressor degradation and incorrect charge) were not artificially inserted. This field implementation provides proof of concept that the proposed virtual sensors are effective for fault detection.

4.6.1 Implementation of Virtual Airflow Sensor

The virtual airflow sensor presented in Figure 4.6 requires a known heating capacity. The field-operating heat pump unit has a 10 kW electric heater installed, and this electric heat source provides a relatively predictable heat capacity. When implemented, the airflow sensor must be sensitive to reductions in airflow caused by filter and coil fouling, crushed ductwork, and closed supply registers. Note that the air temperature sensors measure the supply and return air temperatures at a single point in the duct and may therefore require calibration in order to measure the true airflow rate. However, even without calibration, the virtual sensor will still be sensitive to changes in airflow rate that will be reflected in the estimate.

Figure 4.7 shows the estimated airflow of the field-installed heat pump with several levels of restriction applied. The filter area is presented relative to the main supply and return duct cross-sectional area. The airflow rate remains relatively constant when the filter area is much larger than the duct's area. However, there is a significant reduction in airflow when the filter area is about the same size as the duct's area, and the airflow continues to decrease as the filter area is restricted further. Figure 4.7 presents the results of two experiments at each restriction level, and the resulting estimate in airflow is highly repeatable. The accuracy of this estimate must be validated with a ground-truth airflow measurement and the virtual sensor should be calibrated accordingly. However, the estimated airflow with no restriction is near the systemâĂŹs nominal airflow rate even without calibration.

4.6.2 Implementation of Virtual Cooling Capacity Sensor

Having obtained an estimate for the airflow using a known heating capacity, the cooling capacity may be estimated. The Air Sensing Method (Figure 4.6) requires a return air humidity measurement in addition to the supply and return air temperatures for cooling capacity. For this implementation, the return air humidity ratio is obtained using the smart thermostat relative humidity and indoor temperature. This eliminates the need for an additional relative humidity sensor thus reducing the total number of required sensors to six (five temperature and one current). The

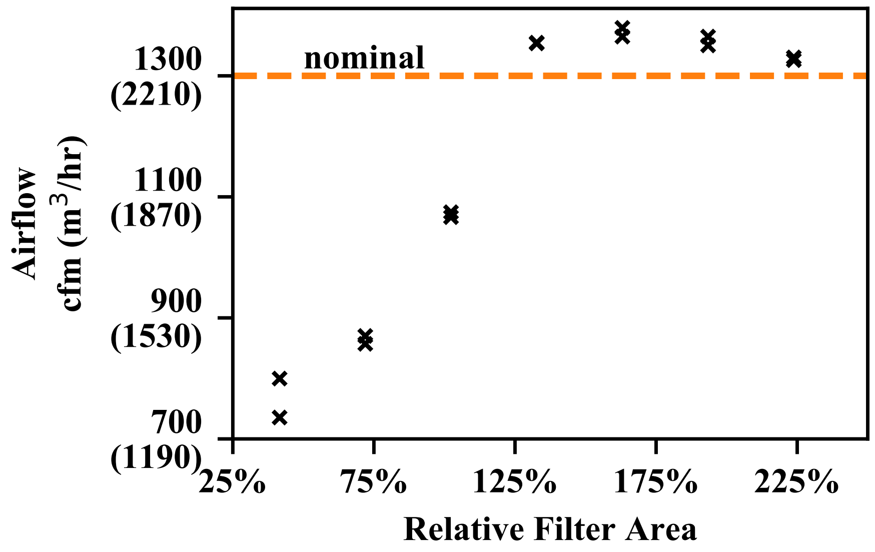


Figure 4.7: Implementation of the virtual airflow sensor using a new filter with artificially introduced filter restriction. The relative filter area denotes the ratio of exposed filter area to the return and supply duct cross-sectional area.

number of required sensors could be further reduced by using the thermostat indoor temperature measurement in place of a return air temperature measurement, but this will reduce the sensor accuracy significantly. Table 4.1 shows that, for the Air Sensing Method, the cooling capacity will increase by 1.49% for every percent increase in return air relative humidity, and the cooling capacity will increase by 5.65% for every percent increase in return air temperature (with respect to the °F scale). The virtual sensor is nearly four times more sensitive to the return air temperature than the return air humidity and therefore an accurate return air temperature measurement should be made.

Figure 4.8 shows the experimental results for the virtual cooling capacity sensor with four levels of condenser fouling applied. Four experiments are applied at each restriction level, and the virtual sensor clearly captures the degradation in cooling capacity as the available condenser area decreases. Some of the variation in the cooling capacity will be due to the variation in outdoor temperature from one test to the next. As this is a field operating unit, the outdoor temperature could not be held constant to more accurately assess the repeatability of this method. The outdoor temperature varied between 74 and 79 °F (23.3 and 26.1 °C) for the 16 tests shown in Figure 4.8.

4.6.3 Implementation of Virtual Efficiency Sensor

After obtaining the cooling capacity, only a current measurement is needed in order to estimate the system efficiency. Figure 4.9 shows the estimated system efficiency for the same 16 tests used to estimate the cooling capacity. These experiments capture the degradation in efficiency well, but this is expected as the efficiency is largely dependent on the cooling capacity. However, the efficiency is in fact more sensitive to the condenser fouling than the cooling capacity is. Recall that the efficiency is the cooling capacity divided by the compressor power such that a decrease in cooling capacity will likely result in a decrease in efficiency. The condenser fouling increases the compressor power and decreases the cooling capacity such that the condenser fouling has an even larger impact on the efficiency. A condenser area of 50% results in a degradation in cooling capacity of about 20% and a degradation in efficiency of about 35%.

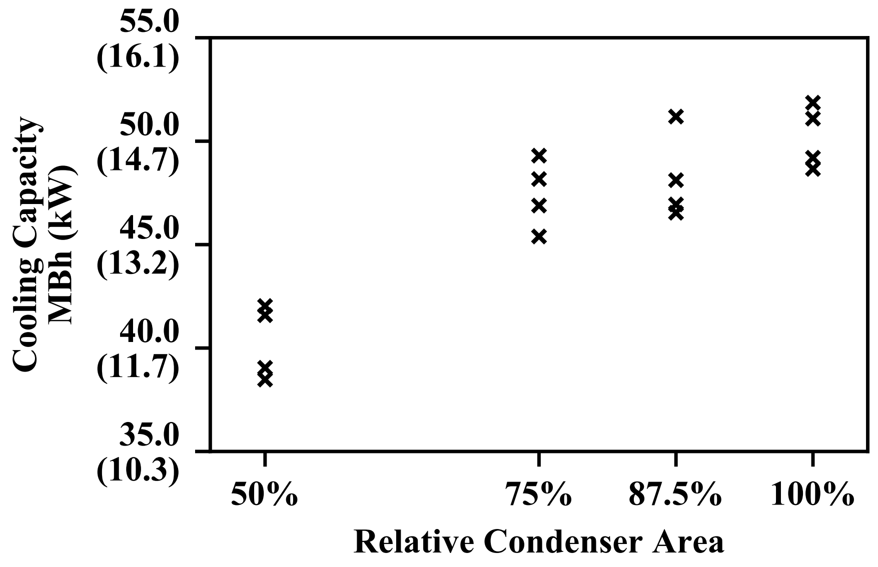


Figure 4.8: Implementation of the virtual cooling capacity sensor with artificially introduced condenser fouling. The relative condenser area is the ratio of unrestricted condenser area to the nominal condenser area.

4.6.4 Implementation of Virtual Refrigerant Mass Flow Sensor

Estimating the refrigerant mass flow requires three refrigerant temperatures (T_{evap}, T_{gas}, and T_{id_liq}) in order to obtain the rise in refrigerant enthalpy through the evaporator Δh_{evap}. Figure 4.10 shows the mass flow estimate for the same 16 cooling tests. The condenser fouling does not seem to affect the mass flow as strongly as it does the capacity and efficiency. Note that the mass flow cannot be measured accurately for a high level of condenser fouling; artificially introducing condenser fouling results in the refrigerant condensing in some branches of the condenser and remaining a gas in other branches of the condenser. The gas and liquid may not be fully mixed at the entrance of the indoor unit and the indoor liquid line measurement will have large variations. The mass flow measurement with extreme condenser fouling would be more accurate if the condenser fouling were applied more uniformly throughout the condenser.

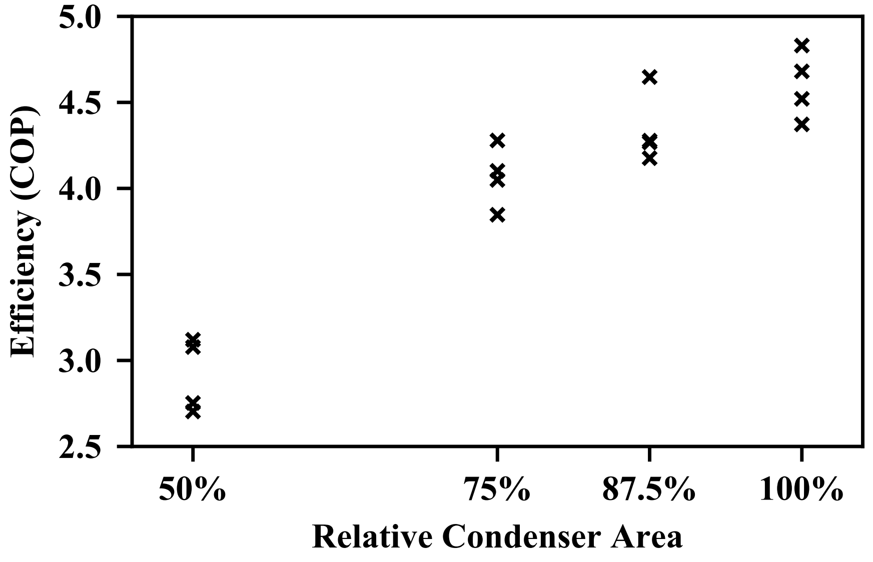


Figure 4.9: Implementation of the virtual efficiency sensor with artificially introduced condenser fouling. The relative condenser area is the ratio of unrestricted condenser area to the nominal condenser area.

4.7 Conclusions

The Air Sensing Method presented herein is a set of virtual sensors designed for fault detection and diagnosis for residential HVAC systems. The method addresses challenges specific to the residential market in that it uses fewer sensors than the Refrigerant Sensing Method and requires no sensors installed on the outdoor unit. The proposed set of virtual sensors uses a known heating capacity to obtain the airflow and then uses the estimated airflow to perform an air-side cooling capacity estimate. The airflow virtual sensor was inspired by [40], but in this earlier work, the authors were only concerned with airflow faults. The Air Sensing Method presented herein uses both the heating mode and the cooling mode separately to decouple air-side and refrigerant-side faults, and this process eliminates the need for a compressor energy balance as was required in [23]. Therefore the airflow, cooling capacity, system efficiency, and refrigerant mass flow may be estimated without sensors installed on the outdoor unit. These virtual sensors have been implemented on an air conditioning system operating in the field, and they successfully capture the

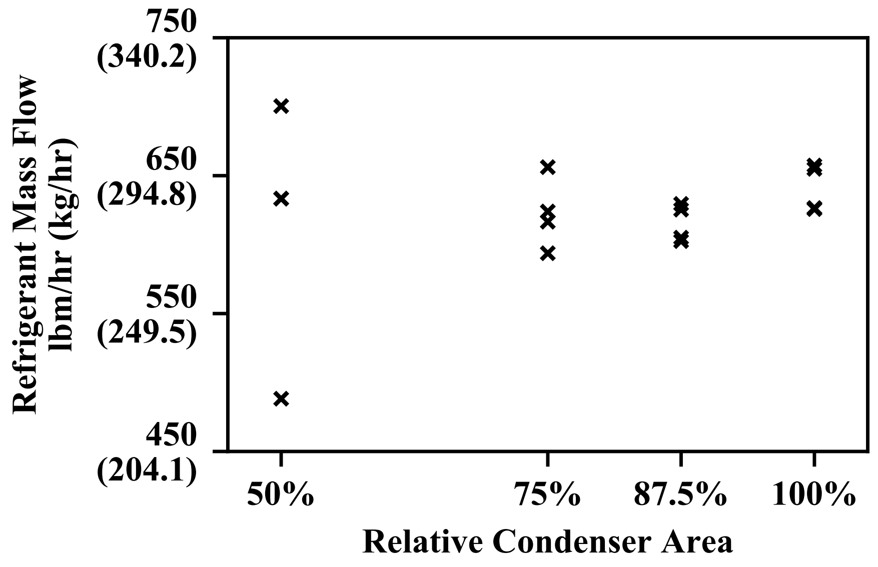


Figure 4.10: Implementation of the virtual refrigerant mass flow sensor with artificially introduced condenser fouling. The relative condenser area is the ratio of unrestricted condenser area to the nominal condenser area.

performance degradation as indoor and outdoor airflow faults are introduced.

This method may be further improved in several ways. First, the virtual sensors should be calibrated using ground truth knowledge; a laboratory calibration may be valid for field operating systems if the location of the installed sensors is standardized. Second, the method may be implemented as a proactive fault detection method by operating the system in heating or cooling at times when the system would not normally run. This proactive test method should be developed carefully and the trade-off with occupant comfort should be characterized. Third, the fault diagnosis capabilities of system-level parameters such as capacity, efficiency, and mass flow should be explored. The proposed method sufficiently decouples indoor airflow faults, but incorrect charge, compressor degradation, and condenser fouling will all result in reduced capacity and efficiency. Still, further decoupling may be possible by considering the magnitude of the effect that each fault has on the estimated system parameters.

4.8 Acknowledgment

This material is based upon work supported by Trane, a subsidiary of Ingersoll Rand. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Trane.

4.9 List of Nomenclature

α_{loss}	Portion of compressor power lost as heat.
Δh_{air_dry}	Enthalpy drop of dry air only (neglecting water vapor) across the evaporator.
Δh_{air_wet}	Enthalpy drop in air and water vapor mixture across the evaporator.
Δh_{comp}	Enthalpy rise across the compressor.
Δh_{evap}	Enthalpy rise of refrigerant across the evaporator.
ω_{ret}	Return air humidity ratio (absolute humidity): humidity ratio of air and water vapor mixture exiting the evaporator.
ω_{sup}	Supply air humidity ratio (absolute humidity): humidity ratio of air and water vapor mixture entering the evaporator.
ϕ_{ret}	Return air relative humidity: relative humidity of the air entering the evaporator.
ρ_{air}	Density of indoor air.
c_{p_air}	Specific heat at a constant pressure of air.
h_{dis}	Discharge enthalpy: enthalpy at the exit of the compressor.
h_{id_liq}	Indoor liquid line enthalpy: enthalpy before the expansion device.
h_{suc}	Suction enthalpy: enthalpy at the inlet of the compressor.
h_{we}	Heat of vaporization of water.
Q	Cooling capacity.
Q_h	Heating capacity from gas furnace or electric heat.
Q_{loss}	Heat loss of compressor.
COP	Coefficient of Performance.
I_{comp}	Electric current delivered to the compressor.

I_{fan+}	Electric current for all components in the outdoor unit except the compressor: primarily the outdoor fan.
I_{od}	Electric current delivered to the outdoor unit.
$\dot{m}_{air}$	Indoor air mass flow rate.
$\dot{m}_r$	Refrigerant mass flow rate.
$\hat{M}_{ratio}$	Ratio of molar mass of water to molar mass of air.
T_{cond}	Condensing temperature measured from the middle of the condenser.
T_{dis}	Discharge temperature: temperature at the exit of the compressor.
T_{evap}	Evaporating temperature measured from the beginning of the evaporator.
T_{gas}	Gas line temperature: temperature at the exit of the evaporator.
T_{id_liq}	Indoor liquid line temperature: temperature before the expansion device.
T_{ret}	Return air temperature: temperature of air entering the evaporator.
T_{sat_dis}	Discharge saturation temperature: temperature at which the compressor exit would be saturated.
T_{sat_suc}	Suction saturation temperature: temperature at which the compressor inlet would be saturated.
T_{suc}	Suction temperature: temperature at the inlet of the compressor.
T_{sup}	Supply air temperature: temperature of the air exiting the evaporator.
P_{cond}	Condensing pressure at point of measurement.
P_{evap}	Evaporating pressure at the point of measurement.
P_{dis}	Discharge pressure: pressure at the exit of the compressor.
P_{dis_loss}	Pressure drop in the discharge line: from the discharge pressure to the condensing pressure.
P_{gas}	Gas line pressure: pressure at the exit of the evaporator.
P_{gas_loss}	Pressure drop through the evaporator: from the evaporating pressure to the gas line pressure.

P_{sat_ret}	Return air saturation pressure: pressure at which the water vapor of the air entering the evaporator is saturated.
P_{sat_sup}	Supply air saturation pressure: pressure at which the water vapor of the air exiting the evaporator is saturated.
P_{suc}	Suction pressure: pressure at the inlet of the compressor.
P_{suc_loss}	Pressure drop in the suction line: from the evaporating pressure to the inlet of the compressor.
P_{sys}	Atmospheric air pressure of the system.
P_{XV}	Pressure drop through the expansion valve and subsequent distributor.
V	System voltage.
$\dot{V}_{air}$	Indoor air volume flow rate.
$\dot{W}_b$	Indoor blower motor power.
$\dot{W}_{comp}$	Compressor power.

5. PAPER C: APPLYING STATIC FAULT DETECTION AND DIAGNOSIS METHODS TO TRANSIENT AIR CONDITIONING DATA USING AN EQUILIBRIUM PREDICTION[1]

5.1 Synopsis

Fault detection and diagnosis methods for air conditioning systems typically apply static models after filtering out transient data using a steady state filter. However, air conditioning systems operating in the field often do not achieve a meaningful steady state and therefore the models cannot be applied because only transient data is available. This paper proposes a solution to this problem by predicting the equilibrium point using an exponential regression. The transient response of many system parameters such as cooling capacity, airflow, and refrigerant mass flow may be approximated as a first order dynamic response because the thermal mass in the system dominates other higher order dynamics. The best-fit for a decaying exponential will therefore yield a prediction for the equilibrium point, and static models may then be applied, thus enabling the use of static models with transient data. The method's performance is quantified using both randomly generated data (Monte Carlo simulations) and the measured response of a field-operating system during both fault-free and faulty operation.

5.2 Introduction

Fault detection and diagnosis (FDD) for vapor compression-based systems such as chillers, air conditioners, and heat pumps improves the operating efficiency and system reliability over time as faults are detected and rectified as soon as possible. The majority of proposed FDD methods use static models that are applied after the operating parameters (e.g. refrigerant and air temperatures) undergo some transient response upon start-up and then settle to a steady-state value. This process of filtering out the transient data to apply static models works well in a laboratory setting, but applying these methods to field-operating systems has some drawbacks.

The first drawback is that a field-operating system may never achieve a meaningful steady state.

[1]Rogers, A.P., and F. Guo, and B.P. Rasmussen. Applying Static Fault Detection and Diagnosis Methods to Transient Air Conditioning Data using an Equilibrium Prediction. **In Review**.

Unless the capacity matches the load (i.e. the system is variable speed or the load is very high), a vapor compression system will successfully lower the temperature and humidity in the conditioned space. The other operating parameters will then drift with the drifting conditioned space. This drawback may be addressed by analyzing the system operation in pseudo-steady state; after the startup period, the system parameters will follow the slowly changing driving conditions such as the conditioned space temperature and the outdoor ambient temperature. However, a second drawback is that a cycling system may not run long enough to achieve pseudo-steady state. This is particularly true during low-load conditions such as those experienced during the shoulder seasons. Performing FDD during the shoulder seasons is especially advantageous because repair costs are typically lower during these seasons and the downtime required for the repair will affect occupant comfort less. However, requiring that the system achieve steady state or pseudo-steady state may limit the use of FDD methods during these low-load conditions.

To address the challenges outlined above, this paper presents a method that enables static models to be applied even before steady-state and pseudo-steady state is achieved. As with many physical systems, the start-up transients in vapor compression systems may be modeled accurately with a first-order dynamic response. By fitting a first-order response to the start-up transients, the equilibrium point at the current operating conditions may be estimated before the system achieves steady state. This paper will quantify the performance of the regression method used to generate a first-order model, and then apply the regression to real air conditioning data from a field-installed system.

5.2.1 Previous Work

Fault detection and diagnosis (FDD) for air conditioning systems has been an active field of research for over twenty-five years [4, 5, 6], and the successful use of these methods can lead to significant energy savings [96, 97]. While some researchers have found value in analyzing the transients of the vapor compression cycle [27, 71], the vast majority of FDD methods use static models. In order to apply the static models, a steady-state filter must first be applied to the operating data. The proposed steady-state filters have traditionally been either slope-based or variance-based, or a combination of both [51]. Slope-based methods set a threshold that the slope during some past window must fall below, and variance-based methods set a threshold that the variance during some past window must fall below.

One of the earliest variance-based methods was presented in [64]; the authors proposed a steady state filter that has become known as the *exponentially weighted variance* method. The variance is computed with higher weights placed on more recent values, and the system is considered steady state when this weighted variance falls below a threshold. Both the threshold and the forgetting factor with which the values are weighted are tuning parameters. The authors of [64] originally applied the exponentially weighted variance method for FDD in air-handling units, but the method has since had a significant impact in the field of air conditioning FDD. The filter was applied directly in [20] and [25], and the filter was used as a foundation for the steady state filters used in [55, 21, 29].

The authors of [55] modified the filter such that a moving window variance was used instead of an exponentially weighted variance, and the authors of [21] then observed that, for the purpose of steady state detection, a moving window variance is nearly identical to a exponentially weighted variance. To improve the robustness of the filter, the authors of [21] combined a variance-based filter with a moving window slope-based filter. A similar combination of variance- and slope-based filters was used in [29]. When combining these filters, both the variance and the slope must fall below a given threshold before the operation is designated as steady state.

Each of the described steady state filters require that the system reach and then maintain some meaningful steady state operating point (though in practice often only pseudo steady state is attainable). The system must maintain the steady state operating point so that the variance (and sometimes slope) during a past period (either an exponentially weighted or a moving window) falls below a given threshold.

5.2.2 Overview

This paper proposes a novel approach to steady state detection which *predicts* the equilibrium point using exponential regression *before* steady state is ever reached. As described in this paper, the proposed method is only applicable to systems that exhibit a first-order dynamic response. However, many physical systems including most air conditioning systems can be approximated as first order.

The following section introduces a method to convert exponential regression to linear regression. The desired exponential form does not have a zero assymptote and therefore may not be converted to linear regression using a simple logarithm. However, an integral equation does convert the exponential regression to simple linear regression [42]. The accuracy of the exponential regression is then quantified using Monte Carlo simulations.

The next section then applies this exponential regression to operating data from a field-installed air conditioning system during both faulty and fault-free operation. To best evaluate the benefits provided by the proposed prediction, several metrics are proposed and the results are presented. These metrics include the accuracy of the predicted equilibrium point and the correlation between the predicted equilibrium point and the fault level.

In addition to the work presented in the main body of this paper, there are also two supplemental appendices. Appendix A further explains the challenges inherent in identifying an equilibrium point for air conditioning systems and Appendix B presents a virtual supply air humidity sensor used in this paper to estimate the system cooling capacity. While these appendices contain relevant and useful information, their content slightly detracts from the purpose of the paper which is to propose a method for predicting the equilibrium point using transient data.

5.3 Exponential Regression

Exponential regression is usually considered in the context of an exponential function with a zero assymptote; i.e. the function either diverges from or converges to zero. With a zero assymptote, a decaying exponential function is of the following form.

$$y(t) = y_0 e^{-t/\tau} \tag{5.1}$$

Note that $y(0) = y_0$. Finding a best-fit of an exponential function with a zero assymptote may be transformed to a linear regression problem by taking the logarithm of the function. When this is done, the logarithm of y is regressed rather than y itself.

$$\ln(y) = \ln(y_0) - \frac{1}{\tau}t \tag{5.2}$$

In many applications, an exponential function will not have a zero assymptote, and the familiar, linear exponential regression given in Equation 5.2 does not apply. Equation 5.3 provides a more general form of a decaying exponential function which has a nonzero final value.

$$y(t) = y_f - (y_f - y_0)e^{-t/\tau} \tag{5.3}$$

Note that $y(0) = y_0$ and $\lim_{t\to\infty} y = y_f$. Applying the natural logarithm to Equation 5.3 does not result in a clean linear regression. If the final value were known, then the data could be shifted such that $y - y_f$ were regressed, and the logarithm would again transform the problem to linear regression. However, the final value is often unknown, and in the context of predicting the equilibrium point, estimating the final value is the primary purpose of generating the regression model.

The author of [42] proposed that the exponential function given in Equation 5.3 could be transformed to linear regression using integration rather than the logarithm. Equations 5.4 and 5.5 show the result of integrating both sides of Equation 5.3.

$$\int_{t_1}^{t} y(u)du = y_f \int_{t_1}^{t} du - (y_f - y_0) \int_{t_1}^{t} e^{-u/\tau}du \tag{5.4}$$

$$\int_{t_1}^{t} y(u)\, du = y_f(t - t_1) + \tau(y_f - y_0)\left[e^{-t/\tau} - e^{-t_1/\tau}\right] \tag{5.5}$$

Note that (t_1, y_1) is the first available data-point that the exponential fit is being applied to. Equation 5.5 may be simplified by considering Equation 5.6.

$$y_1 - y = (y_f - y_0)\left[e^{-t/\tau} - e^{-t_1/\tau}\right] \tag{5.6}$$

Therefore, integrating both sides of the general exponential function in Equation 5.3 ultimately reduces to the linear regression problem shown in Equation 5.7.

$$\begin{aligned} S(t) =& y_f(t - t_1) + \tau(y_1 - y) \\ \text{where } S(t) =& \int_{t_1}^{t} y(u)du \end{aligned} \tag{5.7}$$

After obtaining a numerical representation for $S(t)$, linear regression may be performed using $S(t)$ as a dependent variable, and using both $t - t_1$ and $y_1 - y$ as independent variables. The regression results in coefficients y_f and τ. Note that this linear regression minimizes the error of the cumulative integral of y rather than of y itself.

The final value y_f and the time constant τ may be obtained by performing linear regression with Equation 5.7. The author of [42] proposed that the initial value y_0 may then be obtained using a second linear regression. The author proposed that the estimated time constant be substituted back into Equation 5.3 and then $e^{-t/\tau}$ would be used as a dependent variable with coefficient $y_0 - y_f$. However, this second regression is not necessary if a best-fit estimate for the initial value y_0 is not desired.

5.3.1 An Example of Exponential Regression Using the Integral Method

In order to implement the linear regression outlined in Equation 5.7, the integral of the variable y should be determined numerically. There are many available methods, but the author of [42] suggested using the simple trapezoid rule, and this is the method employed in this paper. The trapezoidal integration is outlined in Equation 5.8, but this common method may be found pre-packaged in many programming languages (e.g. `scipy.integrate.cumtrapz` in Python).

$$\begin{cases} S_1 = 0 \\ S_k = S_{k-1} + \frac{1}{2}(y_k + y_{k-1})(x_k - x_{k-1}) \end{cases} \tag{5.8}$$

Using S as the dependent variable and $t - t_1$ and $y_1 - y$ as the independent variables (Equation 5.7), the function parameters y_f and τ are straightforward to obtain using linear regression.

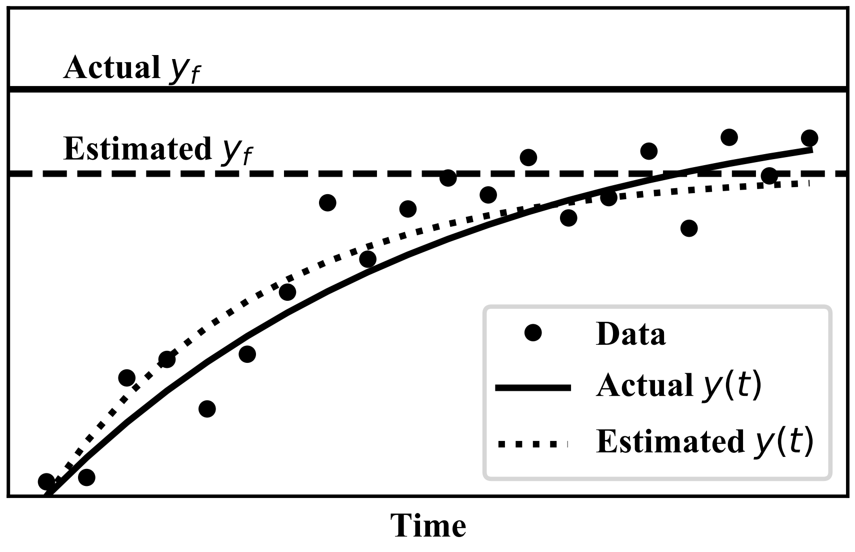


Figure 5.1: An example of exponential regression using the integral method applied to a simulated time series. The time series is generated with normally distributed noise with standard deviation of 10% of the rise ($y_f - y_0$), and with a length T equal to twice the nominal time constant τ.

One example using this method is provided in Figure 5.1. The data in Figure 5.1 is generated by adding normally distributed noise to the nominal exponential function. The resulting fit underestimates the rise $y_f - y_0$ by 21% and underestimates the time constant τ by 46%. There are two intuitive characteristics of the time series visualized in Figure 5.1 that substantially limit the accuracy of the estimated parameters. First, the magnitude of the noise is high relative to the magnitude of the rise $y_f - y_0$. Second, the length of the time series T is short relative to the time constant τ. Additionally, the accuracy of the regression will depend on the specific noise profile that was randomly generated for this time series; i.e. the accuracy of the estimated parameters y_f and τ will be different for two different randomly generated noise profiles of the same magnitude.

5.3.2 Analyzing the Accuracy of the Exponential Regression

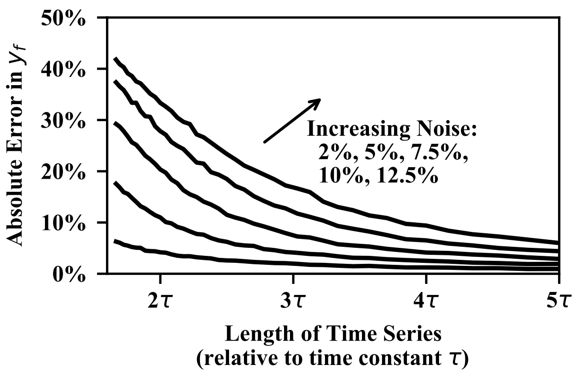
The example provided in Figure 5.1 estimated y_f with 21% error and τ with 46% error. However, the estimates would have been more accurate had there been less noise *or* had the duration been longer. Furthermore, the accuracy also depends on the specific randomly generated noise profile. To determine the *typical* accuracy for a given combination of noise and length, many time series with the same combination were generated and the average accuracy was determined. The average accuracy in y_f was 27.8% and the average accuracy in τ was 53.1% for 1000 randomly generated time series with the same characteristics as those in Figure 5.1: noise with standard deviation of $10\% \times [y_f - y_0]$ and $T = 2\tau$.

The Monte Carlo method described above involves generating many time series with a given combination of noise and length and then taking the average to determine the typical accuracy that one could expect. Unfortunately, this process misses an important aspect; the regression may result in a *negative* value for τ such that the best-fit function exponentially grows to infinity thus rendering the estimate in final value y_f meaningless. In fact, the 27.8% error for y_f and 53.1% error for τ reported in the previous paragraph were determined using only the simulations that resulted in a positive τ; 20 of the 1000 simulations (2%) resulted in a negative value. In this context, a negative value for τ will be considered a 'failure' and the failure rate must be considered in addition to the typical errors in y_f and τ.

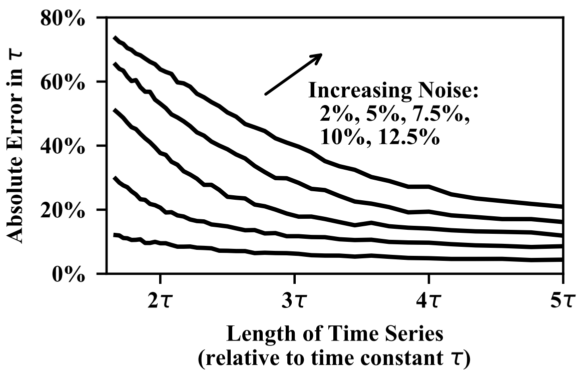
This Monte Carlo method was used to quantify the typical accuracy in y_f and τ and the failure rate for several combinations of noise and length. 1000 simulation were performed at each combination and the results are presented in Figure 5.2. The plots given in Figure 5.2 are intended as a design tool in implementing the integral-based exponential regression. An engineer may identify the length of data required to obtain a desired accuracy for a given noise magnitude (or *vice versa*). If the signal has 5% noise relative to the rise ($5\% \times [y_f - y_0]$) and an average error in τ of less than 10% is required, then the length of data T must be at least 4τ. For the purpose of predicting the equilibrium point, the error in the time constant is of less concern than the error in the final value y_f. Fortunately, the estimate for the final value is, on average, more accurate than the estimate for the time constant.

5.4 Application to Air Conditioning Operation

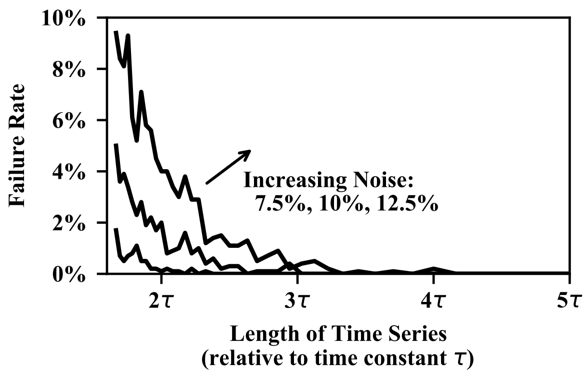
In order to further test the exponential regression for the purposes of equilibrium prediction and fault detection, the integral-based exponential regression was applied to the estimated cooling



(a) The average error in the final value y_f.



(b) The average error in the time constant τ.



(c) The failure rate of the regression.

Figure 5.2: An analysis of the effectiveness of the integral-based exponential regression for various magnitudes in noise (relative to rise $y_f - y_0$) and for various lengths (relative to time constant τ). These results were obtained using 1000 simulations at each combination of these two parameters, and 'failure' occurs when the estimated time constant is negative.

capacity during 16 tests on a field-operating air conditioning system. These 16 tests contain various degrees of outdoor airflow faults introduced by covering a portion of the condenser, and the cooling capacity was estimated using the air-side evaporator energy balance given in Equation 5.9.

$$Q = \dot{m}_{air}\Big(c_p(T_{ret} - T_{sup}) + h_{we}(\omega_{ret} - \omega_{sup})\Big) \tag{5.9}$$

Equation 5.9 was applied using the nominal air flow rate, the measured return and supply air temperatures, the measured return air humidity, and the virtual supply air humidity sensor presented

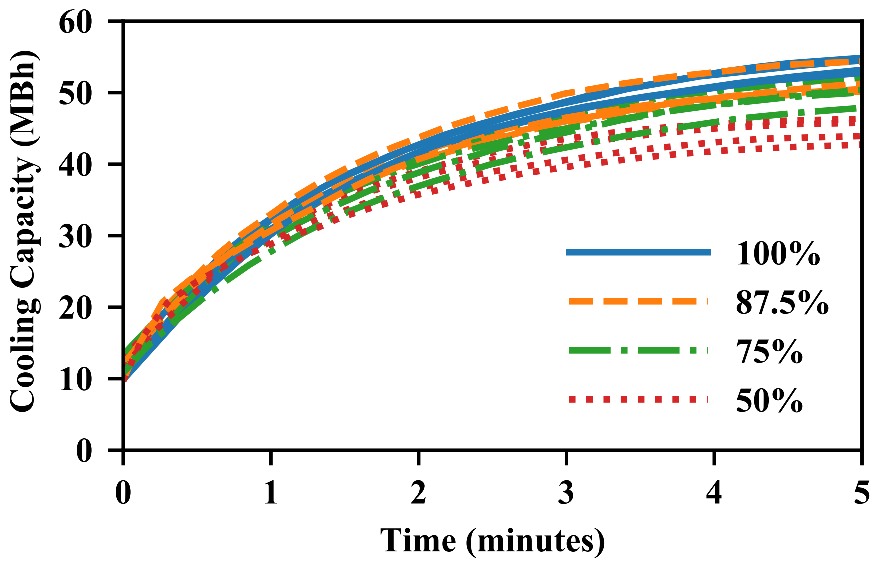


Figure 5.3: Estimated cooling capacity for 16 tests with varying degrees of condenser airflow faults. Fault levels include 100% (fault-free), 87.5%, 75%, and 50% condenser area relative to the nominal condenser area.

in Appendix B.

Figure 5.3 shows the time series for each of these 16 tests *measured from the point at which the coil becomes wet*. The shift from dry to wet coil conditions results in a shift in slope (Figure 5.11 in Appendix B) and the use of a decaying exponential is more appropriate after this shift in slope. The estimated cooling capacity seems to nearly reach steady state within five minutes for the tests shown in 5.3. However, if the system were to continue to run, the cooling capacity would degrade as shown in Appendix A.

5.4.1 Metrics for Evaluating Performance

The integral-based exponential regression could be applied to the five-minute tests shown in Figure 5.3 in order to obtain an approximate prediction for the steady state cooling capacity at the corresponding driving conditions. However, the full five minutes do not need to be used. The exponential regression could be applied only to the first two or three minutes, and the resulting steady state prediction should be similar to the prediction obtained using the full five minutes. The

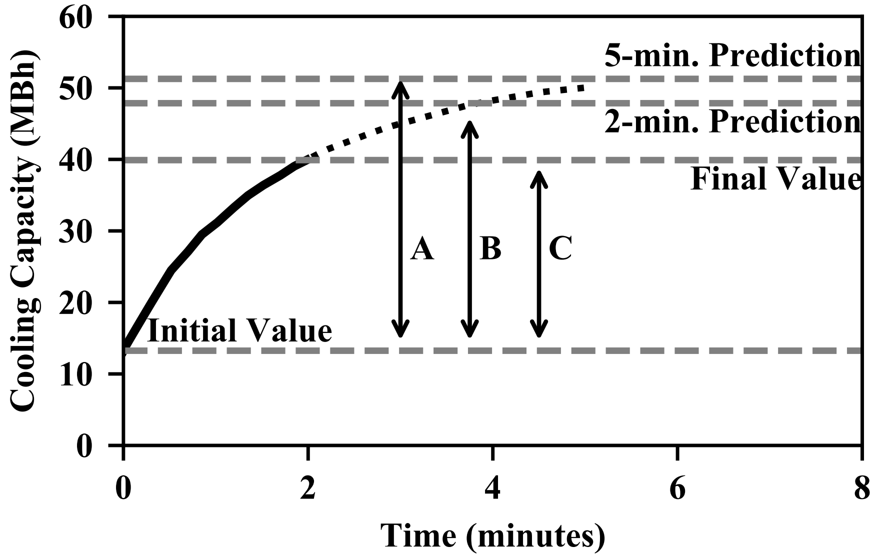


Figure 5.4: The estimated cooling capacity for a single test. The predicted steady state value using 5 minutes (5-min. prediction) and 2 minutes (2-min. prediction) are shown with the final value after 2 minutes of run time and the initial value. Comparing the 2-minute prediction with the 5-minute prediction and the final value provides insight to how advantageous a steady state prediction is.

advantage gained by using this prediction may be evaluated using several metrics outlined below.

Figure 5.4 shows one of the tests with (a) the prediction that results from using 5 minutes (5-minute prediction), (b) the prediction that results from using only 2 minutes (2-minute prediction), and (c) the final value after 2 minutes without any prediction. Ideally, the 2-minute prediction will be close to the 5-minute prediction. If the prediction is near the final value, then the prediction provides little benefit; the final value could be used instead. The ratio of the prediction to the 5-minute prediction (B:C) and the ratio of the prediction to the final value (B:A) are useful metrics in evaluating the prediction.

The 5-minute prediction should be a more accurate representation of the true equilibrium point such that the ratio of the prediction to the 5-minute prediction (B:C) provides an estimate of the relative *accuracy* of the prediction. In addition to this metric, the ratio of the prediction to the final value (B:A) provides a representation for how *useful* the prediction is; if the prediction simply returns the final value then the advantage of performing the prediction is minimal.

In addition to the above metrics, the correlation between the predicted value and the fault level should also be evaluated. The intended application for this prediction is to perform fault detection and diagnosis, and therefore the correlation between the estimated cooling capacity and the fault level is a key metric for evaluating the performance. In this paper, the correlation is represented by the coefficient of determination R^2 that results from a 2nd-order polynomial fit between fault level and cooling capacity.

5.4.2 Results

Figure 5.5 shows how the predicted equilibrium point compares with the 5-minute-prediction when less than five minutes are used for the regression (ratio B:C from Figure 5.4). The relative prediction is always 100% when 5 minutes are used because the prediction is plotted relative to the 5-minute prediction. However, less than 5% error is introduced when only 4 minutes are used and less than 10% error is introduced when only 3 minutes are used for regression.

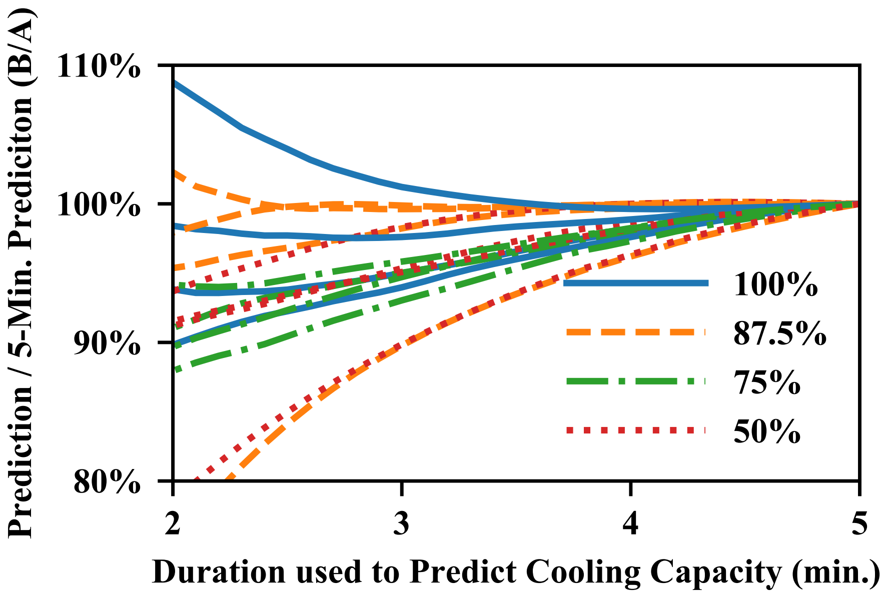


Figure 5.5: The predicted steady state cooling capacity estimate when less than five minutes are used for regression. The predicted capacity is presented relative to the estimate obtained using the full five minutes of run time.

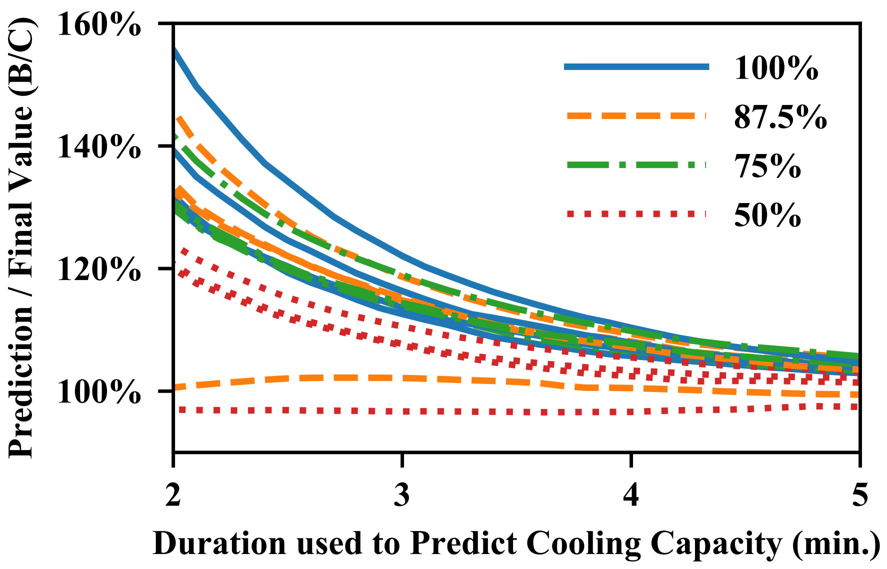


Figure 5.6: The ratio of the predicted steady state cooling capacity to the final value. The prediction is more advantageous when it is far from the final value.

Figure 5.6 shows how the predicted equilibrium point compares with the final value (ratio B:A from Figure 5.4). The 5-minute prediction is typically very close to the final value because the operation is nearly steady state after 5 minutes. Therefore, predicting the steady state operating point using 5 minutes is less advantageous. Two of the tests result in a prediction that is nearly identical to the final value even when only 2 minutes are used. The decaying exponential model is evidently less effective for these two tests, and these are the same two tests with the greatest error in Figure 5.5. However, the method effectively predicts an equilibrium point above the final value for the vast majority of tests.

The ratios shown in Figure 5.5 and 5.6 provide insight to how accurate the prediction is and how much run time the prediction could eliminate. However, this prediction is proposed for the purpose of fault detection and diagnosis, and sensitivity of the predicted equilibrium value to the fault level is therefore an important measure of its utility. Figure 5.7 provides further validation of the usefulness of this method for fault detection. In Figure 5.7, the 5-minute predictions and the 3-minute predictions are plotted next to each other for each fault level. Figure 5.5 showed that a 3-

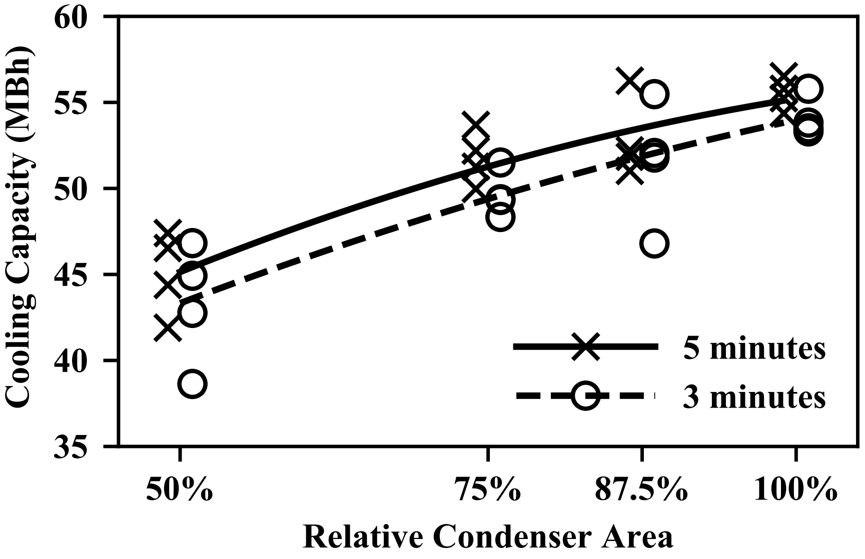


Figure 5.7: Prediction for steady state cooling capacity at several outdoor airflow fault levels using both 5 minutes and 3 minutes of run time. Both predictions show similar behavior. A 2nd-order polynomial fit for both data sets highlights this similar behavior.

minute prediction introduces less than 10% error relative to the 5-minute prediction, and Figure 5.7 shows that both predictions result in the same cooling capacity behavior; both predictions clearly show that the cooling capacity degrades as the outdoor airflow fault level increases. The 3-minute prediction does tend to be lower than the 5-minute prediction indicating that the prediction should alwayhs be made with a consistent length of data.

The 2nd-order polynomial fit shown in Figure 5.7 further confirms that both the 5-minute and the 3-minute prediction show similar behavior as the fault level increases. However, the 5-minute prediction is more repeatable (i.e. has less variation) such that the correlation between fault level and predicted steady state capacity is stronger for the 5-minute prediction. The coefficient of determination (R^2 value) that results from using between 2 and 5 minutes is plotted in Figure 5.8. Using 5 minutes for the prediction results in an R^2 value of 0.83 whereas using 3 minutes results in an R^2 value of 0.75.

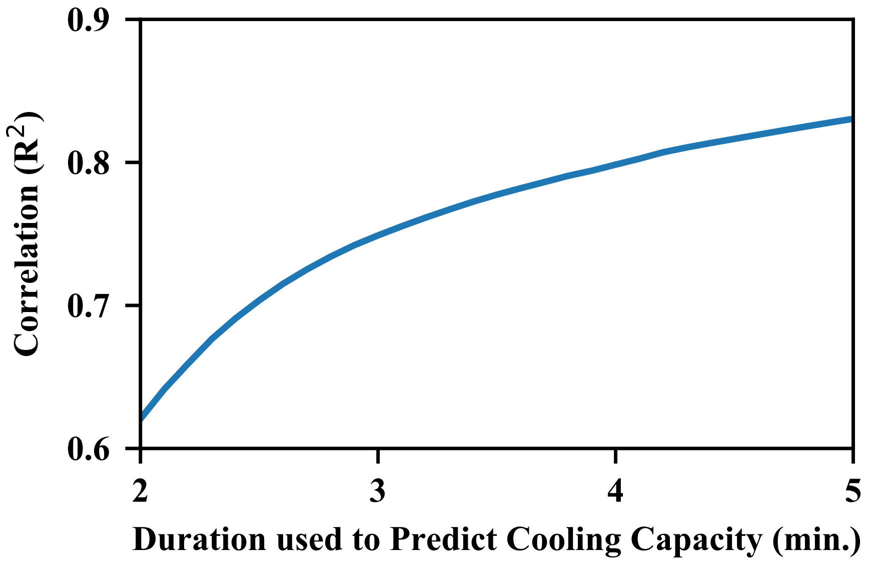


Figure 5.8: The R^2 value that results from predicting steady state using between 2 and 5 minutes of run time. The R^2 value is a measure of the correlation between fault level and predicted steady state capacity using a 2nd-order polynomial fit.

5.5 Conclusions

The proposed prediction method provides an alternative to traditional steady state filtering methods. However, the proposed method relies on an exponential regression and is therefore only applicable to parameters which may be approximated with a first order dynamic response. This approximation is most valid for system level parameters such as the cooling capacity and the refrigerant mass flow rate. More specific parameters such as the refrigerant temperature at specific locations (e.g. two-phase coil temperatures or liquid-line temperature) are less likely to exhibit first-order behavior. To address this challenge, other prediction methods could be developed for these parameters, or fault diagnosis methods could be developed using system-level parameters such as capacity, airflow, massflow, and efficiency.

This paper has presented a method to estimate the equilibrium point using transient data. To further validate the methods proposed herein, the prediction method should be incorporated into a more complete fault detection and diagnosis framework which includes determining residuals using static models.

5.6 Acknowledgment

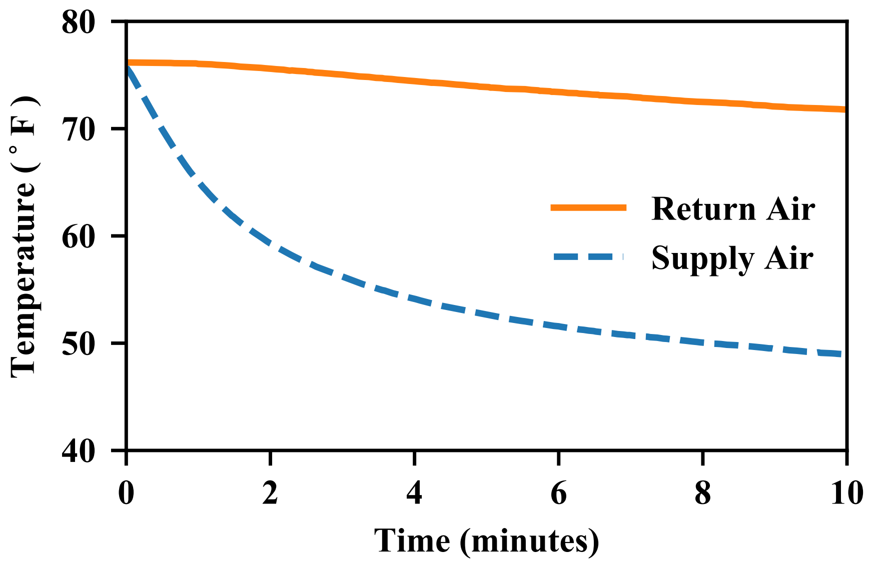
This material is based upon work supported by Trane, a subsidiary of Ingersoll Rand. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Trane.

5.7 Appendix A: Practical Challenges with Field-Operating Systems

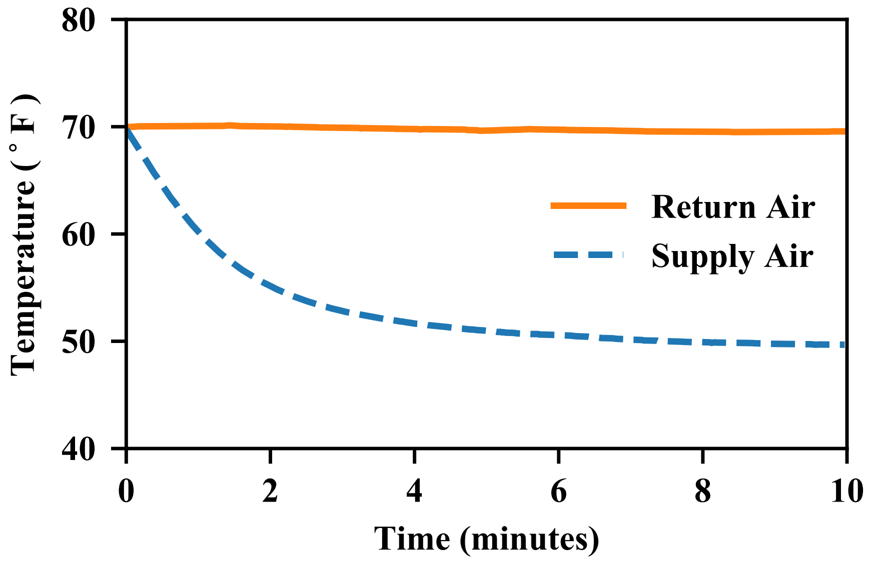
The main body of this paper asserts that air conditioning systems will typically reach only pseudo-steady-state rather than obtaining a true steady state. This is because, unless the cooling capacity matches the cooling load, the air conditioning system will cause the indoor temperature to reduce and system parameters will drift with this driving condition. This appendix provides an example of this behavior.

Consider Figure 5.9 which plots the return and supply air temperature for two different tests. The return and supply air conditions for a 3.5 ton packaged heat pump unit were obtained using temperature and relative humidity data loggers. However, to eliminate the need for the supply air humidity sensor, the supply air humidity was estimated using the other three measurements as explained in Appendix B.

The first test (Figure 5.9a) shows typical operation in which the supply air decreases as the air conditioner operates and the return air correspondingly decreases. When the cooling load is relatively low (e.g. the ambient temperature is cooler), an appropriately-sized air conditioning system should be capable of reducing the indoor temperature as it operates; therefore, the the indoor temperature reduces from 76.2 °F to 70.7 °F in Figure 5.9a. On the other hand, the indoor temperature changes very little during the second test (Figure 5.9b). This is because fresh, outdoor air is used for the return air during the second test such that the return air remains nearly constant. Assuming that the outdoor conditions are constant, this second test will approach a meaningful steady state operation whereas the first test will only reach a true steady state after the indoor temperature has decreased to the point where the degraded capacity matches the cooling load.



(a) Temperature response during typical operation with low-load conditions.



(b) Temperature response when fresh, outdoor air is used such that the return air temperature remains relatively constant.

Figure 5.9: Two examples of the air temperature response for a field-operating air conditioning system.

This example illustrates a challenge in performing steady state detection on a field-operating air conditioning system, especially during low-load conditions.

An air conditioner's operation depends primarily on the return air temperature and humidity and on the ambient air temperature [37]. As the system cools the home, the return air temperature decreases (Figure 5.9a) and system parameters such as the cooling capacity and efficiency change with the driving conditions. The latent cooling capacity is especially affected by the decreasing return air temperature. Consider Figure 5.10 in which the humidity ratio drop across the evaporator ΔW is plotted for both tests shown in Figure 5.9. When fresh air is returned such that the return air conditions remain relatively constant, the latent cooling cooling capacity (proportional to the drop in humidity ratio ΔW) may be approximated as a first order dynamic response and it approaches a steady state value. However, the return air temperature decreases during typical operation and this results in a drop in latent cooling capacity. As the return air temperature drops, the return air humidity also drops and there is less water vapor in the air to condense during the cooling process. Thus, the latent cooling capacity's behavior during the first two minutes of both tests is similar, but during typical operation the latent cooling capacity will start decreasing rather than reach a steady state. During typical operation, the latent cooling capacity will eventually reach a true steady state

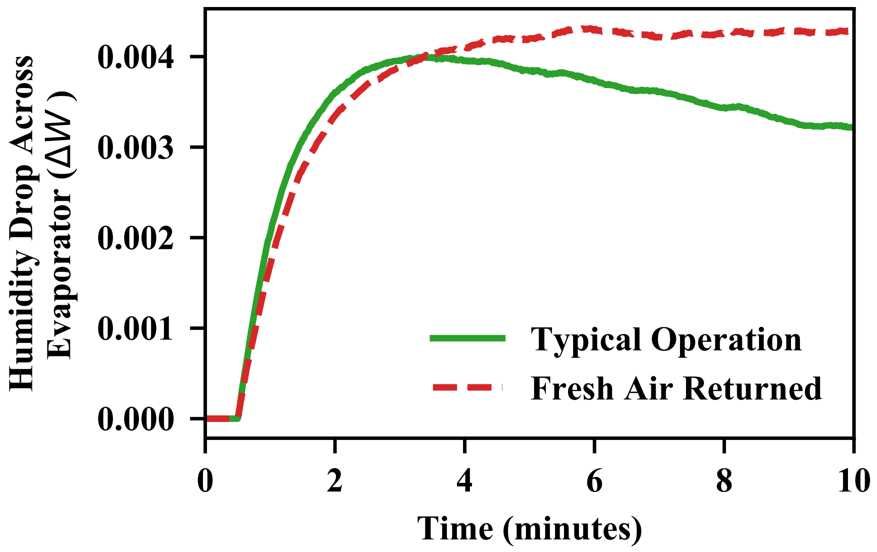


Figure 5.10: The estimated drop in humidity ratio (lbm water/ lbm dry air) associated with the tests shown in Figure 5.9.

after the indoor temperature has been cooled to a point such that the degraded cooling capacity matches the load. However, the operating conditions would be outside the typical and desired operating conditions, and this steady state would not be helpful in performing fault detection and diagnosis on the system.

5.8 Appendix B: A Virtual Supply Air Humidity Sensor

The humidity ratio in Equation 5.9 and Figure 5.10 is estimated using a virtual supply air humidity sensor. The return air temperature and relative humidity are measured, and the return air humidity ratio is obtained using Equation 5.10.

$$W\Big(\phi, P_{sat}(T)\Big) = 0.621945\frac{\phi P_{sat}(T)}{P - \phi P_{sat}(T)} \tag{5.10}$$

where the water vapor saturation pressure P_{sat} is a function of absolute (Rankine scale) dry air temperature shown in Equation 5.11 [98], and P is the absolute system pressure (typically 14.7 psi near sea level).

$$\begin{aligned} P_{sat}(T) =& e^{(C_8/T + C_9 + C_{10}T + C_{11}T^2 + C_{12}T^3 + C_{13}\ln T)} \\ \text{where } C_8 &= -1.0440397\text{E}{+}04 \\ C_9 &= -1.1294650\text{E}{+}01 \\ C_{10} &= -2.7022355\text{E}{-}02 \\ C_{11} &= +1.2890360\text{E}{-}05 \\ C_{12} &= -2.4780681\text{E}{-}09 \\ C_{13} &= +6.5459673\text{E}{+}00 \end{aligned} \tag{5.11}$$

When the cooling coil is dry, the supply air can be expected to have the same humidity ratio as the return air. As the supply air temperature drops below the saturation point, water will condense out of the air and the supply air will effectively be saturated. However in practice, the supply air is typically not fully saturated during wet coil conditions. Rather, a relative humidity of 90%

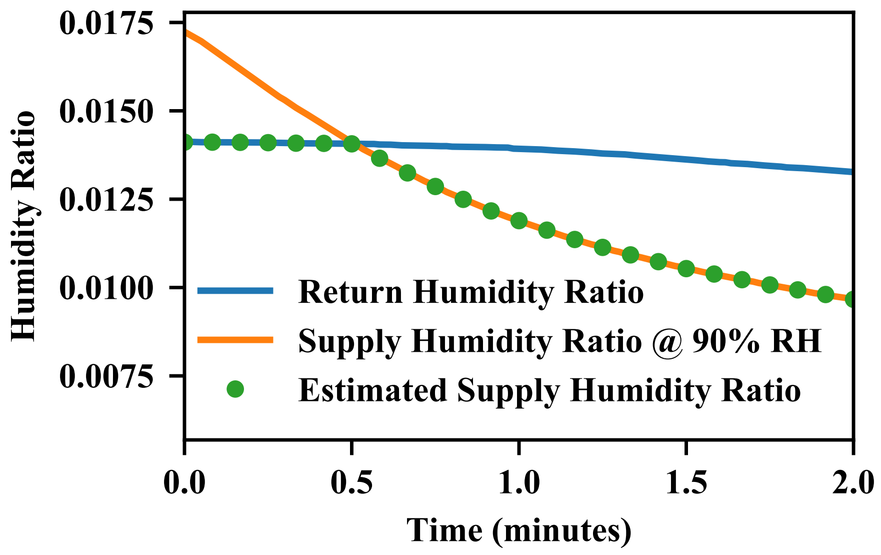


Figure 5.11: The estimated supply air humidity ratio for the first 2 minutes of the test shown in Figure 5.9a. The supply air humidity ratio is estimated as the minimum of the return air humidity ratio and the supply humidity ratio assuming 90% RH (Equation 5.12).

is commonly assumed [99, 100, 101]. A reasonable estimate for the supply air temperature is therefore given in Equation 5.12.

$$W_s = \min\Bigg(W\Big(\phi_r, P_{sat}(T_r)\Big), W\Big(0.9, P_{sat}(T_s)\Big)\Bigg) \tag{5.12}$$

In Equation 5.12, the left part in the minimum operator represents the return air humidity ratio and the right part represents the supply air humidity ratio assuming 90% relative humidity (RH). Figure 5.11 shows this estimation for the first 2 minutes of the tests from Figure 5.9a. The supply air humidity ratio is equal to the return air humidity ratio until the supply air becomes effectively saturated (90% RH).

5.8.1 Validating the Supply Air Humidity Virtual Sensor

To validate the supply air humidity virtual sensor, the estimated supply air humidity ratio is compared with the supply air humidity ratio derived from the measured relative humidity. With

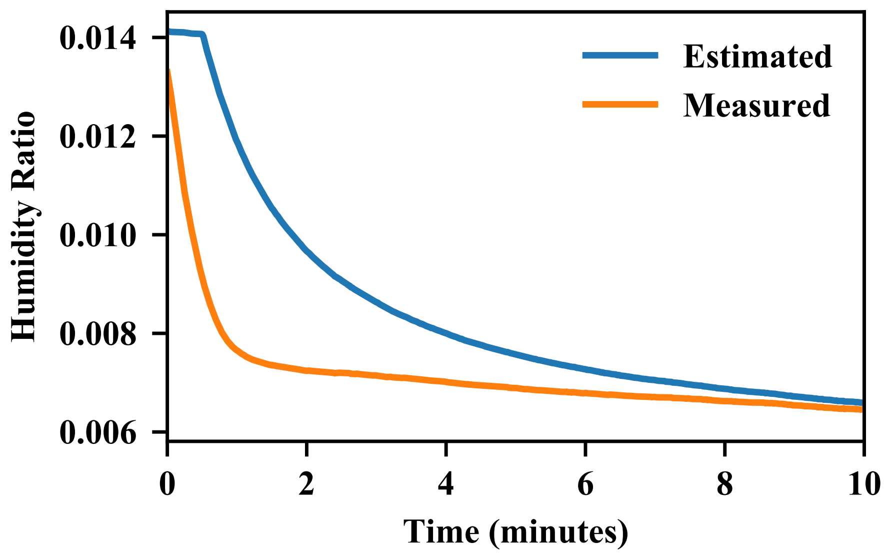


Figure 5.12: The supply air humidity ratio obtained using Equation 5.12 (Estimated) and using the measured supply relative humidity (Measured) for the test shown in Figure 5.9a.

both the supply air temperature and relative humidity measured, the supply air humidity ratio becomes simply $W_s = W(\phi_s, P_{sat}(T_s))$. Figure 5.12 shows the supply air humidity ratio obtained using both the virtual sensor (Equation 5.12) and the measured supply air relative humidity. While there is a significant difference in the transient behavior, both methods approach the same steady state value. Part of the discrepancy in the transient behavior is likely due to the dynamics introduced by the sensors, and higher quality sensors would provide better insight to the true transient behavior. However, the comparison shown in Figure 5.12 confirms that the virtual supply air humidity sensor is valid for the purpose of predicting the equilibrium point. Figure 5.12 also validates the assumption that the supply air has 90% relative humidity when the coil is wet.

6. PAPER D: LABELING MODES OF OPERATION AND EXTRACTING FEATURES FOR FAULT DETECTION WITH CLOUD-BASED THERMOSTAT DATA[1]

6.1 Synopsis

This paper presents a method for transforming raw cloud-based thermostat data for cycling systems into a set of operating modes that is useful for large scale data analysis. Thermostat data typically includes the setpoint temperatures, the actual indoor temperature and the operating mode. These raw thermostat operating modes include 'cooling on', 'heating on', and 'system off'. The transformed operating modes include regulating modes, tracking modes, and free response modes. These new modes, which can be generated during data preprocessing, are used to more clearly show key system performance metrics and identify change points in the time series thermostat data. This paper includes the filtering logic used to label the operating modes and examples of insightful behavior that has been captured using this preprocessing method.

6.2 Introduction

Combined, air conditioning and space heating account for 30% of electricity expenditures and 40% of total energy expenditures in the U.S. residential sector [3]. Given that the residential sector accounts for 39% of U.S. electricity expenditures [3, 1] and 20% of total U.S. energy expenditures [47], *residential air conditioning and space heating accounts for 11.7% of U.S. electricity expenditures and 8.0% of total U.S energy expenditures.*

6.2.1 Background

There are three primary means to reduce energy consumption of residential air conditioning and space conditioning: (1) improving the base or installed efficiency of a system, (2) implementing more efficient control methods, and (3) improving system maintenance such that the installed installed efficiency is maintained over time. Fault detection and diagnostics (FDD) falls within this

[1]Rogers, A.P., and F. Guo, and J. Martinez, and B.P. Rasmussen. Labeling Modes of Operation and Extracting Features for Fault Detection with Cloud-Based Thermostat Data. **In Review**.

third category; detecting a fault early allows for more timely maintenance. Diagnosing the fault automates the troubleshooting process thus increasing the cost-effectiveness of system maintenance. While FDD has been researched extensively for vapor compression-based systems [4, 5, 6], the state-of-the-art methods are still cost-prohibitive in the residential sector. This is because the proposed methods require sensors installed in the vapor compression cycle in order to extract useful features [23, 24].

While FDD methods that require many additional sensors may currently be cost-prohibitive in the residential sector, the system thermostat data may still be used to detect fault behavior. Furthermore, with the relatively recent advent of smart thermostats, the thermostat data for many systems are maintained on remote servers allowing for these thermostat-based methods to be developed using real data.

Thermostat-based FDD is still a relatively new field of research. For example, in 2016 [77] proposed simple physics-based methods for determining the lumped home thermal resistance and thermal heat capacity using thermostat heating data. In 2013 [79] presented an algorithm for analyzing temperature-only data (using a temperature data logger by the thermostat rather than a thermostat itself) and then recommending heating behavior to reduce energy consumption. While [77] was focused on building construction-level faults, [79] was focused on occupant-level faults. [80] proposed a thermostat-based fault detection method using the recursive least squares algorithm to identify model parameters. The method, presented in 2017, was focused on operational faults such as airflow restriction and a total loss of heating or cooling. The method was demonstrated with data generated using building energy simulation software, but the method was not demonstrated using real operating data.

A major challenge with thermostat data is that, in it's raw form, the data is very sparse. At a minimum, thermostat data will contain the indoor temperature, the setpoint (separate heating and cooling setpoints if applicable), and the system mode (heating, cooling, fan-on, off). More recent thermostats may include an indoor relative humidity measurement and an estimate of the outdoor temperature obtained from online sources. Due to the inherent limitations of this dataset, effective preprocessing and filtering will be critical to extract useful information from thermostat data.

6.2.2 Introducing Modes of Operation

This paper presents thermostat data preprocessing methods for systems that cycle on and off to control the indoor temperature (i.e. cycling systems). Specifically, the proposed preprocessing methods filter the thermostat data in order to extract useful modes of operation. These filtered modes of operation differ from those already reported in the thermostat data (cooling, heating, off, etc.). There are several reasons that the modes of operation should be redefined:

1. When the thermostat is regulating the indoor temperature (i.e. maintaining a *constant* temperature), defining the control input to be the *duty cycle* is more helpful than using the raw on/off cycles.

2. As an extension to the point above, the system should not really be viewed as 'off' when a constant temperature is being maintained. In other words, the system is often still actively controlling the temperature during periods between two run cycles. The system should be defined as 'off' when the temperature is not being actively controlled (e.g. when the indoor temperature is between heating and cooling setpoints or when the system is manually shut-off).

3. The cooling effort required to lower the indoor temperature from 76°F to 72°F should be analyzed differently than the cooling effort required to maintain an indoor temperature of 72°F.

The three points above suggest that the raw operating modes (cooling, heating, off) could be transformed into a new set of operating modes that is better suited for data analysis. The proposed modes include cooling and heating regulating modes, cooling and heating tracking modes, and the free response mode.

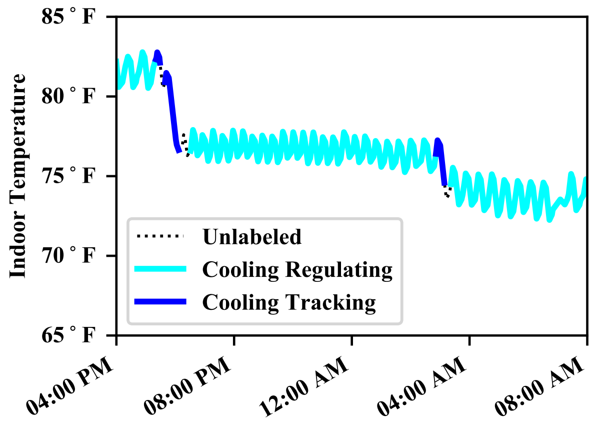
1. **Regulating Modes:** In control theory, a system maintaining a constant setpoint is said to be regulating [102]. A cycling air conditioner or space heating system is considered regulating

when the system is cycling on and off to maintain a constant temperature. The system is therefore controlling the temperature regardless of whether it is currently on or off and the control input is effectively the duty cycle.

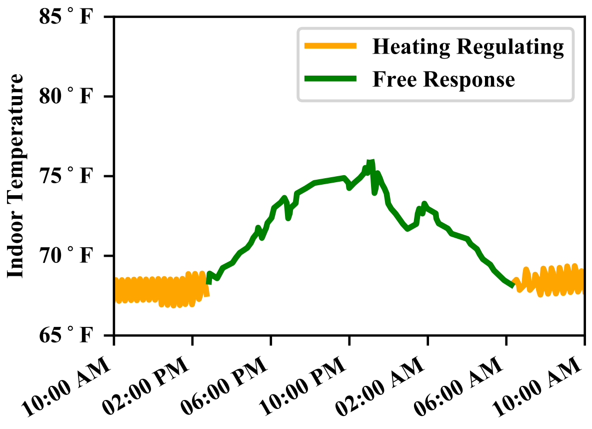
2. **Tracking Modes:** In control theory, a system following a reference trajectory is said to be tracking [102]. While a thermostat setpoint is only ever constant a given point in time, the setpoint can be changed suddenly such that the system must exert control effort to reach the new setpoint. The period of time in which the system is striving the reach the setpoint is considered a tracking period.

3. **Free Response Mode:** The free response mode consists of periods of time in which the setpoints are not activated. This may be because the indoor temperature is between the heating and cooling setpoints or because the occupant has manually switched the system off.

Figure 6.1 provides a demonstration of several of these proposed modes of operation. Cooling and heating regulating modes consist of periods when the indoor temperature oscillates around a constant value. The cool-tracking mode consists of periods when the indoor temperature decreases (presumably due to a reduction in setpoint). The free response period shown in Figure 6.1 occurs



(a) An example of cool-regulating and cool-tracking modes.



(b) An example of heat-regulating and free response modes.

Figure 6.1: Examples of the proposed modes of operation using cloud-based thermostat data.

in between two heat-regulating periods and the indoor temperature rises and then falls during this period. Presumably, this is because the indoor temperature was above the heating setpoint and had not yet reached the cooling setpoint.

The remainder of this paper provides further support for transforming the modes of operation, further details regarding the process by which this transformation takes place, and several interesting observations that have been made using these labeled modes.

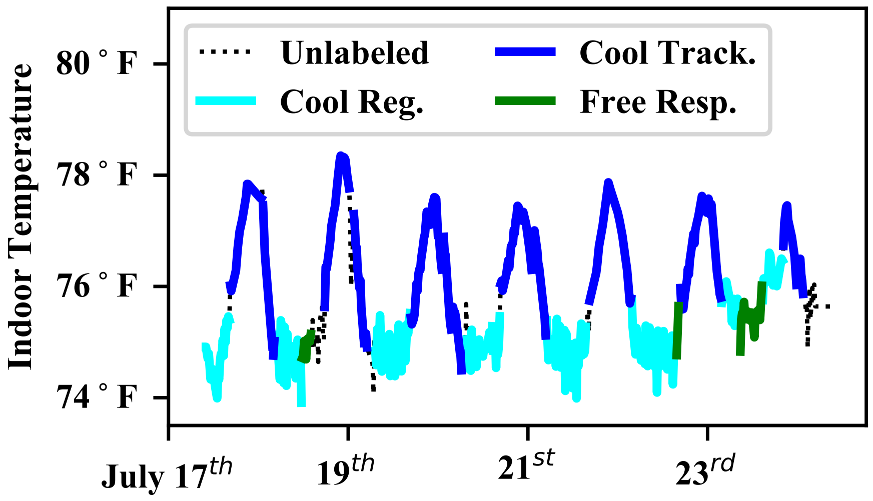
6.3 Data Cleansing through Mode Labeling

Filtering thermostat according to the control modes such that regulating modes, tracking modes, and the free response mode can be analyzed individually results in a dataset which more clearly shows the underlying system performance. This section provides several examples that support this claim.

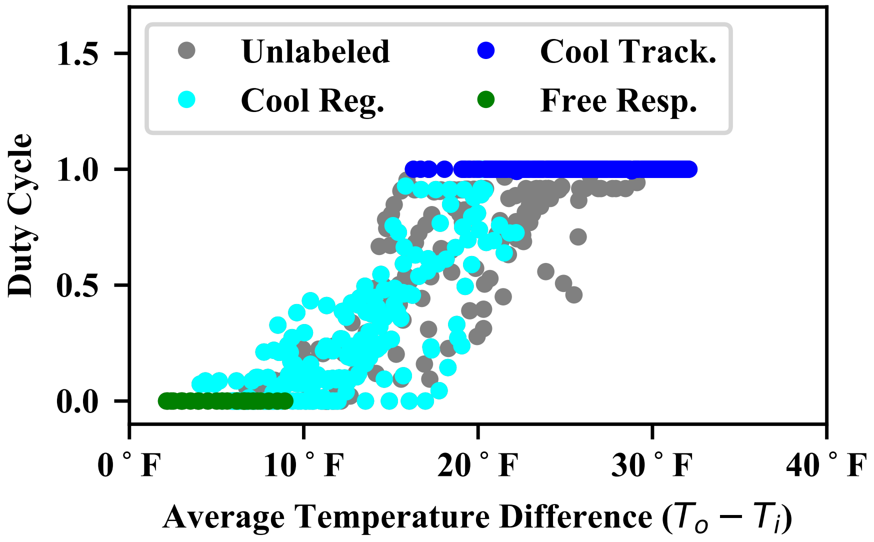
The relationship between the temperature difference between outdoor and indoor temperature ($T_o - T_i$) and the cooling effort or duty cycle is a very common relationship of interest. However, this relationship should represent the duty cycle required to maintain a constant temperature and therefore the tracking and free response modes should not be included. Figure 6.2 shows this relationship for a single week for a single system. The regulating periods exhibit the expected linear relationship between duty cycle and temperature difference. If this relationship were to be quantified (e.g. using linear regression), the tracking and free response modes must be filtered out in order to avoid a biased result.

Notice that the tracking modes in Figure 6.2 are not associated with a reduction in indoor temperature. Rather, the temperature rises and falls during the tracking mode such that the overall indoor temperature rise is approximately zero. This tracking behavior occurs when the system is unable to maintain the setpoint and the compressor remains on for an extended time until the setpoint is achieved; and this behavior is associated with either a degraded cooling capacity or an undersized capacity relative to the load.

Another important relationship is the relationship between the temperature difference and the rise/fall in indoor temperature; i.e. how much does the indoor temperature rise in an hour when the



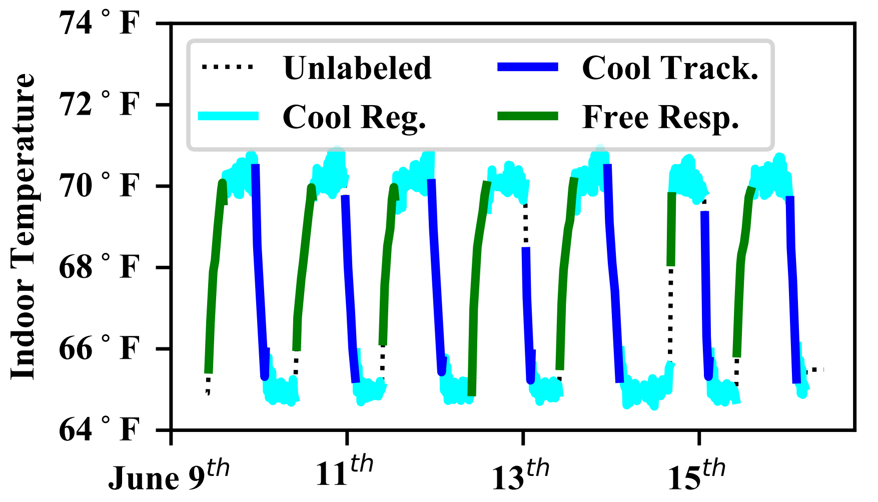
(a) Time series showing one week of indoor temperature with modes labeled.



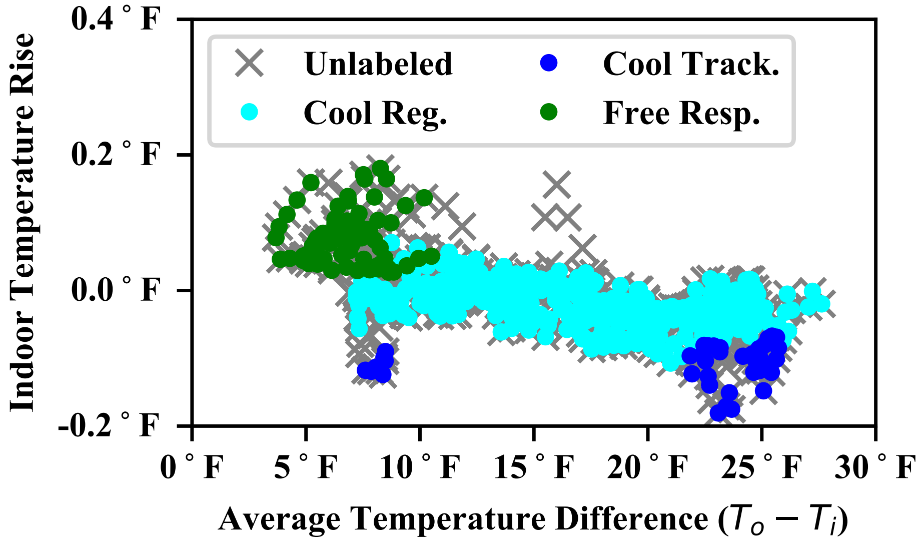
(b) Relationship between duty cycle and temperature difference using the time series in (a). Each point represents a 1-hour period.

Figure 6.2: An example of how preprocessing the data by labeling the control modes (regulating, tracking, free) produces a more clear relationship between driving conditions and performance.

average difference between outdoor and indoor is 20°F? Figure 6.3 shows this relationship for a single week for a single system. As expected, the indoor temperature rise during regulating modes is near zero; the regulating mode is defined as a period in which the temperature is being maintained constant. However, the indoor temperature rise during free response and tracking periods is of



(a) Time series showing one week of indoor temperature with modes labeled.



(b) Relationship between indoor temperature rise and temperature difference using the time series in (a). Each point represents a 1-hour period.

Figure 6.3: Another example of how preprocessing the data by labeling the control modes (regulating, tracking, free) produces a more clear relationship between driving conditions and performance.

interest. Figure 6.3 shows that, as expected, the indoor temperature falls during cool tracking period and rises during free response periods. The rise in indoor temperature when the system is off provides insight to the construction of the home (e.g. surface area, insulation quality), and the fall in indoor temperature when the system is on provides insight to the cooling capacity relative to the heat load. When a system is unable to maintain its setpoint (such as that shown in Figure 6.2), the average drop in indoor temperature during tracking periods will be near zero.

6.4 The Mode Labeling Algorithm

This section describes the logic used to filter raw thermostat data into operating modes. However, the nature of this raw data must be understood in greater detail. Like much cloud-based data, thermostat data is *event-based*. This means that data is added when some type of change occurs; e.g. the thermostat is changed, the system starts/stops to run, the indoor temperature changes by a certain amount. This event-based methodology is far more space-efficient than evenly sampled time-series data and preprocessing methods should be designed to work with this form of data. While event-based data can be transformed to evenly sampled data using interpolation methods, this is not efficient data handling practice and is best avoided.

One challenge with event-based data is the possibility of 'missing' an event when there is a lapse in internet connection. For example, if the cloud-based database fails to capture that the the system has turned on, then the indoor temperature may be decreasing even though it is 100°F outside and the system appears off. Similarly, if the cloud-based database fails to capture a setpoint change, then the system may regulate around 76°F when the setpoint appears to be 70°F. For this reason, the mode labeling algorithm avoids dependency on the setpoint; only the indoor temperature, the mode, and their associated timestamps are necessary.

The purpose of the mode labeling algorithm is to classify each cycle (both on and off cycles) as either a tracking, regulating, or free response cycle. Intuitively, one can understand that tracking periods will only contain 'on' cycles, and free response periods will only contain 'off' cycles. Regulating periods, on the other hand, will contain both 'on' and 'off' cycles. Tracking and regulating modes can naturally be bisected into heating and cooling by simply observing whether the 'on' cycles involves heating or cooling.

6.4.1 Algorithm Overview

Figure 6.4 provides an overview of the mode labeling algorithm. The algorithm transforms the thermostat indoor temperature and mode (and their associated timestamps) to the transformed modes. In addition to the data, the algorithm requires two thresholds (denoted α and β in Figure 6.4). The algorithm also includes several functions, and a rough outline is provided below. Further details are then provided for the functions.

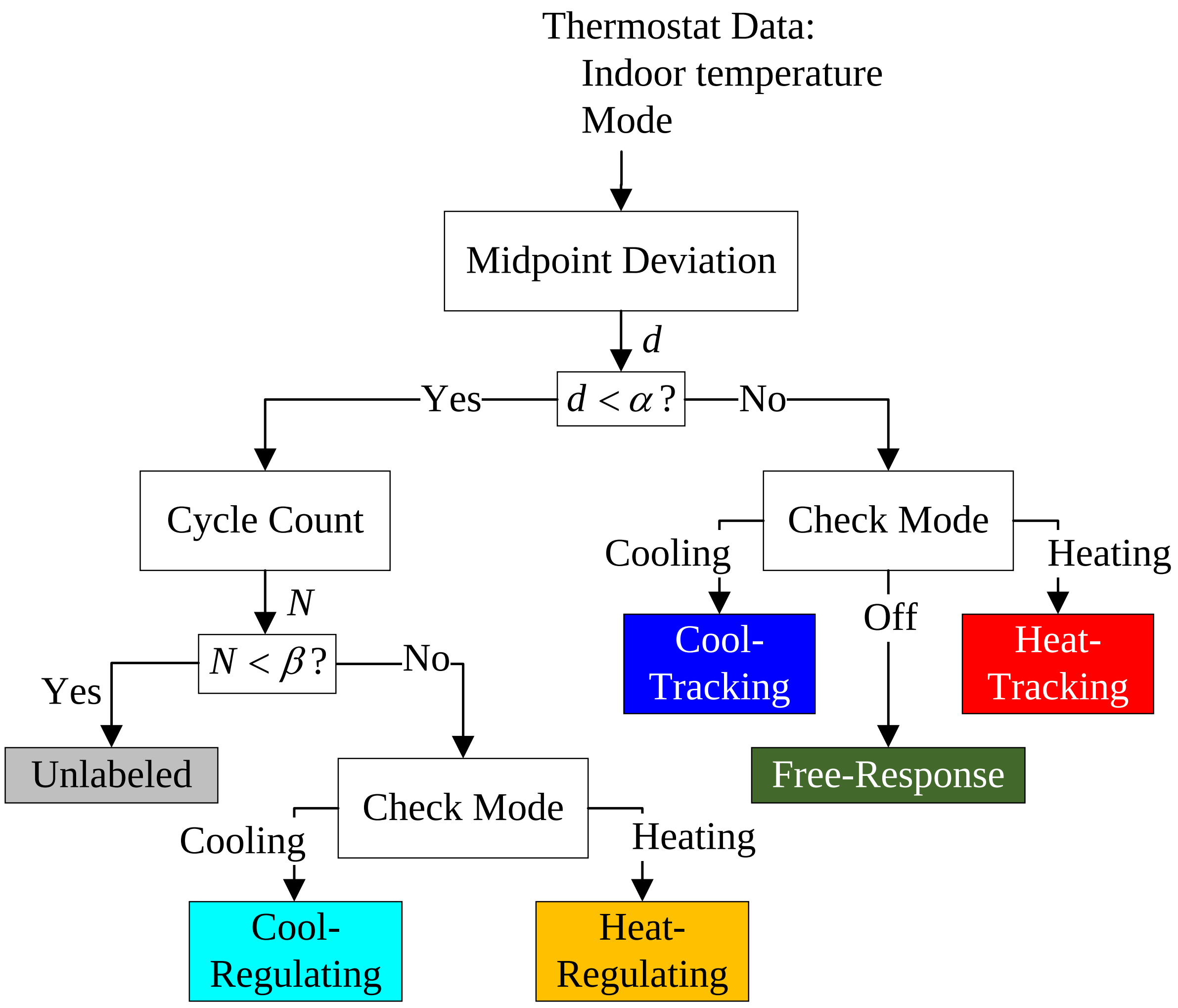


Figure 6.4: An overview of the mode labeling algorithm.

1. The Midpoint Deviation function calculates a deviation d for each 'on' and each 'off' cycle. This deviation is essentially a comparison between the indoor temperature of the current cycle to the indoor temperature of the cycles before and after the current.

2. A sufficiently low deviation ($d < \alpha$) indicates that the indoor temperature of the current cycle is similar to that of the neighboring cycles. If the deviation is sufficiently low, the Cycle Count function is used to determine the number of consecutive cycles that are similar.

3. A high deviation ($d \geq \alpha$) indicates that the indoor temperature of the current cycle is different from the indoor temperature of neighboring cycles, and the transformed mode will be tracking or free response. The Check Mode function then simply determines the transformed mode based on the raw mode.

4. The β threshold enforces a minimum number of cycles required for regulating modes; i.e. the system must be cycling on and off around a constant temperature for β cycles before this period will be labeled as regulating. For periods with a sufficient number of cycles, the Check Mode function is used to distinguish cool-regulating from heat-regulating.

6.4.2 Midpoint-Deviation

As stated previously, the midpoint deviation d is a measure of how a given cycle compares to its neighboring cycles. More specifically, the midpoint deviation is the maximum difference between the midpoint temperature of a given cycle and the extremums of the cycles both preceding and following the given cycle. This could be expressed as:

$$\begin{aligned} d = \max\Big(&T - \max(\mathbf{T}_{pre}),\ \ T - \max(\mathbf{T}_{post}), \\ &\min(\mathbf{T}_{pre}) - T,\ \ \min(\mathbf{T}_{post}) - T\Big) \end{aligned} \tag{6.1}$$

where T is the midpoint temperature of the cycle-of-interest, $\mathbf{T}_{pre}$ is the set of temperatures preceding the current cycle, and $\mathbf{T}_{post}$ is the set of temperatures following the current cycle.

Figure 6.5 provides two visual examples of how this midpoint deviation is calculated. In both of

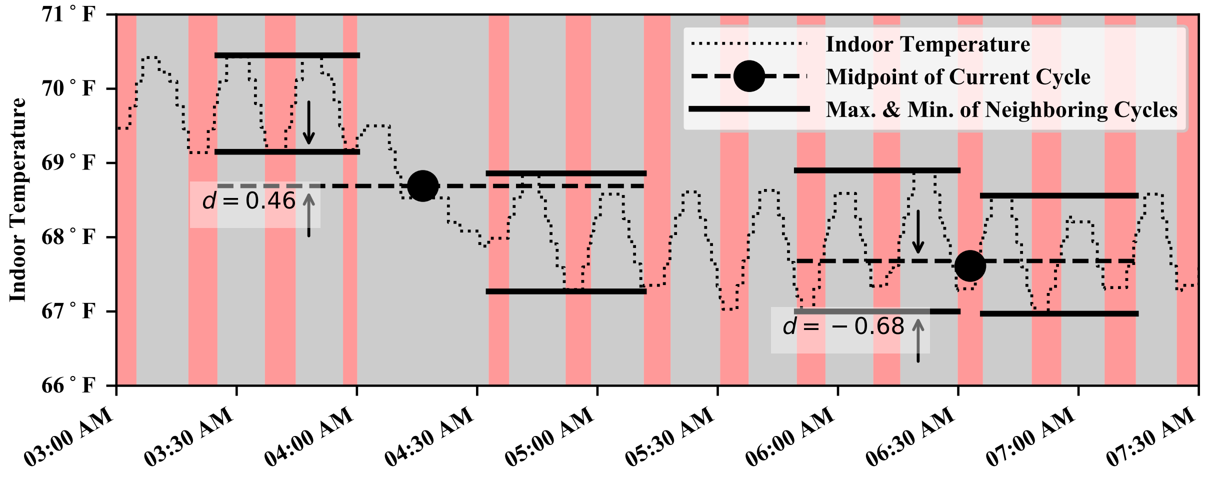


Figure 6.5: Two examples of how the midpoint deviation d is determined. Red shading indicates heating cycles, and gray shading indicates off cycles. The midpoint deviation is the *maximum* distance between the *midpoint* of the current cycle and the *range* of the neighboring cycles (i.e. the distance above the maximum OR the distance below the minimum). When the midpoint is contained within the neighboring ranges, then the midpoint deviation is negative. The examples above use four neighboring cycles before and four neighboring cycles after the current cycle.

these examples, the maximum deviation is the difference between the midpoint and the minimum of the preceding temperatures. The example on the left shows a positive midpoint deviation and the example on the right shows a negative midpoint deviation. Recalling Figure 6.4, a midpoint deviation less than the threshold α will be further processed for regulating modes. A midpoint deviation greater than α will either be a tracking or free response mode. A threshold of $\alpha = 0.1$ performs well and has been used for the analysis presented in this paper. The example on the left in Figure 6.5 is therefore a free response mode because the system is off and the midpoint deviation is greater than α. The example on the right may be a heat-regulating period, but this depends on the when the cycle count N is greater than the threshold β.

6.4.3 Cycle-Count

As stated previously, the cycle count N is the number of consecutive cycles that are similar. More specifically, the cycle count is the number of consecutive cycles that have a midpoint deviation d less than threshold α. Only periods in which a sufficient number (β) of consecutive cycles have a low midpoint deviation are eligible candidates for regulating modes. This condition ensures that regulating modes have at least β cycles.

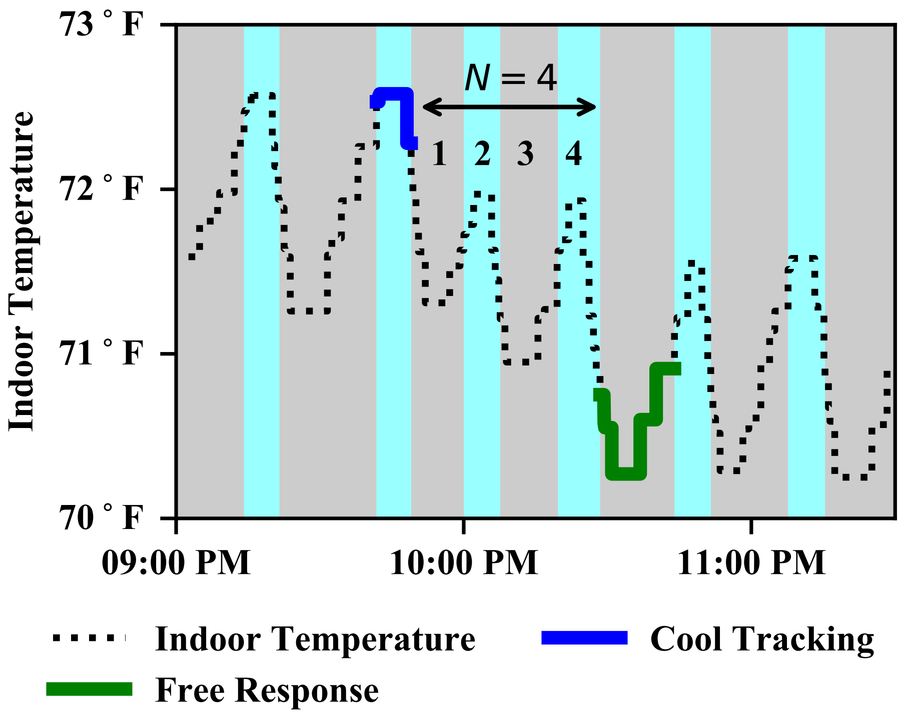


Figure 6.6: An example of how the cycle count N is calculated. Having determined the midpoint deviation, the tracking and free response modes are labeled. The cycle count is then the number of consecutive cycles in between tracking and free response modes.

Figure 6.6 provides an example of how the cycle count is calculated and how this can be useful. In Figure 6.6, the cooling setpoint seems to change from 72°F to 71°F just before 10 PM. Rather than achieving the new setpoint in a single run, the air conditioning system cycles on and off and the average indoor temperature approaches the setpoint. Without a limit on the number of consecutive cycles, this period would incorrectly be labeled as regulating.

6.5 Mode Features

The previous sections have described a mode labeling algorithm that transforms the raw thermostat mode for cycling systems into a set of more useful modes. Ideally, the air conditioning operation would be similar within all regulating modes and within all tracking modes, and the home's response would be similar within all free response modes. In order to quantify this system behavior, a set of features is necessary in addition to the mode labeling algorithm.

Table 6.1 lists several features and the modes for which the features are applicable. Table 6.2 describes each of these features. Some of these features were used earlier in Figures 6.2 and 6.3. Figure 6.2 showed the relationship between temperature difference and duty cycle, and Figure 6.3 showed the relationship between temperature difference and temperature rise. The duty factor for tracking and free response period is not very interesting because it is one and zero respectively. Similarly, the temperature rise is not very interesting for regulating periods because it is centered at zero.

Table 6.1: Supporting Features for Each Mode

Symbol	Feature	Tr.	F.R.	Reg.
ΔT_{oi}	Temperature Difference	X	X	X
ΔT_{ii}	Temperature Rise	X	X	
D	Duty Factor			X
F	Cycle Frequency			X
L	Load Factor	X		X
μ_{ec}, μ_{eh}	Setpoint Error (mean)			X
σ_{ec}, σ_{eh}	Setpoint Error (st. dev.)			X

The presented list of features does not include features associated with humidity such as the rise in indoor humidity and the humidity difference between outdoor and indoor. These humidity features will be more significant when there is a drastic difference between indoor and outdoor humidity, and should be explored in the future. The list of features also does not include the time interval over which the features are averaged. The features shown in Figures 6.2 and 6.3 are averages for rolling one-hour periods. Longer periods (an hour or more) are appropriate for regulating and free response periods which will often persist for several hours. However, shorter time periods could be used for tracking modes because they consist of a single 'on' cycle and will

Table 6.2: Mode Labeling: Definition of Features

Symbol	Definition
ΔT_{oi}	Average temperature difference between home and ambient; defined as $\Delta T_{oi} = \bar{T}_o - \bar{T}_i$.
ΔT_{ii}	Temperature rise (or drop) in a given time period: $\Delta T_{ii} = T_i(t_2) - T_i(t_1)$ for time period t_1 to t_2.
D	Duty factor. This is the portion of time that the system is running in a given time period0.
F	Cycle frequency. This is the number of start/stop cycles in a given time period.
L	Load factor. This is the average portion of full capacity at which the system runs in a given time period (when the system is running). When multiplied by the duty factor D, the result LD denotes the overall portion of capacity for the given time period.
μ_{ec}, μ_{eh}	Average cooling and heating setpoint error respectively. Defined as the average difference between T_i and the respective setpoint.
σ_{ec}, σ_{eh}	Standard deviation of the setpoint error during a given time period.

generally be less than an hour.

6.6 Case Studies

The transformed modes and their associated features could be used to identify faulty systems in two ways. First, faulty systems may be identified by analyzing the time series data for individual systems and determining when the performance has degraded. This is referred to as *single-system degradation detection*. Second, faulty systems may be identified by comparing performance across many systems and then identifying poorly performing systems. This is referred to as *multi-system performance comparison*. An example for each of these approaches is provided in Figures 6.7 and 6.8.

6.6.1 Single-System Degradation Detection

Figure 6.7 shows the relationship between temperature difference and duty cycle during cool-regulating modes for a single-stage system that has a fairly obvious level of performance degradation. Two weeks of data are plotted, and the weeks are approximately three months apart. The system has a significantly higher duty cycle in September than in June. This degradation could

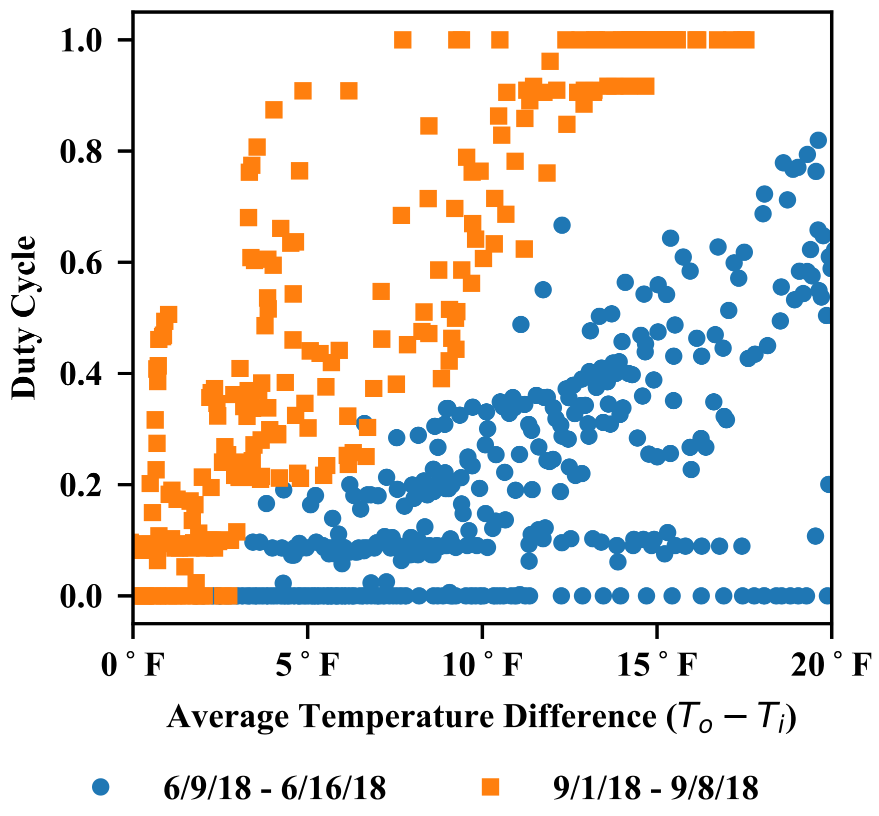


Figure 6.7: An example of how the transformed modes highlight degradation in a single system. The duty cycle during 1-hour cool-regulating periods is plotted for two different time periods for the same single-stage system. More cooling is used the later time period for the same temperature difference.

have also been captured using the tracking modes. The system uses constant setpoint such that there are nearly zero tracking modes when then system is performing well. However, as the the performance degraded and the system was unable to maintain the setpoint during high cooling loads, the system began entering the tracking mode in order to try to recover the setpoint. This is similar to the tracking mode behavior that was observed in Figure 6.2.

6.6.2 Multi-System Performance Comparison

Figure 6.8 shows the relationship between indoor temperature rise and temperature difference during free response modes for two different systems. The homes in which these systems are

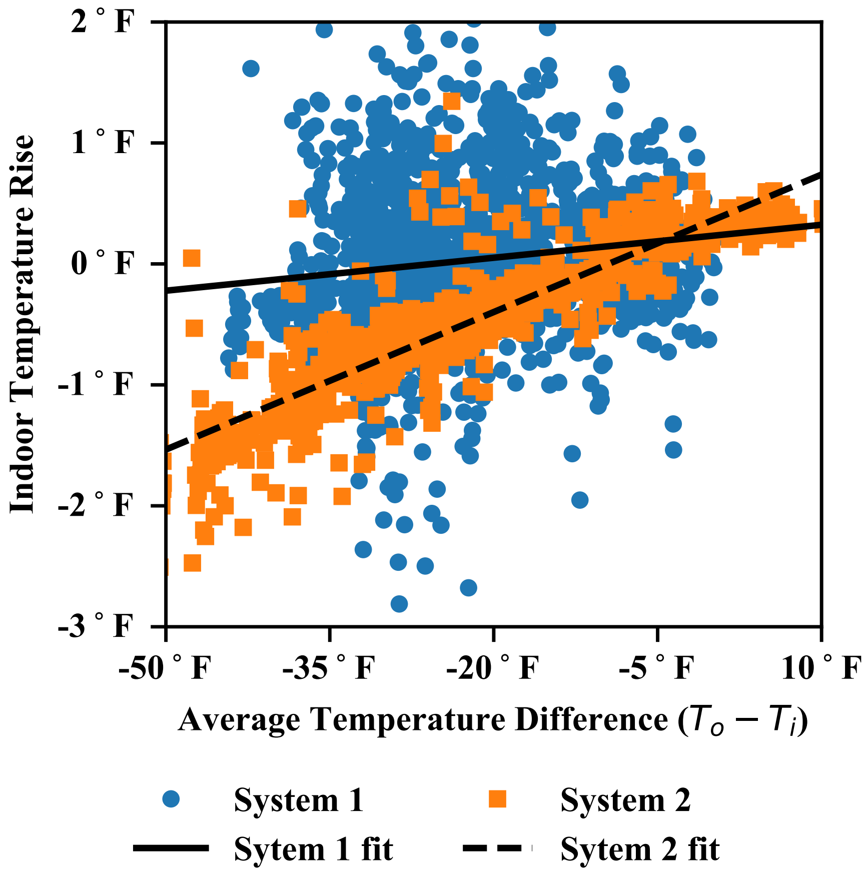


Figure 6.8: An example of how the transformed modes highlight differences between systems. The indoor temperature rise for 1-hour free response periods is plotted for two different systems for September 2018 through November 2018.

installed have clearly different responses to the outdoor conditions. The first system has a large spread in the data while the seconds system shows a more clearly linear relationship. On average, the outdoor temperature has a strong effect on the second system and a fairly weak effect on the first system. A large spread in the data may indicate a higher thermal heat capacity and a stronger relationship between difference and rise may indicate lower thermal resistance in the walls. The free response analysis introduced here only provides insight to the home construction and loading behavior and does not provide insight to the performance of the HVAC system. However, the free response analysis is critical in order to provide context when analyzing the HVAC performance. For example, the apparent performance degradation shown in Figure 6.7 may have been caused simply by home owner leaving all the windows and doors open rather than any faults within the HVAC system. If this were the case, then analyzing the free response modes would reveal that the home's response to the outdoor temperature has changed significantly.

6.7 Conclusions

The mode labeling algorithm presented in this paper is a preprocessing method used to transform raw thermostat modes into modes that are more useful for data analysis. The presented case studies highlight this preprocessing method's usefulness in observing both single-system degradation and in comparing performance across different systems. The mode algorithm presented and used in this analysis contains the essential pieces. However, the algorithm could be improved in several simple ways.

1. The algorithm presented here uses only the indoor temperature and the raw modes of operation. The thermostat setpoint could be used to improve the logic. For example, when the setpoint changes, the system should not be in a regulating mode. The system will either enter a tracking mode or a free response mode until the new setpoint is obtained.

2. The cycle time could be used to improve the mode labeling algorithm. For example, free response modes could be limited to 'off' cycles with a minimum length, and regulating modes could be limited to cycles less than a maximum length.

3. The free response and tracking modes should be modes in which the indoor temperature is changing rather than oscillating around a mean. Therefore, a threshold could be placed on the change in indoor temperature during the cycle such that these modes contain cycles in which the indoor temperature is changing significantly.

This paper has focused on the preprocessing method and has not presented fully developed fault detection methodology using this preprocessing method. Much work must still be done in order to fully take advantage of the presented preprocessing algorithm. Further data analysis may be built on this preprocessing algorithm such as change-point detection algorithms, multi-variate statistical approaches, and machine learning methods.

7. PAPER E: A CHANGE POINT DETECTION ALGORITHM WITH APPLICATION TO SMART THERMOSTAT DATA[1]

7.1 Synopsis

Applying change point detection to residential smart thermostat data is an important step towards performing fault detection using this data. However, most change point detection algorithms require either that the nominal operating conditions be known or that parameters be tuned carefully in order for the algorithm to succeed. A change point detection algorithm for smart thermostat data must be simple in both tuning requirements and computational load, and the algorithm must be capable of being deployed across thousands of systems simultaneously. The change point detection method presented and applied to smart thermostat data in this paper is based on the t-statistic which is commonly used for hypothesis testing when sample sizes are small. The advantage of using the t-statistic is that it conveniently accounts for situations when little data is available for the nominal operating behavior. The resulting algorithm is robust and easy to implement. Monte Carlo methods are used to determine appropriate thresholds and evaluate the effectiveness of the algorithm in terms of how the Type II error rate increases as both the sample size and the signal-to-noise ratio decreases. The algorithm is applied to smart thermostat data both retrospectively and recursively and several interesting case studies are presented. Finally, a methodology for handling multiple change points and false positives is presented.

7.2 Introduction

Residential and commercial buildings account for 40% of total U.S. energy consumption [103], and approximately 48% of this energy is due to heating, ventilation, and air conditioning (HVAC) systems [2, 3]. Several studies have found that HVAC systems commonly operate for an extended period with one or more faults present, and that these faults have a substantial impact on the system efficiency [12, 11, 104, 10]. One method to reduce the energy consumption in these systems is to

[1]Rogers, A.P., and F. Guo, and B.P. Rasmussen. A Change Point Detection Algorithm with Application to Smart Thermostat Data. **In Review**.

implement automated fault detection and diagnosis (FDD) methods. With effective FDD methods in place, faults may be serviced with minimal delay in order to maintain the design efficiency over time.

The typical fault detection and diagnosis process for air conditioning systems involves the selection of features derived from measurements and fault-free models, and then the application of fault detection and fault diagnosis classifiers to the selected features [51]. The goal of the fault detection classifier is to detect with confidence when one of the features has had a significant change. These classifiers are often statistics-based and proposed methods include a Bayesian classifier [37], the *Normalized Distance Fault Detection Classifier* [21], and the Generalized Likelihood Ratio (GLR) test [29, 28, 57]. These methods all require important assumptions about the nominal operating point, but in many applications this nominal operating point will be unique to a specific installation. There is therefore value in developing fault detection classifiers that require minimal assumptions and tuning and that may be easily implemented across many different systems.

Minimizing the required tuning is particularly important if an algorithm is to be deployed across a large scale data set such as a database of smart thermostat data. The application of fault detection methods to thermostat data is still emerging [79, 77, 80, 31, 78, 32], and this current body of literature has not reached the scale of data that will be possible over the next decade. A smart thermostat database may contain data from hundreds of thousands of thermostats, but the data for each thermostat is relatively sparse. Smart thermostat data typically includes indoor temperature, system status (e.g. cooling, heating, off), indoor relative humidity, and an estimate for the outdoor temperature which is usually obtained from open sources such as the National Oceanic and Atmospheric Administration (NOAA). The parameters included in smart thermostat data are important factors for describing the operation of an air conditioning or heating system, but there are many other factors involved such as the solar irradiation and home construction [105, 106], the number of occupants, and the presence of other heat sources such as plug loads. The lack of data for these other important factors results in a high level of uncertainty in the observed relationship between smart thermostat data streams. The fault detection classifier employed for analyzing smart thermostat data must be able to deal with this high level of uncertainty and be simple enough to be deployed across many systems.

In other fields, change-point detection algorithms perform the role of fault detection classifiers [107, 108, 109, 110]. Change point detection has been explored for decades and continues to be an active area of research. [111] provided an introduction to early methods such as the cumulative sum (CUSUM) algorithm and the Generalized Likelihood Ratio (GLR) algorithm. As stated previously, the GLR algorithm has been applied to HVAC chillers [29, 28, 57], but this algorithm requires that the nominal distribution be known. In many applications such as smart thermostat data, the nominal distribution will not be known. For thermostats this is largely because the nominal cooling load depends on many unknown factors and will therefore vary widely from one home to the next.

More recent change point detection methods characterize the subspace of the data through principal component analysis (PCA) [112], singular spectrum analysis (SSA) [113, 110], or subspace identification [109]. Another promising set of methods is based on the direct estimation of the probability density ratio [114, 115, 116]. This is formulated as a convex optimization problem that is solved at each time step for recursive change point detection. In addition to the required optimization, these density ratio methods involve tuning parameters that have a large impact on the result of the algorithm. The ideal change point detection method for smart thermostat data must be simple in both computational load and tuning parameters. The computation required to identify the subspace and the tuning required for the density ratio estimation is not ideal for the application of smart thermostat data.

This paper presents a robust method designed exclusively for mean change point detection that does not require a known distribution for the nominal operating behavior. The univariate input that is analyzed for change point detection should be nominally constant with Gaussian noise, but no assumption needs to be made regarding it's mean and variance. Implementing the proposed method is simple and the few required parameters may be set universally for all applications. The algorithm accounts for the limited knowledge that may be available for the nominal operating point by using the t-statistic used in hypothesis testing for small sample sizes.

7.3 Overview

The primary purpose of any change point detection algorithm is to confidently identify a change in measured behavior. A recursive algorithm should identify this change as soon as statistically feasible, whereas a retrospective algorithm is designed to analyze a past dataset. The proposed method is based on the t-test that is commonly used to determine whether two samples are chosen from the same population. The t-test was specifically designed to test small sample sizes, and the test statistic therefore accounts for the inherently limited knowledge. Using a t-statistic for the purpose of change point detection involves systematically dividing a time series into possible samples, and the bisection that yields the highest t-statistic will be associated with the instant in time at which a change is most likely. This maximum t-statistic (MTS) will not follow the traditional Student's t distribution, but a simple Monte Carlo method is used with Kernel Density Estimation (KDE) to identify its probability density and appropriate thresholds for the maximum t-statistic are then chosen.

Section 7.4 explains the proposed method in detail; Section 7.4.1 reviews the t-statistic, Section 7.4.2 introduces the process of determining the t-statistic at all possible bisections in a time series in order to find the maximum, Section 7.4.3 explores the distribution of this test metric using a Monte Carlo method and then identifies appropriate thresholds, and Section 7.4.4 shows how the method may be vectorized for efficient deployment. Section 7.5 demonstrates the algorithm's effectiveness in simulation using a Monte Carlo method, and Section 7.6 applies the algorithm to smart thermostat data and highlights several detected change points. Finally, Section 7.7 presents a methodology for accounting for multiple change points.

7.4 The Change Point Detection Algorithm

7.4.1 The t-Statistic

The t-statistic is used to compare two samples when the sample sizes are not large. A common way to define the t-statistic for two samples X and Y with means $\bar{X}$, $\bar{Y}$, standard deviations s_X,

s_Y, and sizes n_X, n_Y is given in Equation 7.1 [117].

$$t = \frac{\bar{X} - \bar{Y}}{\sqrt{s_X^2/n_X + s_Y^2/n_Y}} \tag{7.1}$$

This definition of the t-statistic is troublesome when samples of X and Y have drastically different sizes. For example, consider $n_X \gg 0$ and $n_Y = 2$. The denominator of Equation 7.1 may be near zero if $s_Y \approx 0$. While this edge case may be of little concern in typical applications of the t-test, it will be of great concern here. When populations X and Y have nearly equal variances, the t-statistic may be defined in terms of a pooled variance as shown in Equations 7.2 and 7.3 [117].

$$t = \frac{\bar{X} - \bar{Y}}{s_p\sqrt{1/n_X + 1/n_Y}} \tag{7.2}$$

where the pooled variance s_p^2 is defined as:

$$s_p^2 = \frac{(n_X - 1)s_X^2 + (n_Y - 1)s_Y^2}{n_X + n_Y - 2} \tag{7.3}$$

This second definition for the t statistic will not become very large when sample sizes are disparate because pooling the sample variances weighs them according to their respective sample size.

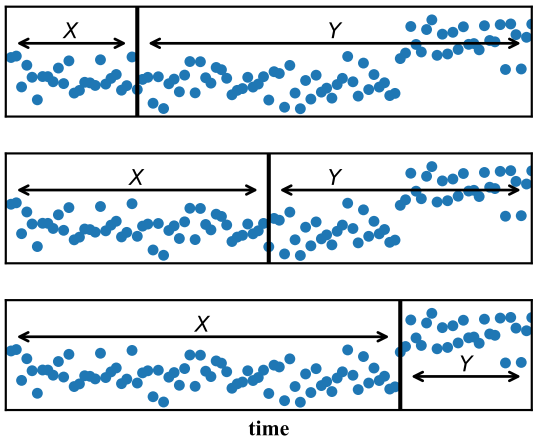
7.4.2 Bisecting a Time Series

In order to apply the t-test to time series data, the time series must be bisected into two samples. Let the samples X and Y be portions of a time series, and let the samples of X occur before Y. In order to detect a change in mean as soon as possible, the samples X and Y cannot simply be randomly drawn from the time series. Rather, every possible bisection of the algorithm should be considered. This is illustrated in Figure 7.1a. Figure 7.1a shows three possible bisections for a given time series. This time series consists of randomly generated data with an artificial change in mean after approximately 75% of the time has elapsed. The proposed change detection method involves calculating the t-statistic for each possible bisection. The bisection with the highest t-statistic is associated with the point at which a change in mean *is most likely.* In Figure 7.1a, the lowermost bisection is expected to yield the highest t-statistic.

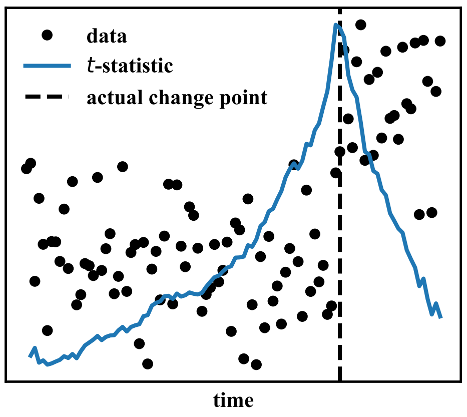
Figure 7.1b shows the calculated t-statistic at each possible bisection for the time series shown in Figure 7.1a. As expected, the maximum t-statistic (MTS) occurs at the actual change point. This MTS is likely much higher than the maximum would have been without the change point. In order to use the MTS as a metric for change point detection, the expected MTS under the condition that there is no change must be determined. The following section determines the expected MTS under the condition that there is no change using a Monte Carlo method.

7.4.3 Maximum t-Statistic Distribution

As shown in the previous section, the maximum t-statistic (MTS) is determined by systematically calculating the t-statistic for all possible bisections in a time series. The distribution of the MTS when there is no change present must be determined in order to determine the likelihood of encountering a given MTS. Similar to how the Student's t distribution depends on the degrees of freedom, the MTS distribution will also depend on the size of the time series tested.



(a) Three example bisections for a given time series. The t statistic could be calculated given the samples X and Y, and the bottom plot will likely yield the greatest t-statistic because the time series is bisected at the change point.



(b) Calculated t-statistic at each possible bisection for the same time series shown in (a). The maximum t-statistic (MTS) appropriately occurs at the location of the change in mean.

Figure 7.1: An overview of how the Maximum t-Statistic (MTS) is calculated; (a) a time series is bisected and (b) the t-statistic is calculated at all possible bisections.

The MTS distribution has been determined by generating many time series of size n from a randomly distributed population, and then calculating the MTS for each sample. Note that n is the size of the full time series, i.e. $n = n_X + n_Y$. Intuitively, one may suppose that the MTS is more likely to occur at certain locations in the time series. In other words, given a randomly generated time series, the maximum value of the t-statistic may be more likely to occur when the time series is bisected near an edge rather than in the center. Or perhaps the reverse is more likely. However, the Monte Carlo approach shows that the distribution of the t-statistic is the same at all bisections and therefore *the maximum is equally likely to occur at any location*. This is shown in Figure 7.2. In Figure 7.2, 100,000 time series of size 100 are generated and the 90th, 95th, 99th, and 99.5th quantiles of the t-statistic at each possible bisection are plotted. Note that Figure 7.2 shows how the distribution of the t-statistic at each location is the same. This unbiased nature of the t-statistic is the primary advantage in using the t-statistic rather than assuming knowledge of the nominal operating behavior; the t-statistic appropriately accounts for the limited knowledge available on the left-hand side (sample X).

Having shown that the MTS is not biased with respect to the location of the bisection, the distribution of the MTS when there is no change may be determined using a similar Monte Carlo method. For several values of n, 20,000 time series were generated, and the MTS was determined for each time series. This results in 20,000 values of an MTS for the given size n. A probability density function is then generated using kernel density estimation (KDE). Figure 7.3 shows the probability density function of the MTS for n=10, 20, 30, 150, and 200. The distribution's dependency on n evidently decreases as n becomes large indicating that applying the distribution for $n = 200$ to higher values of n would be appropriate.

The MTS distributions shown in Figure 7.3 are useful for understanding the expected value of the MTS provided that there is no change. These distributions may be used to determine appropriate thresholds for the MTS given the size of the time series n. Figure 7.4 shows the 99.9th quantile of the MTS for several values of n. When the time series is small (e.g. $n = 10$), the MTS has greater variance, and an MTS greater than 8.2 is required in order to exceed the threshold. Note

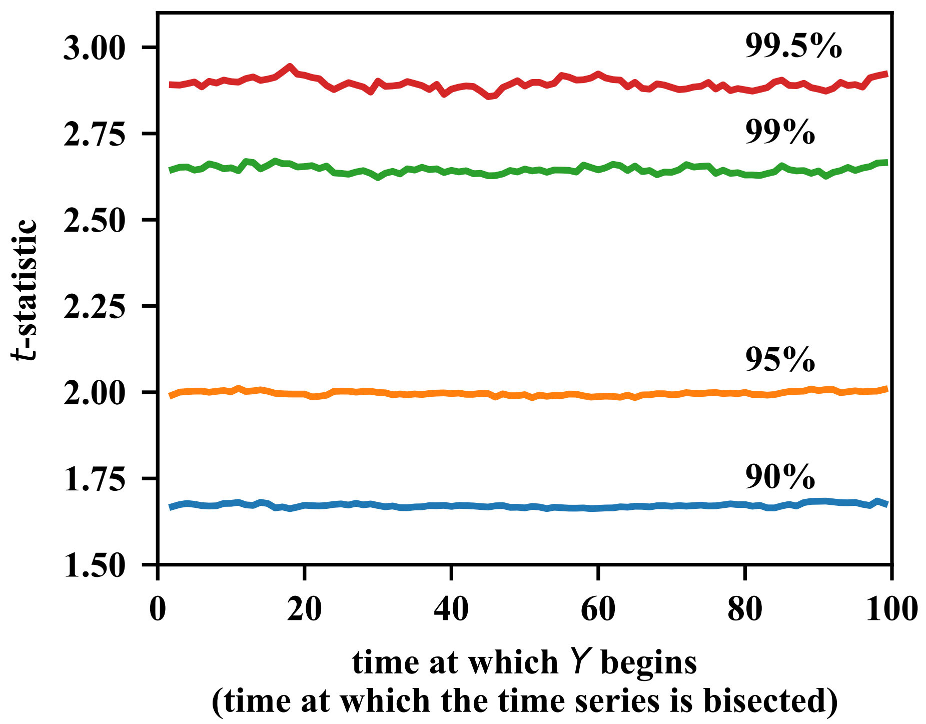


Figure 7.2: Several quantiles of the t-statistic for 100,000 randomly generated time series with $n = 100$. The distribution of the t-statistic is relatively independent of the location in time at which the bisection occurs. The maximum t-statistic (MTS) is therefore equally likely to occur at any bisection provided that there is no change in mean.

that the distributions are not required when implementing this method. The distributions are only used to determine an appropriate threshold, and the method may be implemented using only the thresholds.

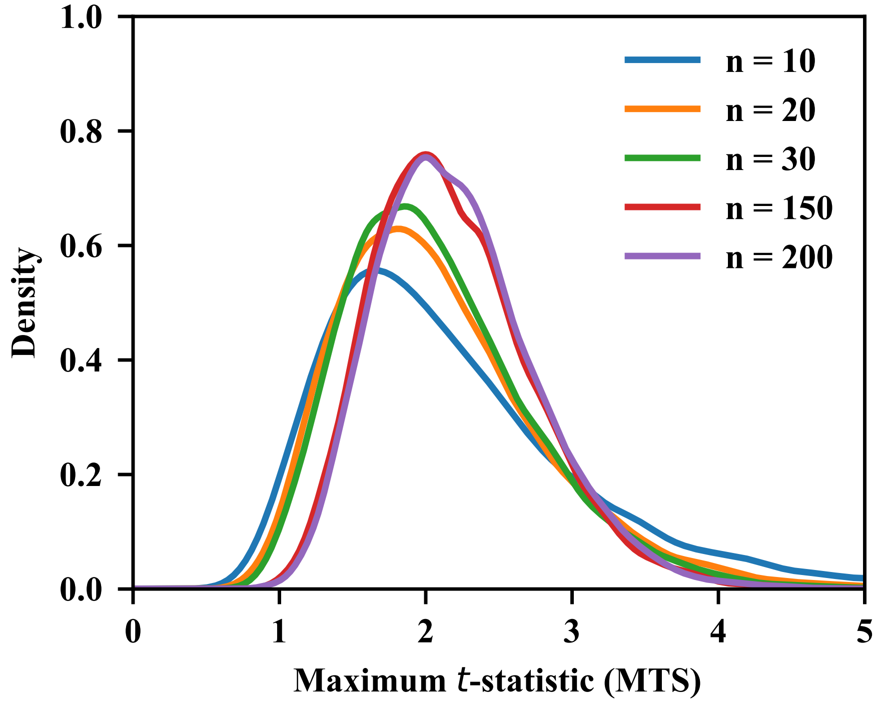


Figure 7.3: Probability density function of the maximum t-statistic (MTS) when there is no change.

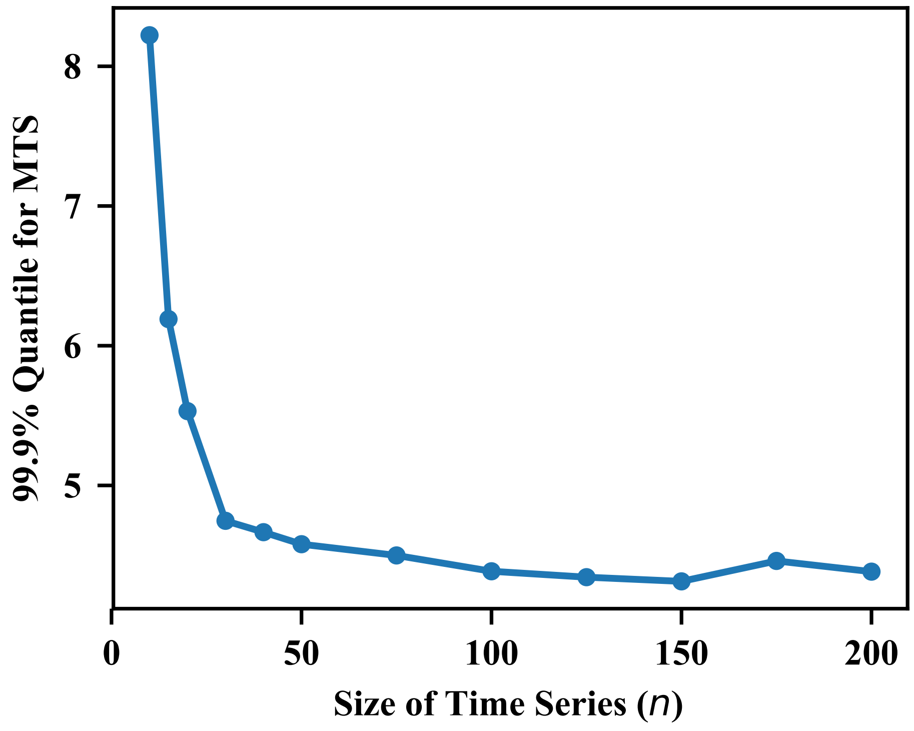


Figure 7.4: The threshold for the maximum t-statistic for $10 \leq n \leq 200$ that results from using the 99.9% quantile.

7.4.4 Computational Efficiency

The proposed process of calculating a t-statistic at each possible bisection of a time series may initially seem computationally expensive. However, this section will show how the bisection process may be vectorized using `cumsum` operations in order to minimize the computational load. For this formulation, the t-statistic will be generalized such that variable weights may be used, i.e. some of the data may exercise a greater influence on the mean and standard deviation. The purpose of generalizing the formulation with variable weights is to provide a general formulation. The non-weighted formulation may be recovered simply by applying the same weight to all data points. The typical definition for the weighted mean is as follows.

$$\bar{x}_w = \frac{\sum_{i=0}^{n-1} w_i x_i}{\sum_{i=0}^{n-1} w_i} \tag{7.4}$$

The following non-standard weighted variance will also be used.

$$s_x^2 = \frac{\sum_{i=0}^{n-1} w_i (x_i - \bar{x}_w)^2}{\sum_{i=0}^{n-1} w_i} \tag{7.5}$$

When using these variable weights, the samples sizes n_X and n_Y may be defined as the sum of the weights. Note that the sample variance is typically defined with $n-1$ in the denominator. However, the weighted variance above only uses Σw in the denominator which is akin to using n instead of $n-1$. This allows the variance to be defined when $n=1$ and therefore a t-statistic will be defined when the right sample has a size of one ($n_Y = 1$). A change may then be detected even after the addition of a single data point (if that data point is significantly disparate from previous ones).

Consider Table 7.1 which includes an example of how the t-statistic is calculated at all bisections for a time series of size 10. A series of size 10 has 9 possible bisections. Therefore the t-statistic must be undefined at one of the time points on the edge. In Table 7.1, the columns Σwx and Σw under the Left-Hand-Side section denote the numerator and the denominator of the

Table 7.1: Example t-Statistic Calculation

					X (Left-Hand Side)					Y (Right-Hand Side)					
time	x	w	wx	wx^2	Σw	Σwx	Σwx^2	$\bar{x}$	s_x^2	Σw	Σwx	Σwx^2	$\bar{x}$	s_x^2	t
0	1.05	1.0	1.05	1.1	1.0	1.05	1.1	1.05	0.0	9.0	12.5	20.77	1.39	0.38	0.52
1	0.87	1.0	0.87	0.76	2.0	1.92	1.86	0.96	0.01	8.0	11.63	20.01	1.45	0.39	1.07
2	0.99	1.0	0.99	0.98	3.0	2.91	2.84	0.97	0.01	7.0	10.64	19.03	1.52	0.41	1.44
3	0.69	1.0	0.69	0.48	4.0	3.6	3.32	0.9	0.02	6.0	9.95	18.55	1.66	0.34	2.5
4	1.04	1.0	1.04	1.08	5.0	4.64	4.4	0.93	0.02	5.0	8.91	17.47	1.78	0.32	3.29
5	1.23	1.0	1.23	1.51	6.0	5.87	5.91	0.98	0.03	4.0	7.68	15.96	1.92	0.3	4.03
6	0.97	1.0	0.97	0.94	7.0	6.84	6.85	0.98	0.02	3.0	6.71	15.02	2.24	0.0	13.32
7	2.16	1.0	2.16	4.67	8.0	9.0	11.52	1.12	0.17	2.0	4.55	10.35	2.28	0.0	3.73
8	2.25	1.0	2.25	5.06	9.0	11.25	16.58	1.25	0.28	1.0	2.3	5.29	2.3	0.0	1.88
9	2.3	1.0	2.3	5.29	10.0	13.55	21.87	1.36	0.35	0.0	0.0	0.0	nan	nan	nan

weighted mean (Equation 7.4) respectively. These values grow larger as the size of the left-hand sample X grows. The corresponding columns under the Right-Hand Side section therefore decrease as the size of the right-hand sample Y decreases.

Note that the column Σw for the left-hand side is simply the cumulative sum of w. Similarly, Σwx is the cumulative sum of wx for the left-hand side. The weighted mean of the left-hand side for all possible bisections is therefore straightforward to calculate using cumulative sums. The right-hand side is nearly the reverse of the left-hand side. For example, Σw for the right-hand samples may be calculated using `sum(w)-cumsum(w)`. Figure 7.5 outlines this process in the Python language and highlights the simplicity with which this may be implemented. Note that, with the formulation shown in Figure 7.5, the maximum t-statistic (MTS) is located at the *last sample before the change*; in Table 7.1, the MTS occurs at time 6 and the change occurs in time 7.

7.5 Effectiveness of the Online Implementation

In order to detect a change point as soon as is statistically feasible, the maximum t-statistic (MTS) must be determined with the addition of each new data point. The process outlined in Table 7.1 and Figure 7.5 must therefore be applied in a rolling manner with window size n. As with any change point detection method, the effectiveness depends on the signal-to-noise ratio

```
def f(x): return cumsum(x)
def g(x): return sum(x) - cumsum(x)
n_X = f(w)
mean_X = f(w*x) / f(w)
var_X = (f(w*x**2) + f(w)*mean_X**2 - 2*f(w*x)*mean_X) / f(w)
n_Y = g(w)
mean_Y = g(w*x) / g(w)
var_Y = (g(w*x**2) + g(w)*mean_Y**2 - 2*g(w*x)*mean_Y) / g(w)
sp = ((n_X - 1)*var_X + (n_Y-1)*var_Y) / (n_X + n_Y - 2)
t = abs(mean_X - mean_Y) / sqrt(sp*(1/n_X + 1/n_Y) )
```

Figure 7.5: Python code for calculating the t-statistic at all possible bisections given data x with weights w. Intermediate variables include the sample sizes n_X and n_Y, sample means $\bar{X}$ and $\bar{Y}$, sample variances s_X^2 and s_Y^2, and the pooled variance s_p^2.

which is the magnitude of the change in mean with respect to the standard deviation.

Figure 7.6 shows an example of the algorithm applied recursively with a change point magnitude of one standard deviation. In this example, a maximum window size of 200 is used. Before

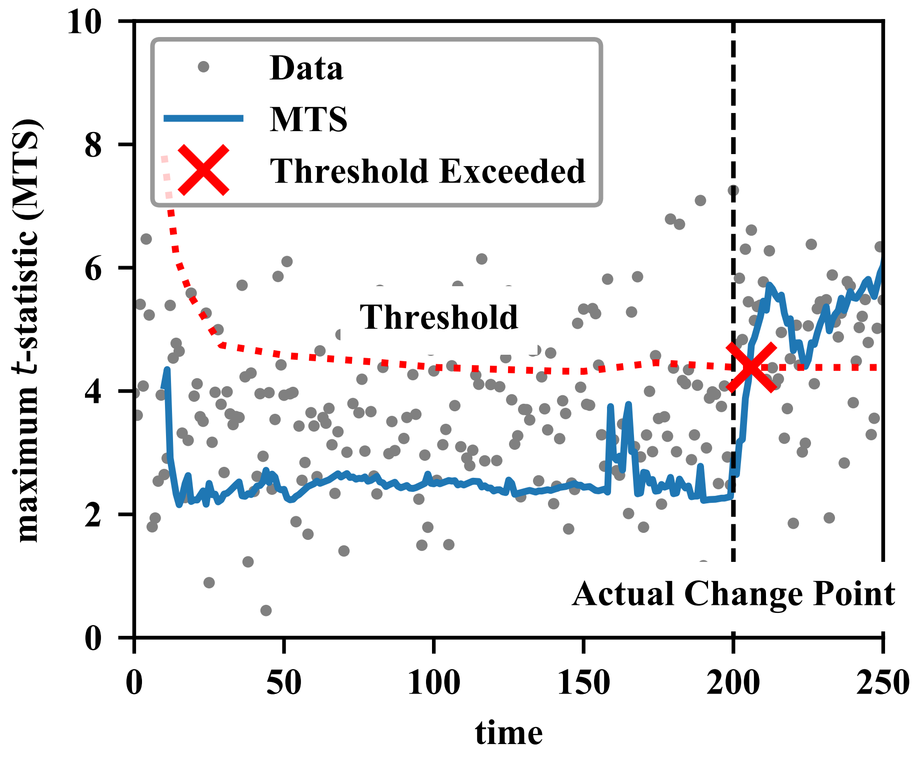


Figure 7.6: Example of online change point detection using generated data, a change in mean of magnitude 1σ, and a maximum window size of 200.

time=10, the maximum t-statistic is not calculated; the thresholds presented in Figure 7.4 only apply to time series of size greater than or equal to 10, and a time series with size less than 10 would have a very high threshold. After time=10, all of the available data is used, and the threshold decreases as the size of the time series grows. However, after time=200, the window rolls such that the window remains a size of 200. The change point is applied at time=200, and the MTS for the past 200 points exceeds the threshold soon after this change point. The actual delay between the change point and the detection of the change point will depend on the specific time series used.

A Monte Carlo method was employed to quantify the typical effectiveness of the online algorithm given various signal-to-noise ratios and various window sizes. Figure 7.7 shows how this is done for a single simulation. As the window rolls through the change point, the change will either be detected with some delay or the change will not be detected at all. If the change is detected

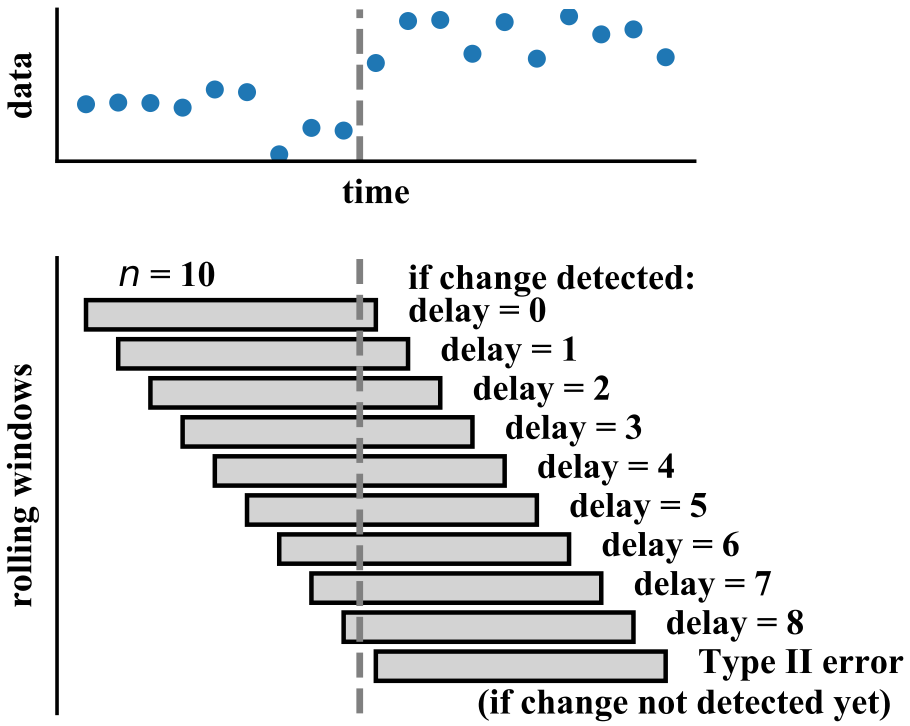


Figure 7.7: An example of how the detection delays and Type II errors are defined for n=10 as the window rolls through the change point. If a change is never detected, then a Type II error has occurred.

with only one data point after the change, then the delay is 0. If the change is never detected by the time the rolling window has completely passed through the change point, then a Type II error has occurred. To evaluate the effectiveness of the algorithm, the analysis shown in Figure 7.7 is repeated many times using generated data at several signal-to-noise ratios and with several window sizes.

Figure 7.8 presents boxplots for the delay between when the threshold is exceeded (change point detected) and when the change point actually occurred. These boxplots are presented for

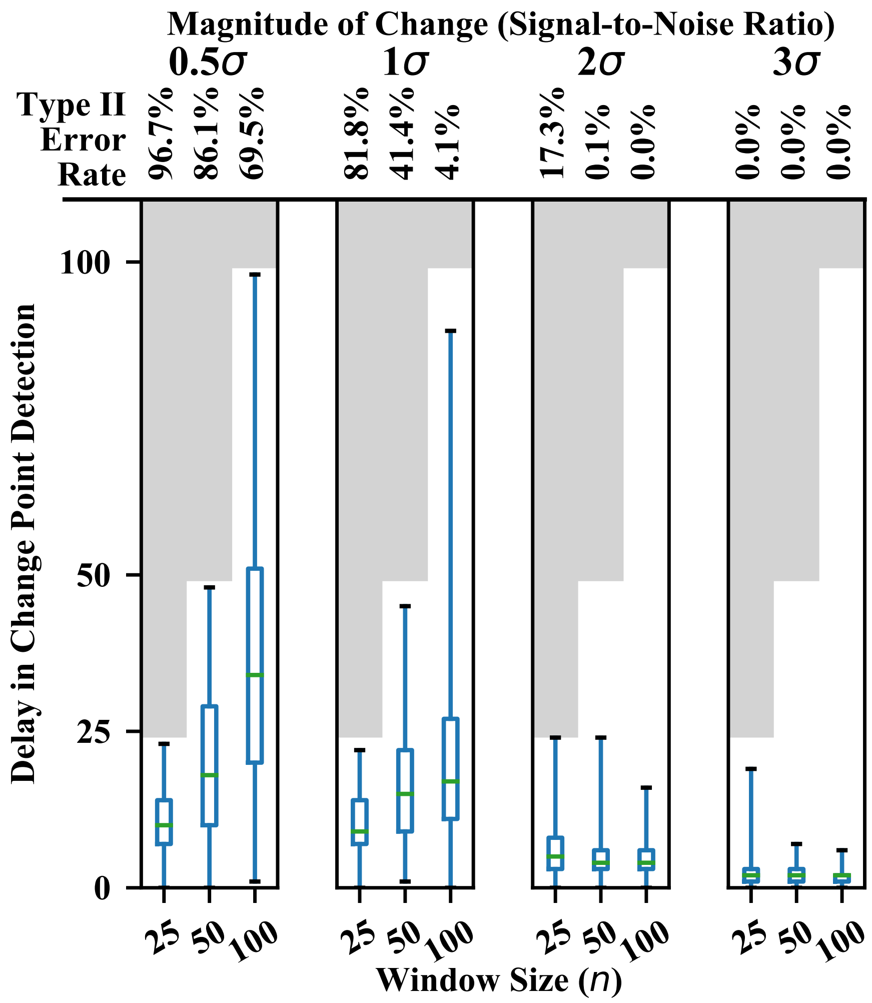


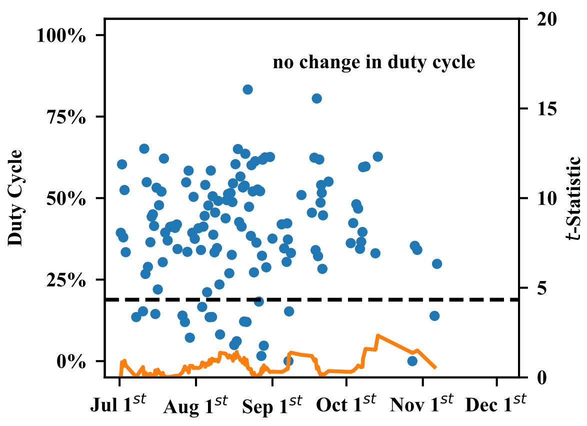
Figure 7.8: Effectiveness of windows sizes of 25, 50, and 100 at detecting a change in mean of 0.5σ, 1σ, 2σ, and 3σ. The boxplots are generated using 1,000 Monte Carlo simulations for each combination, and a Type II error occurs when the change-point is never detected.

window sizes of 25, 50, and 100 and for changes of magnitude 0.5σ, 1σ, 2σ, and 3σ. In this context, a Type II error occurs when the change point is never detected. A Type II error occurs 96.7% of the time when the mean is only 0.5σ and the window size is only 25. However, a Type II error occurs only 4.1% of the time when the mean changes by 1σ and the window size is 100. A Type II error is very unlikely when the mean changes by 3σ even for small window sizes. Given an expected signal-to-noise ratio, Figure 7.8 may be used to determine the required window size.

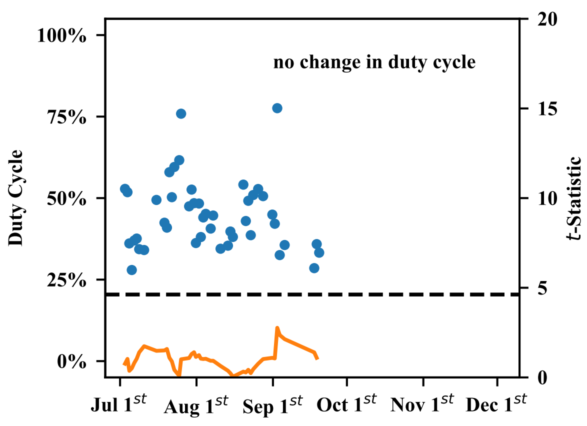
7.6 Application to Smart Thermostat Data

The change point detection algorithm is well-suited for smart thermostat data because it requires minimal tuning and it may be applied to both large and small time series. The algorithm parameters include the threshold and the maximum window size, but the same values for these parameters may be a used for all systems. The algorithm may therefore be applied simultaneously to many different systems. This section presents several examples of applying the algorithm to smart thermostat data. The air conditioning cooling load when the temperature gradient falls within a certain range is used as the change point detection variable. The temperature gradient is the difference between the outdoor temperature (T_o) and the indoor temperature (T_i), and the temperature gradient is typically the most significant driving factor that determines the cooling load. The cooling load will therefore nominally be near constant when the outdoor temperature is 10 °F above the indoor temperature.

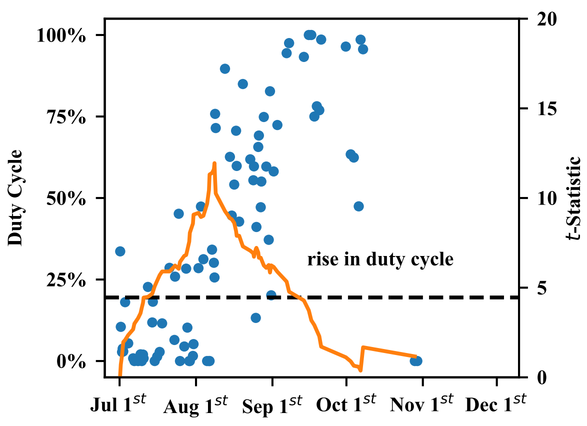
The resulting t-statistic behavior when the change point detection algorithm is applied retrospectively is presented in Figure 7.9 for six homes in Texas. The maximum t-statistic (MTS) behavior for an online implementation is presented in Figure 7.10 for these same six systems. Cases are presented in which no change is detected, a rise in cooling load is detected, and a drop in cooling load is detected. A key point to understand is that changes in the cooling load at a given temperature gradient ($T_o - T_i$) may be the result of changes in occupant behavior (e.g. opening/-closing windows), system faults and degradation, and outdoor conditions other than temperature (e.g. solar irradiation and humidity). A major challenge in performing change point detection for thermostat data is determining the cause of an observed change.



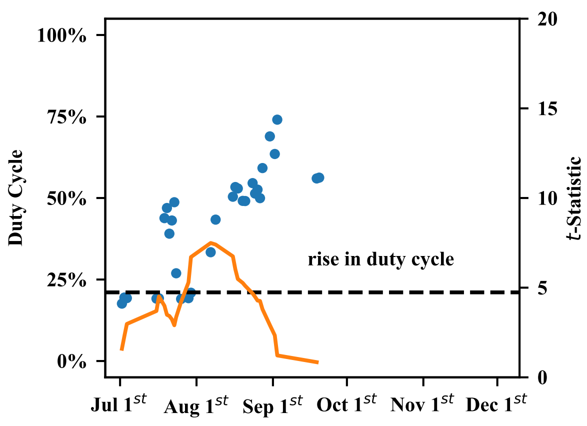
(a) $8°\text{F} < T_o - T_i < 12°\text{F}$



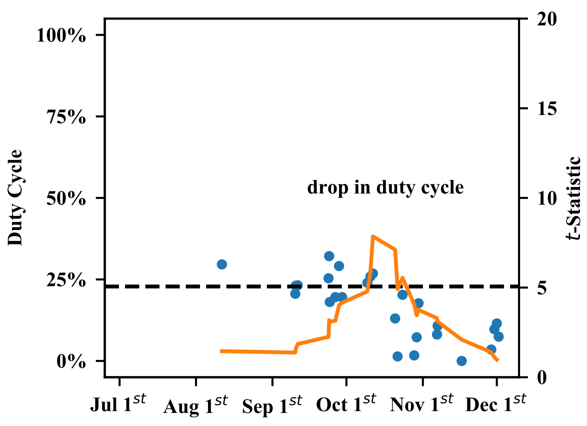
(b) $8°\text{F} < T_o - T_i < 12°\text{F}$



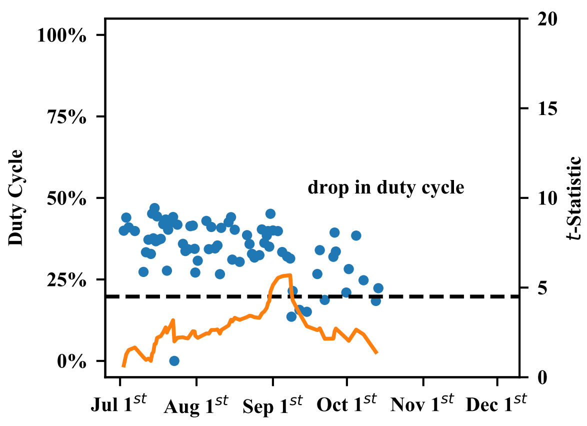
(c) $8°\text{F} < T_o - T_i < 12°\text{F}$



(d) $13°\text{F} < T_o - T_i < 17°\text{F}$

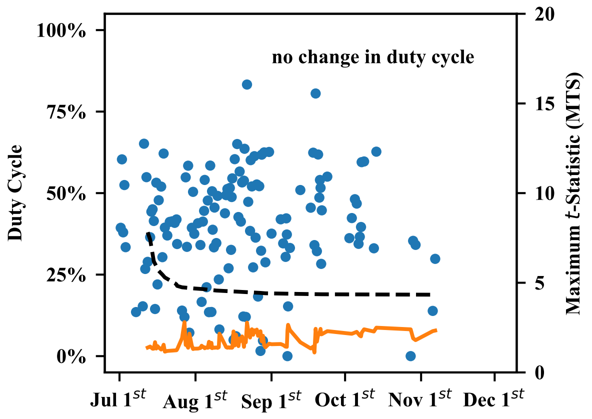


(e) $-4°\text{F} < T_o - T_i < 0°\text{F}$

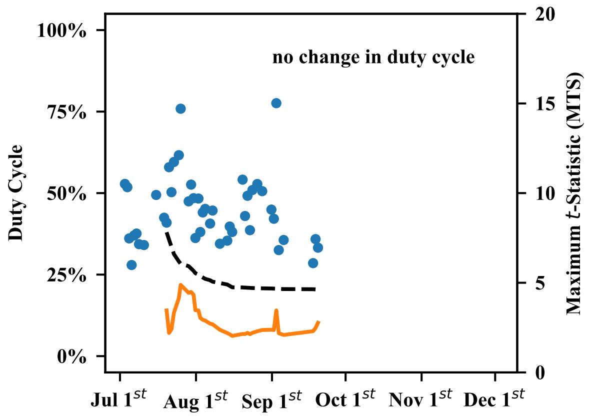


(f) $8°\text{F} < T_o - T_i < 12°\text{F}$

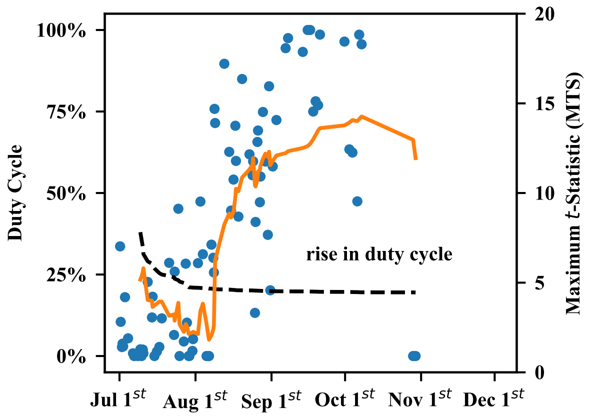
Figure 7.9: Examples of change point detection algorithm applied retrospectively to the cooling load from smart thermostat data at a specific outdoor temperature T_o condition. Each plot shows the cooling data (the average cooling load in a 6-hour period), the t-statistic (solid line), and the threshold (dashed line).



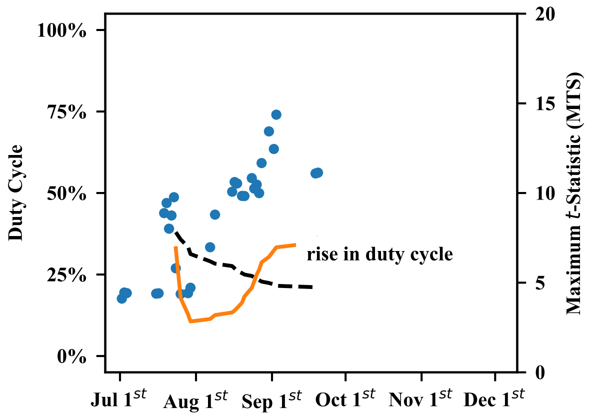
(a) $8°\text{F} < T_o - T_i < 12°\text{F}$



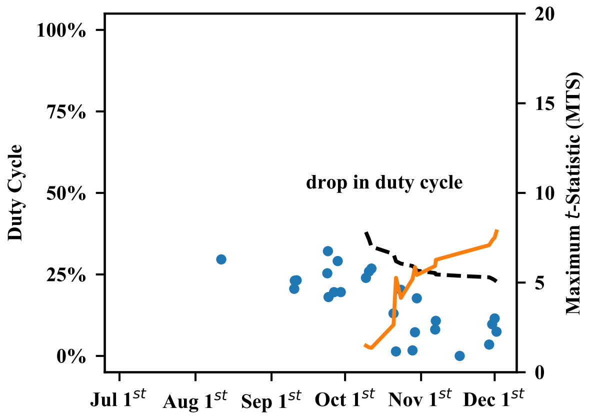
(b) $8°\text{F} < T_o - T_i < 12°\text{F}$



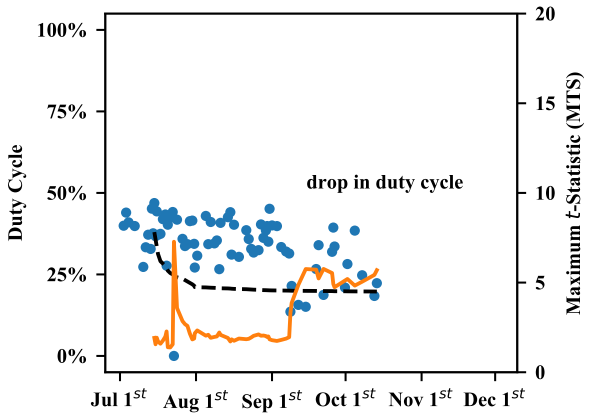
(c) $8°\text{F} < T_o - T_i < 12°\text{F}$



(d) $13°\text{F} < T_o - T_i < 17°\text{F}$



(e) $-4°\text{F} < T_o - T_i < 0°\text{F}$

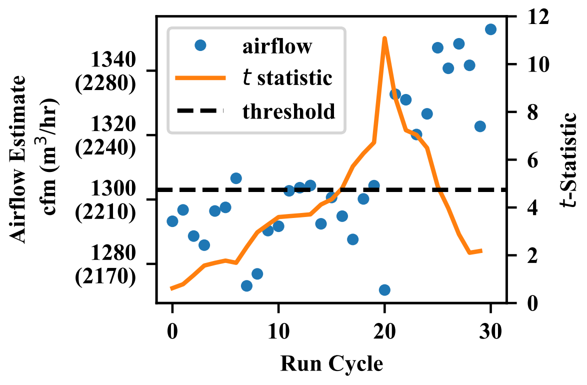


(f) $8°\text{F} < T_o - T_i < 12°\text{F}$

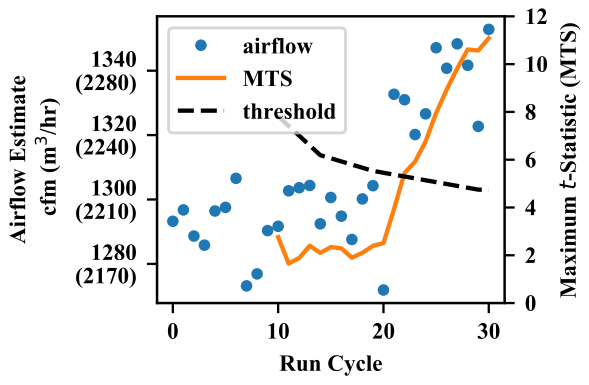
Figure 7.10: Examples of the change point detection algorithm applied recursively to the cooling effort from smart thermostat data at a specific outdoor temperature T_o condition. Each plot shows the cooling data (the average cooling load in a 6-hour period), the t-statistic (solid line), and the threshold (dashed line).

Coupling the change point detection with a cross system comparison would help in determining the cause of the observed changes. If a particular air conditioning system undergoes different change behavior than the other systems in a similar climate, then those observed changes are more likely due to an internal system change (such as those introduced by occupants or system faults) rather than an external change (such as those associated with weather). The proposed algorithm is simple and robust enough to be applied across many systems, and this cross system analysis will be a key to realizing the full potential of performing fault detection with smart thermostat data.

In addition to the data recorded by the smart thermostat, a relatively small portion of residential air conditioning systems are beginning to have diagnostic modules installed such that more detailed operating parameters are recorded. These detailed parameters include air and refrigerant temperatures throughout the vapor compression cycle. Using this data, important system parameters such as the airflow rate, cooling capacity, efficiency, and refrigerant massflow rate may be estimated [23, 40]. Change point detection may also be applied to this data set, and one case study is provided here. The system used for this case study is a field-installed system under ownership of the authors and therefore a change may be manually introduced to provide a case study in which the ground truth is known.



(a) Retrospective implementation.



(b) Recursive implementation.

Figure 7.11: An example of the change point detection algorithm applied both retrospectively and recursively to real air conditioning diagnostic data. The algorithm correctly detects the change in airflow after only 2 run cycles.

Using the air conditioning diagnostic data, the airflow (in cubic feet per minute or cfm) was estimated for each time that the system cycled on and then back off (a run cycle). Figure 7.11 shows the estimated airflow rate for 31 run cycles for a single system. After 21 run cycles, a change point is manually introduced. This provides a ground truth case study with which to test the algorithm whereas the ground truth in the case studies presented in Figures 7.9 and 7.10 is unknown. The maximum t-statistic algorithm correctly detects the change point, and the change point is detected after 2 run cycles.

7.7 Accounting for Multiple Change Points

The maximum t-statistic algorithm is capable of handling multiple change points in a time series with little modification from the formulation described in Section 7.4. After an initial change point is detected, the window should be measured from that change point onward; if the known change point is included in the window, then the algorithm will be unable to detect an additional change point. However, when the initial change point is detected with little delay, then its localization will be made with a small right hand sample (Y). The estimated location in time of the change point would improve as sizes of the left and right hand samples (n_X and n_Y) become balanced.

After a change has been detected, the window will be selected from the location of the detected change point. However, a secondary window that includes the change point may still be analyzed in order to improve the localization of the change point. This secondary window would be maintained until the left and right sample sizes are balanced or until another change is detected.

Figure 7.12 shows an example of this methodology for multiple change point detection. When the first change point is detected at time step 211, the primary window is then measured from the location of the detected change: time step 200. However, a secondary window containing the past 200 samples is still analyzed until time step 300 at which point the left and right sides of this secondary sample are balanced. While this secondary window is being maintained, the location of the change point may be updated because more information pertaining to that change point is becoming available. At time step 308, the change point at time step 300 is detected and a secondary window begins to be analyzed again. However, for this second change point, the secondary window

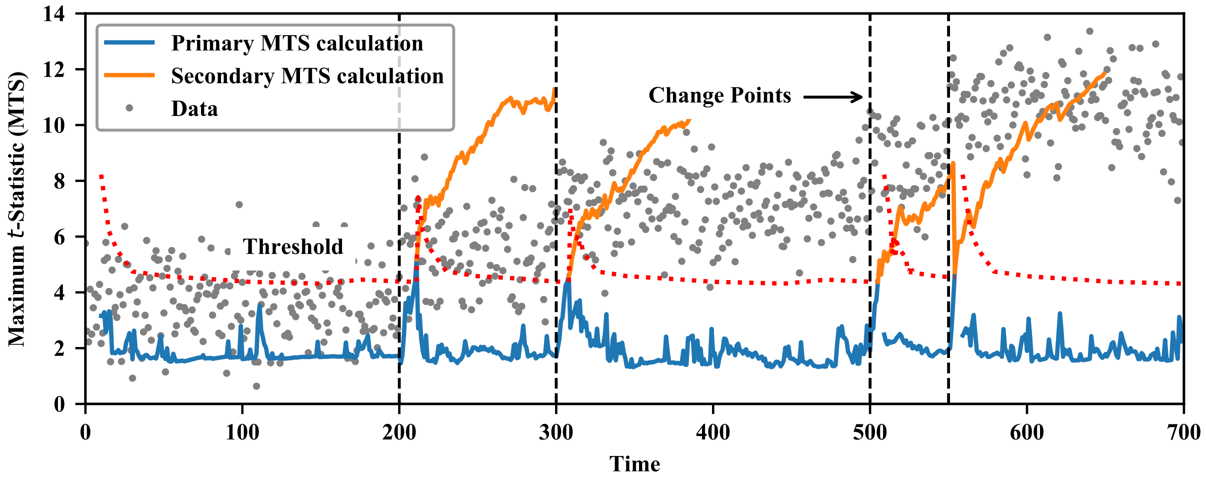


Figure 7.12: An example of multiple change point detection using a secondary window (measured from the next-to-last change point) in addition to the primary window (measured from the last change point). This example uses a maximum window size of $n = 200$, and each change point has a magnitude of 1.5σ.

is not the maximum size of $n = 200$; the maximum window size of 200 would contain both the first and second change points. The secondary window is therefore measured from the first change point (the next-to-last change point). The secondary window is maintained until time step 400 at which point $n_X = n_Y = 100$ for the secondary window.

The proposed methodology for applying the maximum t-statistic algorithm to multiple change points also provides a convenient means to account for Type I errors or false positives. The secondary MTS for all four detected change points in Figure 7.12 continues to grow as the secondary window progresses. However, if the detected change point were actually a false positive, then the secondary MTS calculation would quickly drop back below the threshold. Without this secondary window, the false positive would be logged as a change point without any way to further validate the decision.

7.8 Conclusions

This paper has presented a change point detection algorithm based on the t-statistic that (1) does not require *a priori* knowledge regarding the optimal operating point, (2) requires almost no tuning, and (3) is easy to implement. A methodology for accounting for multiple change points and false positives has also been presented. The maximum t-statistic algorithm has been applied to several sets of smart thermostat data and one set of air conditioning diagnostic data. These case studies highlight how this algorithm may be implemented effectively even on small datasets because the t-statistic inherently accounts for small sample sizes.

The simplicity of the proposed algorithm is particularly suited for Big Data and Internet-of-Things (IoT) applications such as smart thermostat data. The presented case studies highlighted how a cross system analysis that compares the changes detected in similar systems to each other will be an important element of air conditioning fault detection using thermostat data. A simple algorithm is necessary to simultaneously analyze data from thousands of systems with minimal tuning.

7.9 Acknowledgment

This material is based upon work supported by Trane, a subsidiary of Ingersoll Rand. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Trane.

8. CONCLUSIONS AND FUTURE WORK

The contributions outlined in Chapter 2 and presented in Chapters 3 through 7 center on the development of fault detection and diagnosis (FDD) methods for residential HVAC systems. These contributions include FDD methods that rely either on operating data from installed sensors or on thermostat data. However, valuable as they are, these contributions still do not constitute complete FDD methods. There are still several areas of development that must be explored further in order to obtain a complete FDD method for either operating data or thermostat data. Table 8.1 summarizes the most important areas that must be developed in order to commercialize these products and deploy them on a large scale.

8.1 Development of Methods that Rely on Installed Sensors

Chapter 4 proposed a set of virtual sensors that do not require sensors installed on the outdoor unit. The field implementation of these virtual sensors shows that the method is sufficiently sensitive to the changes in indoor airflow, cooling capacity, and system efficiency that result from faulty behavior. However, only indoor airflow and condenser airflow faults were introduced in this field implementation. Introducing other common faults such as a liquid line restriction and incorrect charge is necessary in order to quantify the different response that each of these faults generates. Understanding the different responses associated with various faults would enable some level of fault *diagnosis*. This proposed method would be similar to the rule-based methods that were originally proposed for air conditioning FDD [37, 25, 21]; however, the important difference is that the original rule-based methods used residuals of raw measurements whereas the proposed method would use residuals for more advanced, system-level parameters.

While the field implementation shows that the proposed virtual sensors are sufficiently sensitive to faults, the accuracy in the estimated values could not be validated because a ground-truth was unknown. Even without an accurate estimate, the precision or repeatability of the virtual sensors may still be leveraged to detect faults and diagnose them by using the relative magnitude of the

Table 8.1: Areas for Future Development for Fault Detection and Diagnosis (FDD) Methods for Air Conditioning Systems

Development of Methods that Rely on Installed Sensors	
(1)	Analysis of the fault *diagnosis* capabilities using system level parameters; i.e. distinguishing between condenser airflow, liquid line restriction, and refrigerant charge faults using refrigerant mass flow, cooling capacity, and system efficiency.
(2)	Analysis of the value of the proposed virtual sensors for *commissioning* rather than only for degradation detection. System configurations such as the type of heat source used and a draw- vs. blow-through blower configuration may affect the virtual sensor accuracy.
(3)	Adaptation of fault-free modeling methods for field-installed systems. While models developed in a laboratory provide an initial starting point, the models may need to be adjusted according to the installation.
(4)	Development of a further simplified FDD method using only a supply air temperature sensor. The return air conditions may be obtained from the thermostat, and then the airflow and cooling capacity virtual sensors may be used to detect faults and diagnose between airflow faults and all other faults.
Development of Fault Detection Methods using Thermostat Data	
(1)	Development of anomaly detection methods for thermostat data. Both statistical and more advanced methods may be used to identify which systems are anomalous.
(2)	Incorporation of additional data streams including system parameters (nominal cooling and heating capacity), home construction parameters (built date, square-footage, foundation type, location), and weather parameters (solar irradiation, precipitation, wind speed). The most important parameters should be identified and then used to more clearly highlight anomalous behavior.
(3)	Incorporation of the change point detection algorithm. Detected change points and their magnitudes should be incorporated in the thermostat data, and then anomalous change behavior may be identified.

observed changes. However, in order to use the proposed set of virtual sensors to commission a newly installed system (thereby significantly reducing the labor required for commissioning) estimated values must be accurate. The location at which the physical sensors are installed may need to be standardized, and the virtual sensors may need to be calibrated to provide the most accurate estimates.

Three of the four proposed virtual sensors (cooling capacity, system efficiency, and refrigerant mass flow) will be highly dependent on the driving conditions (outdoor air temperature and indoor air temperature and humidity). Fault free models are therefore essential to compare the observed values with the desired or fault free values. The fault-free models must therefore account for variations from one installation to the next. A method for training models for field-installed system was proposed in [68], and similar methods should be explored to find the most appropriate and practical approach.

The FDD method in Chapter 4 can be further simplified by (a) obtaining the return air conditions from the thermostat measurements and (b) estimating only airflow and cooling capacity in order to detect faults and diagnose between airflow fault and other faults. With these simplifications, only a supply air temperature sensor would be needed, but fault diagnosis capabilities would be limited to distinguishing between airflow faults and other faults. The effectiveness of this simplified method will depend, in part, on the location of the thermostat and how well the thermostat conditions reflect the return air conditions.

8.2 Development of Fault Detection Methods using Thermostat Data

Chapter 6 presents a preprocessing method for thermostat data that more complete fault detection methods may incorporate. To continue the development of fault detection methods using thermostat data, appropriate anomaly detection methods must be identified and adapted for this application. Preprocessing the data is key to highlighting the important relationships in the data, but anomaly detection methods must still be applied or developed. In order to further highlight the important relationships, additional data sources may be incorporated. On its own, thermostat data is relatively sparse and many important parameters are missing; the nominal cooling (or heating) capacity, occupancy behavior, the home size and other construction details, and additional weather information are all potential variables. Identifying which parameters are most important, optimizing the cost of acquiring the data with the benefit that the data provides, and appropriate incorporating the available data are all areas of development to improve thermostat-based fault detection methods.

Yet another potential data stream to incorporate with thermostat data is the results from applying the change-point detection method proposed in Chapter 7. Many changes may be detected in thermostat data, but few of the detected changes will actually be due to the onset of a system fault. Many observed changes will be due to seasonal variations and occupant behavior. Including the detected changes with the rest of the thermostat data will not only help to identify the changes correlated with occupant behavior and changes correlated with weather. but will also enable the detection of anomalous change point behavior as data from many systems are compared.

8.3 Conclusions & Discussion

Fault detection and diagnosis (FDD) for residential air conditioning systems is a field which promises significant energy savings and improved comfort. The recent developments presented in this dissertation focus on (a) reducing and simplifying the required sensors to perform FDD, and (b) leveraging thermostat data for fault detection. The methods presented are novel and promising, but still require significant work in order to produce commercialized FDD products. This future development will focus on analyzing the capabilities of the proposed virtual sensors, applying of fault-free models, and detecting anomalies with thermostat data.

REFERENCES

[1] EIA, "April 2018 Monthly Energy Review," tech. rep., U.S. Energy Information Administration, 2018.

[2] U.S. Energy Information Administration, "Commercial buildings energy consumption survey (cbecs)," 2012. `https://www.eia.gov/consumption/commercial/`, Accessed: 2017-02-13.

[3] EIA, "Residential Energy Consumption Survey (RECS)," tech. rep., U.S. Energy Information Administration, 2015.

[4] S. Katipamula and M. R. Brambley, "Methods for Fault Detection, Diagnostics, and Prognostics for Building Systems - A Review, Part I," *HVAC&R Research*, vol. 11, no. 1, pp. 3–25, 2005.

[5] S. Katipamula and M. R. Brambley, "Methods for Fault Detection, Diagnostics, and Prognostics for Building Systems - A Review, Part II," *HVAC&R Research*, vol. 11, no. 2, pp. 169–187, 2005.

[6] W. Kim and S. Katipamula, "A review of fault detection and diagnostics methods for building systems," *Science and Technology for the Built Environment*, vol. 24, no. 1, pp. 3–21, 2018.

[7] J. E. Braun and H. Li, "Automated Fault Detection and Diagnostics of Rooftop Air Conditioners for California," tech. rep., Final report submitted to Architectural Energy Corporation for the Building Energy Efficiency Program Sponsored by California Energy Commission, Purdue University, 2003.

[8] H. Li and J. E. Braun, "An Economic Evaluation of Automated Fault Detection and Diagnosis for Rooftop Air Conditioners," *International Refrigeration and Air Conditioning Co.*, 2004.

[9] H. Li and J. E. Braun, “Economic Evaluation of Benefits Associated with Automated Fault Detection and Diagnosis in Rooftop Air Conditioners ,” *ASHRAE Transactions*, vol. 113, no. 2, pp. 200–210, 2007.

[10] C. Neme, S. Nadel, and J. Proctor, “Energy Savings Potential From Addressing Residential Air Conditioner and Heat Pump Installation Problems,” *American Council for an Energy Efficient Economy, Report A992, Washington, DC, USA,*, pp. 1–34, 1999.

[11] J. Proctor and T. Downey, “Transforming Routine Air Conditioner Maintenance Practices to Improve Equipment Efficiency and Performance,” *In Proceedings of the 1999 International Energy Program Evaluation Conference*, pp. 1–12, 1999.

[12] T. Downey and J. Proctor, “What Can 13,000 Air Conditioners Tell Us?,” *In the Proceedings of the 2002 ACEEE Summer Study on Energy Efficiency in Buildings*, vol. 1, pp. 53–67, 2002.

[13] M. Kim, W. V. Payne, and P. A. Domanski, “Performance of a residential heat pump operating in the cooling mode with single faults imposed,” tech. rep., National Institute of Standards and Technology, 2006.

[14] W. Kim and J. E. Braun, “Impacts of Refrigerant Charge on Air Conditioner and Heat Pump Performance,” *International Refrigeration and Air Conditioning Conference. Paper 1122*, 2010.

[15] S. H. Yoon, W. V. Payne, and P. A. Domanski, “Residential heat pump heating performance with single faults imposed,” *Applied Thermal Engineering*, vol. 31, no. 5, pp. 765 – 771, 2011.

[16] W. Kim and J. E. Braun, “Evaluation of the impacts of refrigerant charge on air conditioner and heat pump performance,” *International Journal of Refrigeration*, vol. 35, no. 7, pp. 1805 – 1814, 2012.

[17] Z. T. Taylor, V. V. Mendon, and N. Fernandez, "Methodology for Evaluating Cost-Effectiveness of Residential Energy Code Changes," tech. rep., Pacific Northwest National Laboratory, Report PNNL-21294, 2015.

[18] J. Granderson, R. Singla, E. Mayhorn, P. Ehrlich, D. Vrabie, and S. Frank, "Characterization and Survey of Automated Fault Detection and Diagnostics Tools," *Lawrence Berkely National Laboratory, Report LBNL-2001075, Berkeley, CA*, 2017.

[19] T. M. Rossi, *Detection, diagnosis, and evaluation of faults in vapor compression equipment*. PhD thesis, Purdue University, 1995.

[20] M. S. Breuker and J. E. Braun, "Evaluating the Performance of a Fault Detection and Diagnostic System for Vapor Compression Equipment," *HVAC&R Research*, vol. 4, no. 4, pp. 401–425, 1998.

[21] H. Li and J. E. Braun, "An Improved Method for Fault Detection and Diagnosis Applied to Packaged Air Conditioners," *ASHRAE Transactions*, vol. 109, pp. 683–692, 2003.

[22] H. Li and J. E. Braun, "A Methodology for Diagnosing Multiple Simultaneous Faults in Vapor-Compression Air Conditioners," *HVAC&R Research*, vol. 13, no. 2, pp. 369–395, 2007.

[23] H. Li and J. E. Braun, "Decoupling features and virtual sensors for diagnosis of faults in vapor compression air conditioners," *International Journal of Refrigeration*, vol. 30, no. 3, pp. 546 – 564, 2007.

[24] S. Katipamula, W. Kim, R. G. Lutes, and R. M. Underhill, "Rooftop Unit Embedded Diagnostics: Automated Fault Detection and Diagnostics (AFDD) Development, Field Testing and Validation," tech. rep., Pacific Northwest National Laboratory, Report PNNL-23790, Richland, WA, 2015.

[25] B. Chen and J. E. Braun, "Simple Rule-based Methods for Fault Detection and Diagnostics Applied to Packaged Air Conditioners," *ASHRAE Transactions*, vol. 107, no. 1, pp. 847–857, 2001.

[26] P. R. Armstrong, C. R. Laughman, S. B. Leeb, and L. K. Norford, “Fault Detection Based on Motor Start Transients and Shaft Harmonics Measured at the RTU Electrical Service,” *International Refrigeration and Air Conditioning Conference. Paper 664*, 2004.

[27] P. R. Armstrong, C. R. Laughman, S. B. Leeb, and L. K. Norford, “Detection of Rooftop Cooling Unit Faults Based on Electrical Measurements,” *HVAC&R Research*, vol. 12, no. 1, pp. 151–175, 2006.

[28] S. M. Namburu, J. Luo, M. Azam, K. Choi, and K. R. Pattipati, “Fault detection, diagnosis, and data-driven modeling in HVAC chillers,” *In Proceedings of SPIE 5809, Signal Processing, Sensor Fusion, and Target Recognition XIV*, vol. 5809, pp. 143–154, 2005.

[29] K. Choi, S. M. Namburu, M. S. Azam, J. Luo, K. R. Pattipati, and A. Patterson-Hine, “Fault Diagnosis in HVAC Chillers,” *IEEE Instrumentation Measurement Magazine*, vol. 8, pp. 24–32, Aug 2005.

[30] D. Zogg, E. Shafai, and H. Geering, “Fault diagnosis for heat pumps with parameter identification and clustering,” *Control Engineering Practice*, vol. 14, no. 12, pp. 1435–1444, 2006.

[31] B. Huchuk, W. O’Brien, and S. Sanner, “A longitudinal study of thermostat behaviors based on climate, seasonal, and energy price considerations using connected thermostat data,” *Building and Environment*, vol. 139, pp. 199–210, 2018.

[32] M. F. Touchie and J. A. Siegel, “Residential HVAC runtime from smart thermostats: characterization, comparison, and impacts,” *Indoor Air*, vol. 28, no. 6, pp. 905–915, 2018.

[33] A. Rogers, F. Guo, and B. Rasmussen, “A Review Of Fault Detection And Diagnosis Methods For Residential Air Conditioning Systems,” ***In Review***.

[34] F. Guo, A. P. Rogers, and B. P. Rasmussen, “Multivariate Fault Detection for Residential HVAC Systems using Cloud-based Thermostat Data, Part 1: Methodology,” ***In Review***.

[35] F. Guo, A. P. Rogers, and B. P. Rasmussen, “Multivariate Fault Detection for Residential HVAC Systems using Cloud-based Thermostat Data, Part 2: Case Studies,” ***In review***.

[36] A. Rogers, F. Guo, J. Martinez, and B. Rasmussen, "Labeling Modes of Operation and Extracting Features for Fault Detection with Cloud-Based Thermostat Data," ***In Review***.

[37] T. M. Rossi and J. E. Braun, "A Statistical, Rule-Based Fault Detection and Diagnostic Method for Vapor Compression Air Conditioners," *HVAC&R Research*, vol. 3, no. 1, pp. 19–37, 1997.

[38] M. Kim, S. H. Yoon, W. V. Payne, and P. A. Domanski, "Cooling Mode Fault Detection and Diagnosis Method for a Residential Heat Pump," *NIST Special Publication 1087*, 2008.

[39] A. Rogers, F. Guo, and B. Rasmussen, "Uncertainty Analysis And Field Implementation Of A Fault Detection And Diagnosis Method For Residential HVAC Systems," ***In Review***.

[40] D. Yu, H. Li, and M. Yang, "A virtual supply airflow rate meter for rooftop air-conditioning units," *Building and Environment*, vol. 46, no. 6, pp. 1292–1302, 2011.

[41] A. Rogers, F. Guo, and B. Rasmussen, "Applying Static Fault Detection and Diagnosis Methods to Transient Air Conditioning Data Using an Equilibrium Prediction," ***In Review***.

[42] J. Jacquelin, *Regressions et Equations Integrales*. Scribd, 2009. `https://www.scribd.com/doc/14674814/Regressions-et-equations-integrales`.

[43] A. Rogers, F. Guo, and B. Rasmussen, "A Change Point Detection Algorithm With Application To Smart Thermostat Data," ***In Review***.

[44] D. Yu, H. Li, and Y. Yu, "A gray-box based virtual SCFM meter in rooftop air-conditioning units," *Journal of Thermal Science and Engineering Applications*, vol. 3, no. 1, p. 011005, 2011.

[45] M. S. Breuker and J. E. Braun, "Common Faults and Their Impacts for Rooftop Air Conditioners," *HVAC&R Research*, vol. 4, no. 3, pp. 303–318, 1998.

[46] A. P. Rogers and B. P. Rasmussen, "Opportunities for consumer-driven load shifting in commercial and industrial buildings," *Sustainable Energy, Grids and Networks*, vol. 16, pp. 243–258, 2018.

[47] EIA, "Electric Power Monthly with Data for September 2018," tech. rep., U.S. Energy Information Administration, 2018.

[48] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri, "A review of process fault detection and diagnosis Part II: Qualitative models and search strategies," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 313–326, 2003.

[49] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis Part I: Quantitative model-based methods," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[50] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis Part III: Process History based methods," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 327–346, 2003.

[51] J. E. Braun, "Automated Fault Detection and Diagnostics for Vapor Compression Cooling Equipment," *Transactions of the ASME Journal of Solar Energy Engineering*, vol. 125, no. 3, pp. 266–274, 2003.

[52] W. V. Payne, P. A. Domanski, and S. H. Yoon, "Heating Mode Performance Measurements For A Residential Heat Pump With Single-Faults Imposed," *NIST Technical Note 1648*, pp. 1–162, 2009.

[53] P. A. Domanski, H. I. Henderson, and W. V. Payne, "Sensitivity Analysis of Installation Faults on Heat Pump Performance," *NIST Technical Note 1848*, pp. 1–94, 2014.

[54] P. A. Domanski, H. Henderson, and W. V. Payne, "Effect of heat pump commissioning faults on energy use in a slab-on-grade residential house," *Applied Thermal Engineering*, vol. 90, pp. 352 – 361, 2015.

[55] M. Kim, S. H. Yoon, P. A. Domanski, and W. V. Payne, "Design of a steady-state detector for fault detection and diagnosis of a residential air conditioner," *International Journal of Refrigeration*, vol. 31, no. 5, pp. 790–799, 2008.

[56] H. Li and J. E. Braun, “Development, Evaluation, and Demonstration of a Virtual Refrigerant Charge Sensor,” *HVAC&R Research*, vol. 15, no. 1, pp. 117–136, 2009.

[57] S. M. Namburu, M. S. Azam, J. Luo, K. Choi, and K. R. Pattipati, “Data-Driven Modeling, Fault Diagnosis and Optimal Sensor Selection for HVAC Chillers,” *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 3, pp. 469–473, 2007.

[58] M. R. Brambley, “A Novel, Low-Cost, Reduced-Sensor Approach for Providing Smart Remote Monitoring and Diagnostics for Packaged Air Conditioners and Heat Pumps,” *Pacific Northwest National Laboratory, Report PNNL-18891, Richland, WA*, pp. 1–11, 2009.

[59] G. W. Hart, “Residential Energy Monitoring and Computerized Surveillance via Utility Power Flows,” *IEEE Technology and Society Magazine*, vol. 8, pp. 12–16, June 1989.

[60] G. W. Hart, “Nonintrusive Appliance Load Monitoring,” *Proceedings of the IEEE*, vol. 80, pp. 1870–1891, Dec 1992.

[61] L. K. Norford and S. B. Leeb, “Non-intrusive electrical load monitoring in commercial buildings based on steady-state and transient load-detection algorithms,” *Energy and Buildings*, vol. 24, no. 1, pp. 51 – 64, 1996.

[62] L. K. Norford, J. A. Wright, R. A. Buswell, D. Luo, C. J. Klaassen, and A. Suby, “Demonstration of Fault Detection and Diagnosis Methods for Air-Handling Units,” *HVAC&R Research*, vol. 8, no. 1, pp. 41–71, 2002.

[63] D. Luo, L. K. Norford, S. R. Shaw, and S. B. Leeb, “Monitoring HVAC Equipment Electrical Loads from a Centralized Location–Methods and Field Test Results,” *ASHRAE Transactions*, vol. 108, pp. 841–857, 2002.

[64] A. S. Glass, P. Gruber, M. Roos, and J. Todtli, “Qualitative Model-Based Fault Detection in Air-Handling Units,” *IEEE Control Systems*, vol. 15, pp. 11–22, Aug 1995.

[65] H. Li, *A Decoupling-Based Unified Fault Detection and Diagnosis Approach for Packaged Air Conditioners*. PhD thesis, Purdue University, 2004.

[66] H. Li and J. E. Braun, “On-Line Models for Use in Automated Fault Detection and Diagnosis for HVAC&R Equipment,” *Proceedings of the 2002 ACEEE Conference on Energy Efficiency in Buildings, Monterey, CA*, vol. 7, pp. 147–158, 2002.

[67] M. Kim, S. H. Yoon, W. V. Payne, and P. A. Domanski, “Development of the reference model for a residential heat pump system for cooling mode fault detection and diagnosis,” *Journal of Mechanical Science and Technology*, vol. 24, pp. 1481–1489, Jul 2010.

[68] J. Heo, W. V. Payne, P. A. Domanski, and Z. Du, “Self Training of a Fault-Free Model for Residential Air Conditioner Fault Detection and Diagnostics,” *NIST Technical Note 1881*, 2015.

[69] H. T. Grimmelius, J. K. Woud, and G. Been, “On-line failure diagnosis for compression refrigeration plants,” *International Journal of Refrigeration*, vol. 18, no. 1, pp. 31 – 41, 1995.

[70] M. C. Comstock and J. E. Braun, “Experimental Data from Fault Detection and Diagnostic studies on a Centrifugal Chiller,” tech. rep., Purdue University, Ray W. Herrick Laboratories, Report HL99-18, 1999.

[71] M. C. Keir and A. G. Alleyne, “Dynamic Modeling, Control, and Fault Detection in Vapor Compression Systems,” Master’s thesis, University of Illinois at Urbana-Champaign., 2006.

[72] D. P. Yuill and J. E. Braun, “Evaluating fault detection and diagnostics protocols applied to air-cooled vapor compression air-conditioners,” *International Refrigeration and Air Conditioning Conference. Paper 1307*, 2012.

[73] D. P. Yuill and J. E. Braun, “A figure of merit for overall performance and value of AFDD tools,” *International Journal of Refrigeration*, vol. 74, pp. 651–661, 2017.

[74] R. Radhakrishnan, D. Nikovski, K. Peker, and A. Divakaran, “A Comparison between Polynomial and Locally Weighted Regression for Fault Detection and Diagnosis of HVAC Equipment,” *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*, pp. 3668–3673, 2006.

[75] P. Parikh and B. P. Rasmussen, “Development of a Fault Detection and Diagnostic Tool for Use in Walk-Through Energy Audits,” *ASHRAE Transactions*, vol. 121, pp. 281–293, 2015.

[76] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse, “The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes,” in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pp. 211–224, 2010.

[77] W. van der Ham, M. Klein, S. A. Tabatabaei, D. J. Thilakarathne, and J. Treur, “Methods for a Smart Thermostat to Estimate the Characteristics of a House Based on Sensor Data,” *Energy Procedia*, vol. 95, pp. 467–474, 2016.

[78] H. B. Gunay, W. O’Brien, I. Beausoleil-Morrison, and J. Bursill, “Development and implementation of a thermostat learning algorithm,” *Science and Technology for the Built Environment*, vol. 24, no. 1, pp. 43–56, 2018.

[79] A. Rogers, S. Ghosh, R. Wilcock, and N. R. Jennings, “A Scalable Low-Cost Solution to Provide Personalised Home Heating Advice to Households,” *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, 2013.

[80] W. J. N. Turner, A. Staino, and B. Basu, “Residential HVAC fault detection using a system identification approach,” *Energy and Buildings*, vol. 151, pp. 1–17, 2017.

[81] S. Wang and F. Xiao, “AHU sensor fault diagnosis using principal component analysis method,” *Energy and Buildings*, vol. 36, no. 2, pp. 147 – 160, 2004.

[82] S. Wang and J. Qin, “Sensor fault detection and validation of VAV terminals in air conditioning systems,” *Energy Conversion and Management*, vol. 46, no. 15, pp. 2482 – 2500, 2005.

[83] F. Xiao, S. Wang, and J. Zhang, “A diagnostic tool for online sensor health monitoring in air-conditioning systems,” *Automation in Construction*, vol. 15, no. 4, pp. 489 – 503, 2006.

[84] X. Hao, G. Zhang, and Y. Chen, “Fault-tolerant control and data recovery in HVAC monitoring system,” *Energy and Buildings*, vol. 37, no. 2, pp. 175 – 180, 2005.

[85] S. Li and J. Wen, "A model-based fault detection and diagnostic methodology based on PCA method and wavelet transform," *Energy and Buildings*, vol. 68, pp. 63 – 71, 2014.

[86] Z. Du, X. Jin, and L. Wu, "Fault detection and diagnosis based on improved PCA with JAA method in VAV systems," *Building and Environment*, vol. 42, no. 9, pp. 3221 – 3232, 2007.

[87] X. Xu, F. Xiao, and S. Wang, "Enhanced chiller sensor fault detection, diagnosis and estimation using wavelet analysis and principal component analysis methods," *Applied Thermal Engineering*, vol. 28, no. 2, pp. 226 – 237, 2008.

[88] Z. Hou, Z. Lian, Y. Yao, and X. Yuan, "Data mining based sensor fault diagnosis and validation for building air conditioning system," *Energy Conversion and Management*, vol. 47, no. 15, pp. 2479 – 2490, 2006.

[89] Y. Zhu, X. Jin, and Z. Du, "Fault diagnosis for sensors in air handling unit based on neural network pre-processed by wavelet and fractal," *Energy and Buildings*, vol. 44, pp. 7 – 16, 2012.

[90] S. Katipamula, M. R. Brambley, and L. Luskay, "Automated Proactive Techniques for Commissioning Air-Handling Units," *Journal of solar energy engineering*, vol. 125, no. 3, pp. 282–291, 2003.

[91] S. Katipamula and M. Brambley, "Automated Proactive Fault Isolation: A Key to Automated Commissioning," *ASHRAE Transactions*, vol. 113, no. 2, pp. 40–51, 2007.

[92] M. Najafi, D. M. Auslander, P. L. Bartlett, and P. Haves, "Fault Diagnostics and Supervised Testing: How Fault Diagnostic tools can be Proactive?," *Eleventh International Conference on Intelligent Systems and Controls*, 11/2008 2008.

[93] EIA, "Annual Energy Outlook 2018," tech. rep., U.S. Energy Information Administration, 2018.

[94] AHRI, "AHRI Standard 540: 2015 Standard for Performance Rating of Positive Displacement Refrigerant Compressors and Compressor Units," tech. rep., Air-Conditioning, Heating, & Refrigeration Institute, 2015.

[95] H. Li and J. E. Braun, "Virtual Refrigerant Pressure Sensors for Use in Monitoring and Fault Diagnosis of Vapor-Compression Equipment," *HVAC&R Research*, vol. 15, no. 3, pp. 597–616, 2009.

[96] N. Fernandez, S. Katipamula, W. Wang, Y. Xie, M. Zhao, and C. Corbin, "Impacts of Commercial Building Controls on Energy Savings and Peak Load Reduction," tech. rep., Pacific Northwest National Laboratory, 2017.

[97] W. Wang, S. Katipamula, H. Ngo, R. Underhill, D. Taasevigen, and R. Lutes, "Advanced Rooftop Control (ARC) Retrofit: Field-Test Results," tech. rep., Pacific Northwest National Laboratory, 2013.

[98] ASHRAE, *2017 ASHRAE Handbook: Fundamentals*. 2017.

[99] Y. Kashirajima, T. Sugiura, M. Takahashi, Y. Sugihara, and N. Ooshima, "Experimental study of indoor thermal environment for cold air distribution systems using various air outlets," *ASHRAE Transactions*, vol. 108, pp. 16–22, 2002.

[100] F. L. Painter, "Condensate Harvesting from Large Dedicated Outside Air-Handling Units with Heat Recovery," *ASHRAE Transactions*, vol. 115, pp. 573–580, 2009.

[101] A. Kusiak and M. Li, "Cooling output optimization of an air handling unit," *Applied Energy*, vol. 87, no. 3, pp. 901–909, 2010.

[102] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Pearson Higher Education, 2010.

[103] U.S. Energy Information Administration, "December 2017 monthly energy review," 2017.

[104] M. Hewett, D. Bohac, R. Landry, T. Dunsworth, S. Englander, and G. Peterson, "Measured Energy and Demand Impacts of Efficiency Tune-ups for Small Commercial Cooling Systems," tech. rep., Center for Energy and Environment, 1992.

[105] O. T. Ogunsola, L. Song, and G. Wang, "Development and validation of a time-series model for real-time thermal load estimation," *Energy and Buildings*, vol. 76, pp. 440–449, 2014.

[106] O. T. Ogunsola and L. Song, “Application of a simplified thermal network model for real-time thermal load estimation,” *Energy and Buildings*, vol. 96, pp. 309–318, 2015.

[107] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, “Image change detection algorithms: a systematic survey,” *IEEE Transactions on Image Processing*, vol. 14, pp. 294–307, March 2005.

[108] D. Lu, P. Mausel, E. Brondizio, and E. Moran, “Change detection techniques,” *International Journal of Remote Sensing*, vol. 25, no. 12, pp. 2365–2401, 2004.

[109] Y. Kawahara, T. Yairi, and K. Machida, “Change-Point Detection in Time-Series Data Based on Subspace Identification,” *Proceedings of the Seventh IEEE International Conference on Data Mining*, pp. 559–564, 2007.

[110] N. Itoh and J. Kurths, “Change-Point Detection of Climate Time Series by Nonparametric Method,” in *Proceedings of the World Congress on Engineering and Computer Science*, 2010.

[111] M. Basseville and I. Nikiforov, *Detection of abrupt changes: theory and application*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc, 1993.

[112] T. IdÃľ and K. Tsuda, “Change-Point Detection using Krylov Subspace Learning,” *Proceedings of the 2007 SIAM International Conference on Data Mining*, pp. 515–520, 2007.

[113] V. Moskvina and A. Zhigljavsky, “An Algorithm Based on Singular Spectrum Analysis for Change-Point Detection,” *Communications in Statistics - Simulation and Computation*, vol. 32, no. 2, pp. 319–352, 2003.

[114] Y. Kawahara and M. Sugiyama, “Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation,” *Proceedings of the 2009 SIAM International Conference on Data Mining, Sparks, Nevada, USA*, pp. 389–400, 2009.

[115] M. Sugiyama, T. Suzuki, and T. Kanamori, “Density-ratio matching under the Bregman divergence: a unified framework of density-ratio estimation,” *Annals of the Institute of Statistical Mathematics*, vol. 64, no. 5, pp. 1009–1044, 2012.

[116] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, pp. 72–83, 2013.

[117] W. Navidi, *Statistics for Engineerings and Scientists*. McGraw-Hill, 2006.

APPENDIX A

FIELD IMPLEMENTATION - INSTALLATION DETAILS

Chapters 4 and 5 propose fault detection and diagnosis methods for air conditioning systems and then validate these methods experimentally. Chapter 4 proposes a set of virtual sensors which uses indoor unit sensors to estimate system parameters airflow, cooling capacity, system efficiency, and refrigerant mass flow. Chapter 5 then proposes a method to predict the equilibrium point of a system while the system is still operating in the start-up transients. The experimental validation of these methods involved a field-installed air conditioning system in which faults could be introduced and in which the response could be measured.

A 3.5 ton, 14 SEER packaged heat pump unit (model no. Goodman GDH1442H41) was used for the experimental validation. The unit is installed at a manufactured home site. Figure A.1 shows a portion of the home floor-plan (not to scale) with several HVAC components included, and Figure A.2 shows several photographs of the HVAC system. The packaged unit rests on a plaster-coated foam pad outside the home. Insulated, flexible ductwork runs under the home to connect the unit to the return air filter (located in the laundry room of the home) and to a system of supply ductwork and registers. The air-handling portion of the heat pump unit has a draw-through configuration in which the blower motor pulls air through the indoor coil.

Figure A.3 shows the top-view of the air handling unit portion of the heat pump system after the top of the enclosure has been removed. The air from the home is pulled through the return air duct and the indoor coil by the blower motor and then pushed through the supply air duct. The packaged unit used for the experimental validation in this dissertation is not a typical configuration for U.S. residential buildings. Residential air conditioners are more typically ‘split’ systems in which the compressor, condenser, and condenser fan are located in a unit outside; the expansion device, the evaporator, and the blower motor are located in an indoor unit; and refrigerant lines and power cables are run between the two units.

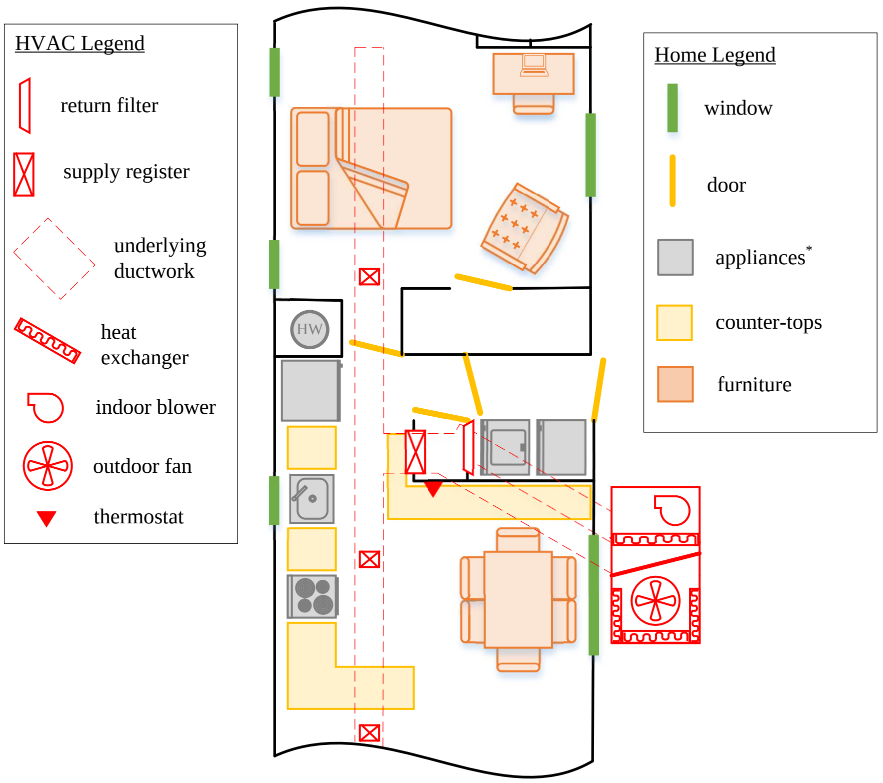


Figure A.1: A partial floor-plan of the 1190 sq. ft. manufactured home (1995 Clayton Homes Texan model) used for experimental work with the Nexia Diagnostic Module (NDM). An NDM was installed on the 3.5 ton, 14 SEER, packaged heat pump (Goodman GDH1442H41) unit at this home. In the diagram, various HVAC components are shown in red.

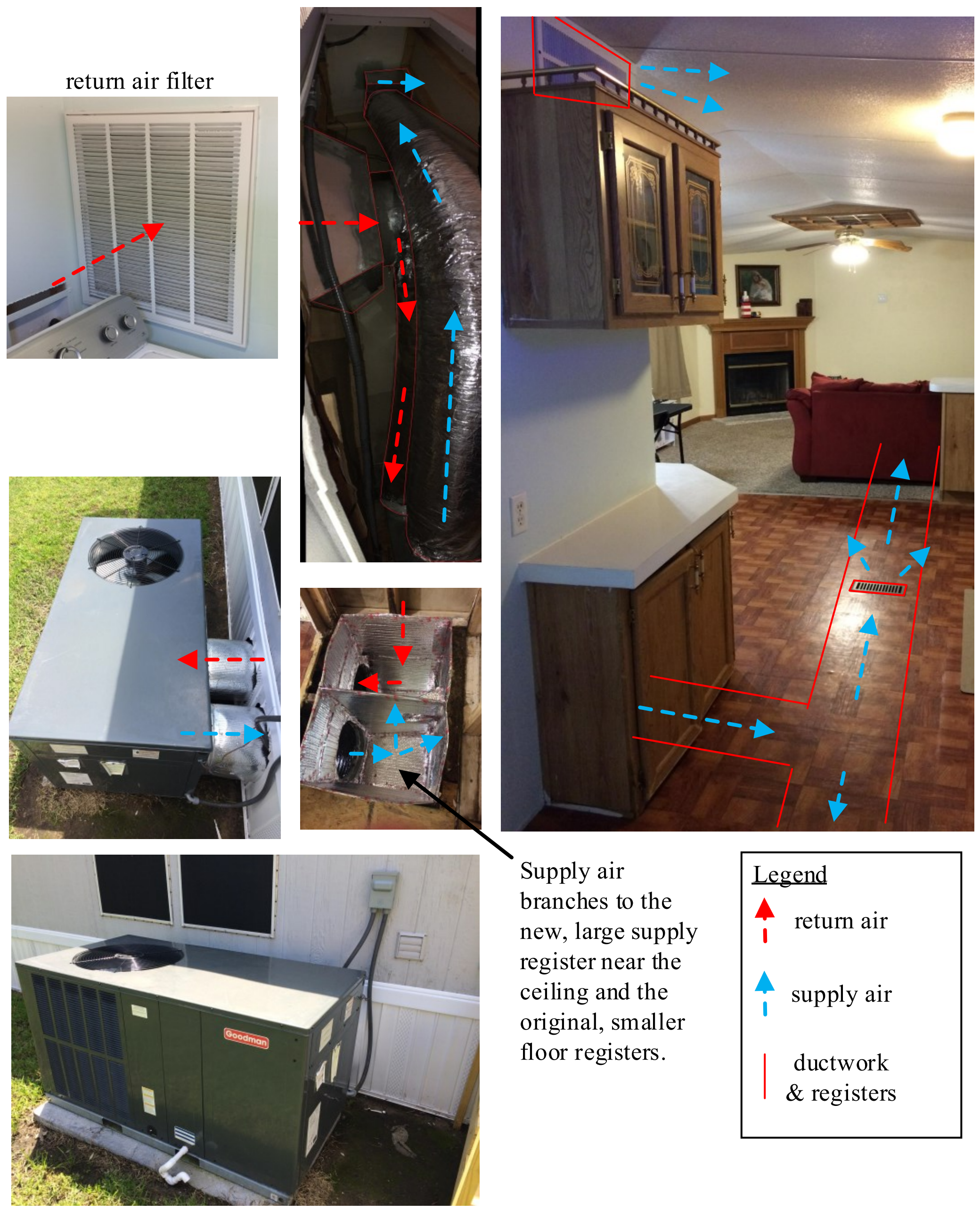

return air filter
Supply air branches to the new, large supply register near the ceiling and the original, smaller floor registers.
Legend
return air
supply air
ductwork & registers

Figure A.2: Several photographs outlining the installation of the HVAC system with the packaged unit, ductwork, return filter, and supply registers.

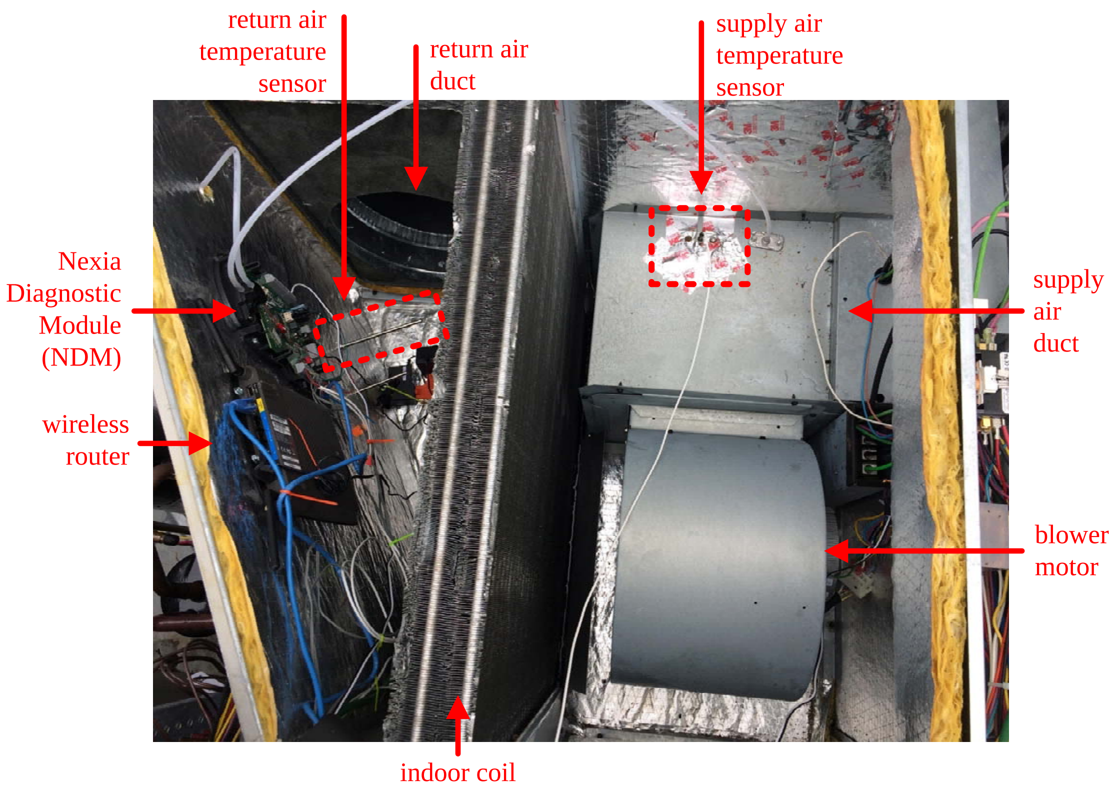

return air temperature sensor
return air duct
supply air temperature sensor
Nexia Diagnostic Module (NDM)
wireless router
supply air duct
blower motor
indoor coil

Figure A.3: A top-down view of the air-handling portion of the packaged heat pump. The indoor coil separates the return air (left) from the supply air (right) and the draw-through blower motor pulls air through the coil and into the supply duct. The Nexia Diagnostic Module (NDM) is installed with a wireless router in the return plenum.

As discussed in Chapter 4, a split system presents a challenge for data acquisition because the two units are located considerably far apart. Chapter 4 then proposes a set of virtual sensors which rely on only indoor-unit sensors. Therefore, only the air-handling portion of the experimental packaged unit was instrumented. Data acquisition was performed using a prototype data acquisition system developed by Trane: the Nexia Diagnostic Module (NDM). The modules enable Trane dealers in the Unites States to instrument systems which would not otherwise have detailed operating data available. This system is currently under development and few NDM systems have been installed to-date. Figure A.3 shows where the NDM is installed in the return air plenum and where the return air and supply air temperature sensors are installed on the unit.

The Nexia Diagnostic Module (NDM) acquires data from various sensors and then posts the

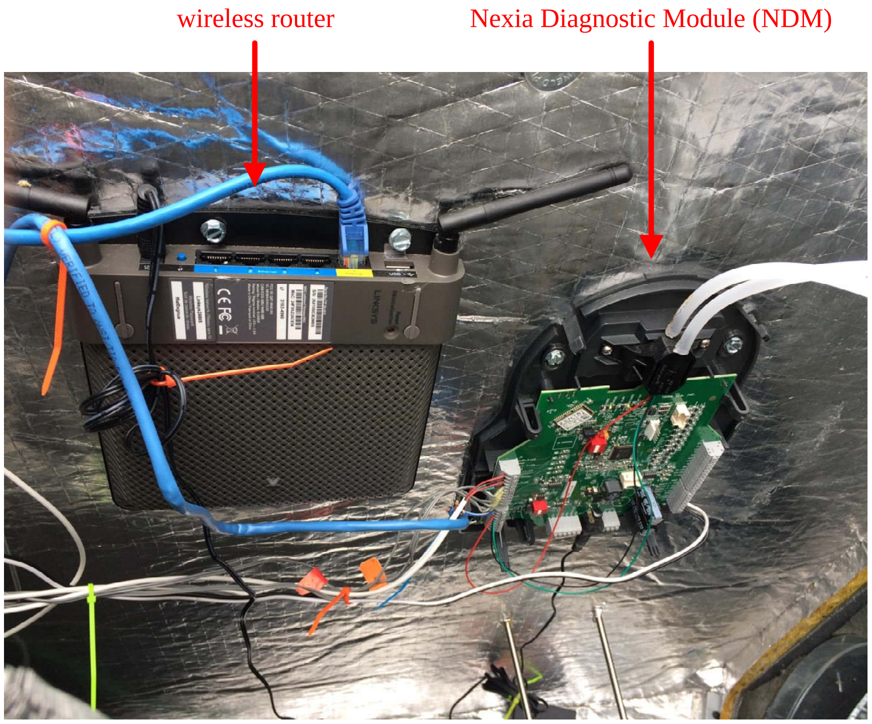


Figure A.4: The wireless router (Linksys EA 4800) and the Nexia Diagnositc Module (NDM) installed in the return air plenum of the packaged heat pump. The NDM requires a wireless internet connection, and the router was installed next to it in order to ensure a strong internet connection in the enclosure.

event-based data to a cloud-based database. The NDM requires a wireless internet connection to perform this task and is unable to be installed with a wired ethernet connection. This requirement is troublesome when the module must be installed in a location which does not receive a strong WiFi signal: inside the return air plenum of a packaged unit is one such example. In order to ensure that the diagnostic module could consistently receive a strong WiFi signal from within the packaged unit, a wireless router was installed alongside the NDM in the unit. Figure A.4 shows a closer photograph of of router and Nexia Diagnostic Module installed in the return air plenum of the unit. The router was set-up as a separate network to the main home wireless network and the NDM was the only device connected to this secondary network.

Estimating refrigerant mass flow using the methods presented in Chapter 4 requires an evaporator energy balance and therefore requires refrigerant temperature sensors. These sensors were

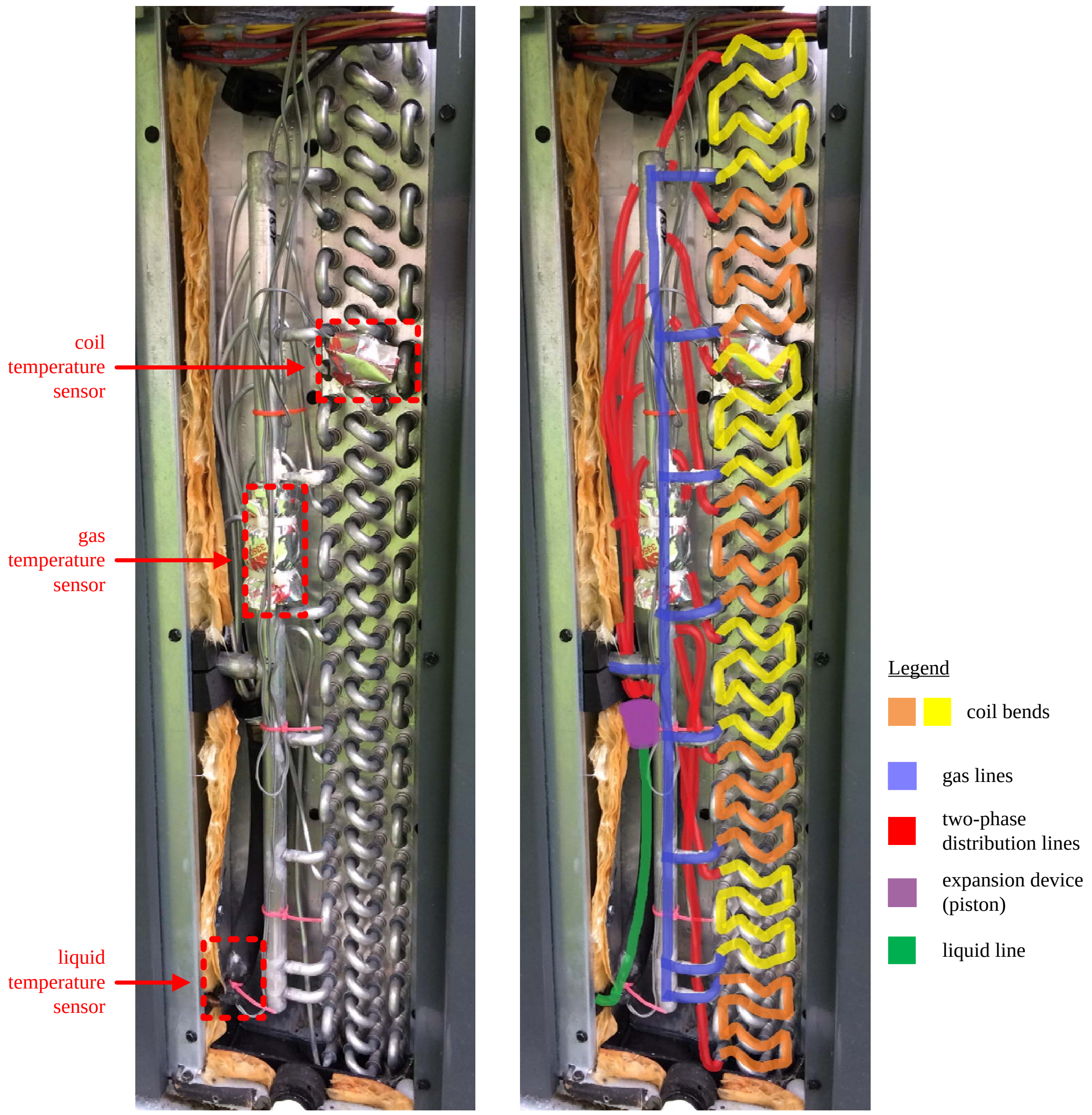


Figure A.5: Two (identical) photos of the side of the indoor coil. (left) The refrigerant temperature sensors are labeled. Note that the measurements from these sensors are only used in cooling mode. (right) Labeled refrigerant lines on the indoor coil showing the liquid line, piston expansion device, two-phase distribution lines, coil bends and passes, and the gas lines.

acquired with the NDM system and installed on the service side of the unit's indoor coil. Figure A.5 shows the side of the indoor unit coil after the service panel has been removed. Figure A.5 provides two identical photographs in order to label the refrigerant lines (right) and in order to show where the sensors are installed (left). The heat pump unit uses a piston expansion device which acts as a fixed orifice during cooling mode and which does not restrict flow when reversed during heating mode. A similar piston expansion device is installed on the condenser for heating mode operation. In Figure A.5, the liquid temperature sensor is installed just before the expansion device and the gas temperature sensor is installed at the outlet of the coil.

In order to install the coil temperature sensor in an appropriate location, the flow of refrigerant through the evaporator must be understood clearly. For the coil shown in Figure A.5, the refrigerant expands and then is distributed through eight branches. The flow of refrigerant through one of these eight branches is clearly outlined in Figure A.6. Each bend includes 16 passes and 15 bends. The coil temperature sensor is installed on the first bend that is on the service side. This is bend number 2 in Figure A.6. In a variety of operating conditions, the refrigerant will be two-phase after expanding and will then evaporate through the coils such that the exit of the coil is a super-heated

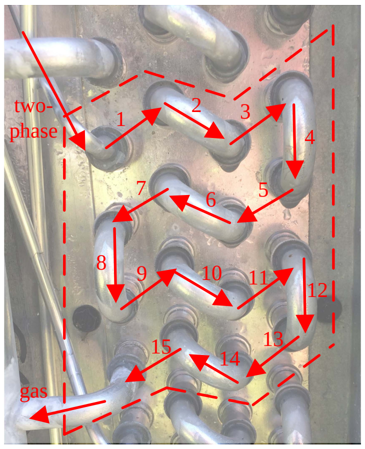


Figure A.6: A detailed view of the side of the indoor coil showing the bends for one of the eight branches. Each of the eight branches consists of sixteen passes and fifteen bends (seven bends on the side shown - even numbers, and eight bends on the opposite side - odd numbers). The coil temperature sensor was installed on bend no. 2 such that the refrigerant at the point of measurement will always be two-phase when operating in cooling mode.

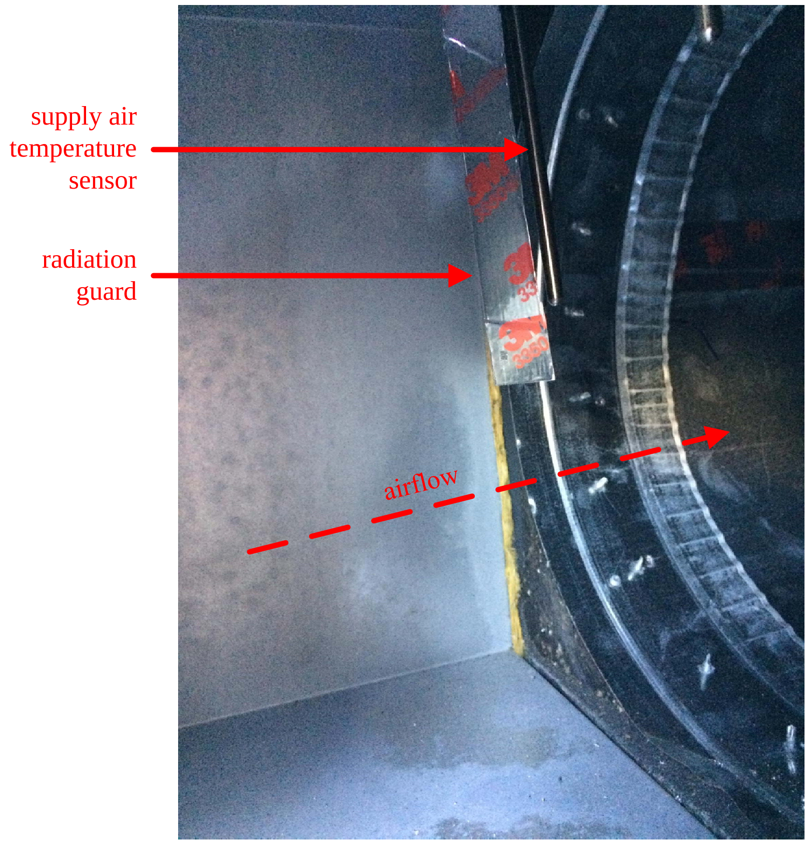


Figure A.7: A photos of the supply air duct inside the packaged heat pump. A sheet metal bracket was installed as a radiation guard upstream of the supply air temperature sensor. This prevents radiation from the supplementary electric resistive heating elements from interfering with the measurement of the supply air temperature when using the virtual airflow meter from Chapter 4.

vapor. The coil temperature sensor is therefore installed at the beginning of the flow in order to ensure that the measurement is in a two-phase region and not in the super-heated region.

Chapter 4 proposes a diagnostic method in which the airflow is determined by applying a known heat source and then performing an energy balance to the air as it passes through the air-handling portion of system. This known heat source would most often be electric resistive heating elements or a natural gas furnace. However, there is a very practical challenge when measuring the supply air temperature when these heat sources are applied; the surface temperature of an electric heating element or a gas heat exchanger will be very high such, if there is line-of-sight between the sensor and the hot surface, the radiation will be significant and the sensor will not measure the air temperature accurately. A 10 kW electric resistive heater coil (Goodman HKP-10C) was installed with the packaged unit in order to provide supplemental heat, and the unit airflow was estimated with this heat source. In order to obtain an accurate supply air temperature sensor measurement, an sheet metal bracket was installed next to the sensor in order to block the line-of-sight between the sensor and the heating elements. Figure A.7 shows a photograph of this installation.

Figure A.3 shows the location of the return air and supply air temperature sensors, and Figure A.5 shows the location of the refrigerant coil, gas, and liquid temperature sensors. However, the air-sensing method presented in Chapter 4 requires an electric current sensor and a return air relative humidity sensor in addition to the five temperature sensors. While an electric current sensor can be connected to the Nexia Diagnostic Module and the measurement can be uploaded to the cloud-based database, the measurement was found to be highly inaccurate; measuring a current greater than 20 A when the current clamp on a digital multimeter measures 12 A. Therefore, the electric current of the outdoor unit was measured using a digital multimeter (ETEKCITY MSR-C600). In order to deploy the methods presented herein, the accuracy of the NDM current sensor must be addressed.

In order to obtain a measurement for the return air humidity, the thermostat indoor conditions were used. While this will invariably introduce some error because the thermostat indoor conditions will be slightly different than the return air conditions, this simplification eliminates the need for an additional relative humidity sensors. In estimating the return air conditions, the temperature was measured directly using the NDM return air temperature sensor and the humidity ratio was obtained from the measured thermostat temperature and relative humidity.

The Nexia application on the iOS application was used to control the field-installed heat pump system. In addition to many other features, this application enables the user to (a) change the system mode between Heating, Cooling, Auto, and Off, (b) change the fan mode between On and Auto, (c) customize the setpoint schedule, and (d) change the current setpoint. The Trane smart thermostat (Trane XL824) uploads the event-based data to the cloud-based database and downloads any setting changes implemented remotely from the mobile application.

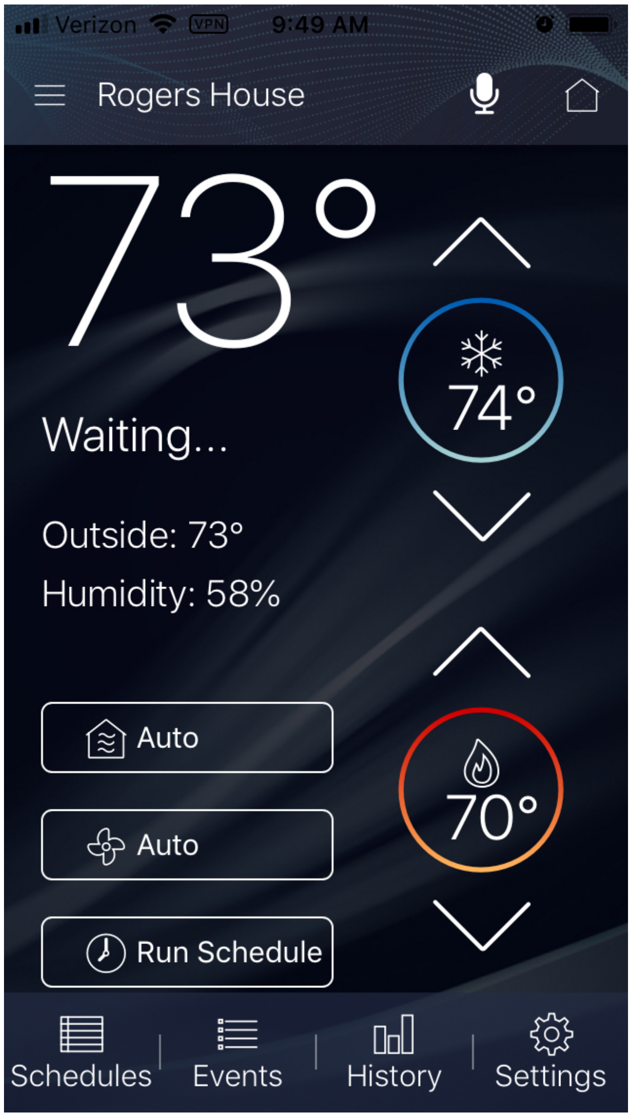


Figure A.8: A screenshot of the iOS mobile application that accompanies the Trane XL824 connected thermostat.

APPENDIX B

RELEVANT CODE

This appendix includes Python code used to analyze data throughout the dissertation.

B.1 A Class for Refrigerant Properties

The following script creates a class `r410a` for determining R-410a refrigerant properties. Tables are loaded as `numpy` arrays, the unit conversions are defined, the units are converted, and functions are created to interpolate the tables. The class also includes P-h and T-s diagrams in order to validate that the property functions are correct.

```
import numpy as np
import os
import scipy as sp
import matplotlib.pyplot as plt

class r410a():
  def __init__(self):
    self.load_data()
    self.define_conversions()
    self.convert()
    self.create_functions()

  def load_data(self):
    self.Hl_pt = np.load(os.path.dirname(__file__) + \
                  '/Hl_pt.npy') # kJ/kg
    self.Hv_pt = np.load(os.path.dirname(__file__) + \
                  '/Hv_pt.npy') # kJ/kg
    self.Sl_pt = np.load(os.path.dirname(__file__) + \
                  '/Sl_pt.npy') # kJ/kg/K
    self.Sv_pt = np.load(os.path.dirname(__file__) + \
                  '/Sv_pt.npy') # kJ/kg/K
    self.Tv = np.load(os.path.dirname(__file__) + \
                  '/Tv.npy') # C
    self.Tl = np.load(os.path.dirname(__file__) + \
                  '/Tl.npy') # C
    self.P = np.load(os.path.dirname(__file__) + \
                  '/P.npy') # kPa
    self.Tsat = np.load(os.path.dirname(__file__) + \
                  '/Tsat.npy') # C
    self.Psat = np.load(os.path.dirname(__file__) + \
                  '/Psat.npy') # kPa
    self.Hvsat = np.load(os.path.dirname(__file__) + \
                      '/Hvsat.npy') # kJ/kg
    self.Hlsat = np.load(os.path.dirname(__file__) + \
                      '/Hlsat.npy') # kJ/kg
    self.Svsat = np.load(os.path.dirname(__file__) + \
                      '/Svsat.npy') # kJ/kg/K
    self.Slsat = np.load(os.path.dirname(__file__) + \
                      '/Slsat.npy') # kJ/kg/K
    self.dHl_dP_Tl = np.load(os.path.dirname(__file__) + \
                          '/dHl_dP_Tl.npy') # (kJ/kg)/(kPa)
    self.dHv_dP_Tv = np.load(os.path.dirname(__file__) + \
                          '/dHv_dP_Tv.npy') # (kJ/kg)/(kPa)
    self.dHl_dTl_P = np.load(os.path.dirname(__file__) + \
                          '/dHl_dTl_P.npy') # (kJ/kg)/(C)
    self.dHv_dTv_P = np.load(os.path.dirname(__file__) + \
                          '/dHv_dTv_P.npy') # (kJ/kg)/(C)
    self.dPsat_dT = np.load(os.path.dirname(__file__) + \
                          '/dPsat_dT.npy') # (kPa)/(C)
    self.dTsat_dP = np.load(os.path.dirname(__file__) + \
                          '/dTsat_dP.npy') # (C)/(kPa)

  def define_conversions(self):
    self.kJ_kg_2_Btu_lbm = lambda x: x/1.055/2.20462
    self.kJ_kg_C_2_Btu_lbm_R = lambda x: x/1.055/2.20462/1.8
    self.C_2_F = lambda x: x*1.8+32 # celcius to Farenheit
    self.K_2_R = lambda x: x*1.8 # kelvin to rankine (or dC to dF)
    self.kPa_2_psi = lambda x: x/6.89476
    self.dkJkg_dkPa_2_dBtulbm_dpsi = lambda x: \
                          x/1.055/2.20462*6.89476
    self.dkJkg_dC_2_dBtulbm_dF = lambda x: x/1.055/2.20462/1.8
    self.dkPa_dC_2_dpsi_dF = lambda x: x/6.89476/1.8
    self.dC_dkPa_2_dF_dpsi= lambda x: x*1.8*6.89476

  def convert(self):
    self.Hl_pt = self.kJ_kg_2_Btu_lbm(self.Hl_pt)
    self.Hv_pt = self.kJ_kg_2_Btu_lbm(self.Hv_pt)
    self.Sv_pt = self.kJ_kg_C_2_Btu_lbm_R(self.Sv_pt)
    self.Sl_pt = self.kJ_kg_C_2_Btu_lbm_R(self.Sl_pt)
    self.Tl = self.C_2_F(self.Tl)
    self.Tv = self.C_2_F(self.Tv)
    self.P = self.kPa_2_psi(self.P)
    self.Tsat = self.C_2_F(self.Tsat)
    self.Psat = self.kPa_2_psi(self.Psat)
    self.Hvsat = self.kJ_kg_2_Btu_lbm(self.Hvsat)
    self.Hlsat = self.kJ_kg_2_Btu_lbm(self.Hlsat)
    self.Svsat = self.kJ_kg_C_2_Btu_lbm_R(self.Svsat)
    self.Slsat = self.kJ_kg_C_2_Btu_lbm_R(self.Slsat)
    self.dHl_dP_Tl = self.dkJkg_dkPa_2_dBtulbm_dpsi(self.dHl_dP_Tl)
    self.dHv_dP_Tv = self.dkJkg_dkPa_2_dBtulbm_dpsi(self.dHv_dP_Tv)
    self.dHl_dTl_P = self.dkJkg_dC_2_dBtulbm_dF(self.dHl_dTl_P)
    self.dHv_dTv_P = self.dkJkg_dC_2_dBtulbm_dF(self.dHv_dTv_P)
    self.dPsat_dT = self.dkPa_dC_2_dpsi_dF(self.dPsat_dT)
    self.dTsat_dP = self.dC_dkPa_2_dF_dpsi(self.dTsat_dP)

  def create_functions(self):
    self.Hl_func = sp.interpolate.interp2d(self.Tl, \
                          self.P, self.Hl_pt, kind='cubic')
    self.Hv_func = sp.interpolate.interp2d(self.Tv, \
                          self.P, self.Hv_pt, kind='cubic')
    self.Sl_func = sp.interpolate.interp2d(self.Tl, \
                          self.P, self.Sl_pt, kind='cubic')
    self.Sv_func = sp.interpolate.interp2d(self.Tv, \
                          self.P, self.Sv_pt, kind='cubic')
    self.Tsat_func = sp.interpolate.interp1d(self.Psat, \
                          self.Tsat, kind='cubic')
    self.Psat_func = sp.interpolate.interp1d(self.Tsat, \
                          self.Psat, kind='cubic')
    self.Hlsat_func = sp.interpolate.interp1d(self.Tsat, \
                          self.Hlsat, kind='cubic')
    self.Hvsat_func = sp.interpolate.interp1d(self.Tsat, \
                          self.Hvsat, kind='cubic')
    self.Slsat_func = sp.interpolate.interp1d(self.Tsat, \
                          self.Slsat, kind='cubic')
    self.Svsat_func = sp.interpolate.interp1d(self.Tsat, \
                          self.Svsat, kind='cubic')
    self.dHl_dP_Tl_func = sp.interpolate.interp2d(self.Tl, \
                          self.P, self.dHl_dP_Tl, kind='cubic')
    self.dHv_dP_Tv_func = sp.interpolate.interp2d(self.Tv, \
                          self.P, self.dHv_dP_Tv, kind='cubic')
    self.dHl_dTl_P_func = sp.interpolate.interp2d(self.Tl, \
                          self.P, self.dHl_dTl_P, kind='cubic')
    self.dHv_dTv_P_func = sp.interpolate.interp2d(self.Tv, \
                          self.P, self.dHv_dTv_P, kind='cubic')
    self.dPsat_dT_func = sp.interpolate.interp1d(self.Tsat, \
                          self.dPsat_dT, kind='cubic')
    self.dTsat_dP_func = sp.interpolate.interp1d(self.Psat, \
                          self.dTsat_dP, kind='cubic')

  def P_h_diagram(self):
    Tiso = np.arange(-60,151,10) # Isothermal lines

    # saturation pressure at the iso thermal lines
    Psat = self.Psat_func(Tiso)

    Hv_under_dome = np.squeeze([self.Hv_func(T,P) \
                          for (T,P) in zip(Tiso,Psat)])
    Hl_under_dome = np.squeeze([self.Hl_func(T,P) \
                          for (T,P) in zip(Tiso,Psat)])
    Hv_dome = self.Hvsat_func(self.Tsat)
    Hl_dome = self.Hlsat_func(self.Tsat)
    Hv = self.Hv_func(Tiso, self.P)
    Hl = self.Hl_func(Tiso, self.P)

    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(Hl, self.P, 'k', linewidth=0.5)
    ax.plot(Hv, self.P, 'k', linewidth=0.5)
    ax.plot([Hl_under_dome, Hv_under_dome], [Psat, Psat], 'k', \
                          linewidth=0.5)
    ax.plot(Hv_dome, self.Psat, color='red', linewidth=3)
    ax.plot(Hl_dome, self.Psat, color='red', linewidth=3)
    ax.set_xlabel('Enthalpy (Btu/lbm)')
    ax.set_ylabel('Pressure (psia)')
    return fig, ax

  def T_s_diagram(self):
    Piso = np.arange(25, 626, 50)
    Tsat = self.Tsat_func(Piso)
    Sv_under_dome = np.squeeze([self.Sv_func(T,P) \
                          for (T,P) in zip(Tsat, Piso)])
    Sl_under_dome = np.squeeze([self.Sl_func(T,P) \
                          for (T,P) in zip(Tsat, Piso)])
    Sv_dome = self.Svsat_func(self.Tsat)
    Sl_dome = self.Slsat_func(self.Tsat)
    Sv = np.transpose(self.Sv_func(self.Tv, Piso))
    Sl = np.transpose(self.Sv_func(self.Tl, Piso))

    Sv[Sv<np.tile(Sv_under_dome, (150,1))] = np.nan
    Sl[Sl>np.tile(Sl_under_dome, (150,1))] = np.nan

    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.plot(Sv, self.Tv, 'k', linewidth=0.5)
    ax.plot(Sl, self.Tl, 'k', linewidth=0.5)
    ax.plot([Sl_under_dome, Sv_under_dome], [Tsat, Tsat], 'k', \
                          linewidth=0.5)
    ax.plot(Sv_dome, self.Tsat, color='red', linewidth=3)
    ax.plot(Sl_dome, self.Tsat, color='red', linewidth=3)
    ax.set_xlabel('Entropy (Btu/lbm/R)')
    ax.set_ylabel('Temperature (F)')
    return fig, ax
```

B.2 Uncertainty and Sensitivity Analysis of Virtual Sensors

The following two scripts analyze the uncertainty and sensitivity in two sets of virtual sensors given assumptions about the uncertainty in the inputs. The `vu` class is created to track the **V**alue and the **U**ncertainty in each variable. Each variable is a member of the `vu` class, and each member of this class has 3 objects: the value `v`, the absolute uncertainty `au`, and the percent uncertainty `pu`. The uncertainty may be set using either the absolute or the percent uncertainty (`set_a` and `set_p` functions). The uncertainty is propagated using partial derivatives, and the sensitivity is determined using these partial derivatives. The sensitivity is organized with a pandas dataframe called `frame`. The final part of each script fills the fields of a pdf with the uncertainty values.

B.2.1 Analysis of the Refrigerant Sensing Method

```
import sys
import importlib

refprop_path = "C:/Users/arogers9/Syncplicity Folders/TexasAM/" \
               "0 Publications Presentations Proposals/" \
               "2019 - STBE - Residential FDD/Python/refprop"

if refprop_path not in sys.path:
  sys.path.append(refprop_path)
if 'load_refprop' not in sys.modules:
  import load_refprop
else:
  importlib.reload(load_refprop)

import numpy as np
import pdfrw
import pandas as pd

class vu(): # uncertainty stored as absolute
  def __init__(self):
    self.v=[]
    self.au=[]
  def set_p(self,v,u): # input the  uncertainty as Percentage
    self.v = v # value
    self.au = (u/100)*v # percent/100 * value
    self.pu = u
  def set_a(self,v,u): # input uncertainty as absolute
    self.v = v
    self.au = u
    self.pu = (u/v)*100 # (absolute/value)*100

ref = load_refprop.r410a()

frame = pd.DataFrame(
           columns = ['v','au','pu','Vdot_air', 'Q', 'COP',
                       'mdotr'],
           index=['Tret','Tsup','phi_ret','Tevap','Tgas','Tid_liq',
                   'Tcond','Tdis','Tsuc','Iod','cp_air','rho_air',
                   'Psys','Mratio','hwe','PXV','Pgas_loss',
                   'Pdis_loss','Psuc_loss','Ifan','V','alpha'],
           data = 0)

temp1 = vu()
temp2 = vu()
temp3 = vu()

Tcond = vu()
Pdis_loss = vu()
Pcond = vu()
Pdis = vu()
Tevap = vu()
Pevap = vu()
Psuc_loss = vu()
Psuc = vu()
Wcomp = vu()
Tsat_dis = vu()
Tsat_suc = vu()
alpha = vu()
Qloss = vu()
Tdis = vu()
hdis = vu()
Tsuc = vu()
hsuc = vu()
Dhcomp = vu()
mdotr = vu()
Pgas_loss = vu()
Pgas = vu()
Tgas = vu()
Tid_liq = vu()
Pid_liq = vu()
PXV = vu()
hgas = vu()
hid_liq = vu()
Dhevap = vu()
Q = vu()
Iod = vu()
Ifan = vu()
Icomp = vu()
V = vu()
Tret = vu()
Tsup = vu()
cp_air = vu()
Dhair_dry = vu()
Psat_ret = vu()
Psat_sup = vu()
phi_ret = vu()
Psys = vu()
Mratio = vu()
om_ret = vu()
om_sup = vu()
hwe = vu()
Dhair_wet = vu()
mdot_air = vu()
rho_air = vu()
Vdot_air = vu()
COP = vu()

Tcond.set_p(130,2) # F
Pdis_loss.set_p(10,60) # psi
Tevap.set_p(50,2) # F
```

```
Psuc_loss.set_p(15,60) # psi
Tgas.set_p(50,2) # F
Tid_liq.set_p(75,2) # F
Pgas_loss.set_p(5,60) # psi
PXV.set_p(250, 50) # psi
Iod.set_p(13.5,5) # amps
Ifan.set_p(0.5,50) # amps
V.set_p(240,5) # volts
alpha.set_p(0.05,60) # unitless
Tdis.set_p(195,2) # F
Tsuc.set_p(60,2) # F

Tret.set_p(72,2) # F
Tsup.set_p(52,2) # F
phi_ret.set_p(0.7,5) # unitless
cp_air.set_p(0.24,1) # Btu/lb/F
hwe.set_p(1061,1) # Btu/lb
Mratio.set_p(0.622,1) # unitless
Psys.set_p(14.7, 2)
rho_air.set_p(0.0765,3) # lbm/ft3

## Discharge Pressure
Pcond_Tcond = ref.dPsat_dT_func(Tcond.v)
Pcond.set_a( # psi
        ref.Psat_func(Tcond.v),
        np.linalg.norm([Pcond_Tcond*Tcond.au]) )

Pdis_Pdis_loss = 1
Pdis_Pcond = 1
Pdis.set_a( # psi
        Pdis_loss.v + Pcond.v,
        np.linalg.norm([Pdis_Pdis_loss*Pdis_loss.au,
                        Pdis_Pcond*Pcond.au]) )

## Suction Pressure
Pevap_Tevap = ref.dPsat_dT_func(Tevap.v)
Pevap.set_a( # psi
        ref.Psat_func(Tevap.v),
        np.linalg.norm([Pevap_Tevap*Tevap.au]) )

Psuc_Pevap = 1
Psuc_Psuc_loss = -1
Psuc.set_a( # psi
        Pevap.v - Psuc_loss.v,
        np.linalg.norm([Psuc_Pevap*Pevap.au,
                        Psuc_Psuc_loss*Psuc_loss.au]) )

## Compressor Power
Tsat_dis_Pdis = ref.dTsat_dP_func(Pdis.v)
Tsat_dis.set_a(
        ref.Tsat_func(Pdis.v),
        np.linalg.norm([Tsat_dis_Pdis*Pdis.au]) )

Tsat_suc_Psuc = ref.dTsat_dP_func(Psuc.v)
Tsat_suc.set_a(
        ref.Tsat_func(Psuc.v),
        np.linalg.norm([Tsat_suc_Psuc*Psuc.au]) )

Icomp_Iod = 1
Icomp_Ifan = -1
Icomp.set_a( # amps
        Iod.v - Ifan.v,
        np.linalg.norm([Icomp_Iod*Iod.au, Icomp_Ifan*Ifan.au]) )

Wcomp_Icomp = V.v
```

```
Wcomp_V = Icomp.v
Wcomp.set_a( # Watts
        Icomp.v*V.v,
        np.linalg.norm([Wcomp_V*V.au, Wcomp_Icomp*Icomp.au]) )
Wcomp.set_a(Wcomp.v/1000, Wcomp.au/1000) # kiloWatts
Wcomp_Icomp *= 1/1000
Wcomp_V *= 1/1000

Qloss_Wcomp = alpha.v
Qloss_alpha = Wcomp.v
Qloss.set_a( # kW
        Wcomp.v*alpha.v,
        np.linalg.norm([Qloss_Wcomp*Wcomp.au,
                        Qloss_alpha*alpha.au]) )

## compressor enthalpy rise
hdis_Pdis = ref.dHv_dP_Tv_func(Tdis.v,Pdis.v)
hdis_Tdis = ref.dHv_dTv_P_func(Tdis.v,Pdis.v)
hdis.set_a( # Btu/lbm
        ref.Hv_func(Tdis.v,Pdis.v)[0],
        np.linalg.norm([hdis_Pdis*Pdis.au, hdis_Tdis*Tdis.au]) )

hsuc_Tsuc = ref.dHv_dTv_P_func(Tsuc.v,Psuc.v)
hsuc_Psuc = ref.dHv_dP_Tv_func(Tsuc.v,Psuc.v)
hsuc.set_a( #Btu/lbm
        ref.Hv_func(Tsuc.v, Psuc.v)[0],
        np.linalg.norm([hsuc_Psuc*Psuc.au, hsuc_Tsuc*Tsuc.au]) )

Dhcomp_hdis = 1
Dhcomp_hsuc = -1
Dhcomp.set_a( # Btu/lbm
        hdis.v - hsuc.v,
        np.linalg.norm([Dhcomp_hdis*hdis.au, Dhcomp_hsuc*hsuc.au]))


## Mass Flow Rate
temp1_Wcomp = 1
temp1_Qloss = -1
temp1.set_a( # kW
        Wcomp.v - Qloss.v,
        np.linalg.norm([temp1_Wcomp*Wcomp.au,
                        temp1_Qloss*Qloss.au]) )

mdotr_temp1 = 1/Dhcomp.v
mdotr_Dhcomp = -temp1.v/(Dhcomp.v**2)
mdotr.set_a( # (lbm/s)*(kJ/Btu)
temp1.v/Dhcomp.v,
np.linalg.norm([mdotr_temp1*temp1.au,
                mdotr_Dhcomp*Dhcomp.au]) )
mdotr.set_a(mdotr.v*3600/1.00506, mdotr.au*3600/1.05506) # lbm/hr
mdotr_temp1 *= 3600/1.00506
mdotr_Dhcomp *= 3600/1.00506


frame.loc['Tcond','mdotr'] = mdotr_Dhcomp*Dhcomp_hdis*hdis_Pdis* \
     Pdis_Pcond*Pcond_Tcond
frame.loc['Pdis_loss','mdotr'] = mdotr_Dhcomp*Dhcomp_hdis* \
     hdis_Pdis*Pdis_Pdis_loss
frame.loc['Tdis','mdotr'] = mdotr_Dhcomp*Dhcomp_hdis*hdis_Tdis
frame.loc['Tevap','mdotr'] = mdotr_Dhcomp*Dhcomp_hsuc*hsuc_Psuc* \
     Psuc_Pevap*Pevap_Tevap
frame.loc['Psuc_loss','mdotr'] = mdotr_Dhcomp*Dhcomp_hsuc* \
     hsuc_Psuc*Psuc_Psuc_loss
frame.loc['Tsuc','mdotr'] = mdotr_Dhcomp*Dhcomp_hsuc*hsuc_Tsuc
frame.loc['alpha','mdotr'] = mdotr_temp1*temp1_Qloss*Qloss_alpha
frame.loc['V','mdotr'] = mdotr_temp1*(temp1_Qloss*Qloss_Wcomp + \
     temp1_Wcomp)*Wcomp_V
frame.loc['Ifan','mdotr'] = mdotr_temp1*(temp1_Qloss* \
     Qloss_Wcomp + temp1_Wcomp)*Wcomp_Icomp*Icomp_Ifan
frame.loc['Iod','mdotr'] = mdotr_temp1*(temp1_Qloss*Qloss_Wcomp + \
     temp1_Wcomp)*Wcomp_Icomp*Icomp_Iod

## Evaporator enthalpy rise
Pid_liq_Pevap = 1
Pid_liq_PXV = 1
Pid_liq.set_a( # psi
Pevap.v + PXV.v,
np.linalg.norm([Pid_liq_Pevap*Pevap.au, Pid_liq_PXV*PXV.au]) )


Pgas_Pevap = 1
Pgas_Pgas_loss = -1
Pgas.set_a( # psi
     Pevap.v - Pgas_loss.v,
     np.linalg.norm([Pgas_Pevap*Pevap.au,
                     Pgas_Pgas_loss*Pgas_loss.au]) )

hid_liq_Tid_liq = ref.dHl_dTl_P_func(Tid_liq.v,Pid_liq.v)
hid_liq_Pid_liq = ref.dHl_dP_Tl_func(Tid_liq.v,Pid_liq.v)
hid_liq.set_a( # Btu/lbm
      ref.Hl_func(Tid_liq.v, Pid_liq.v)[0],
      np.linalg.norm([hid_liq_Tid_liq*Tid_liq.au,
                      hid_liq_Pid_liq*Pid_liq.au]) )

hgas_Tgas = ref.dHv_dTv_P_func(Tgas.v,Pgas.v)
hgas_Pgas = ref.dHv_dP_Tv_func(Tgas.v,Pgas.v)
hgas.set_a( # Btu/lbm
      ref.Hv_func(Tgas.v, Pgas.v)[0],
      np.linalg.norm([hgas_Tgas*Tgas.au, hgas_Pgas*Pgas.au]) )

Dhevap_hgas = 1
Dhevap_hid_liq = -1
Dhevap.set_a( # Btu/lbm
      hgas.v - hid_liq.v,
      np.linalg.norm([Dhevap_hgas*hgas.au,
                      Dhevap_hid_liq*hid_liq.au]) )

## Capacity
Q_mdotr = Dhevap.v
Q_Dhevap = mdotr.v
Q.set_a( # Btu/hr
      mdotr.v*Dhevap.v,
      np.linalg.norm([Q_mdotr*mdotr.au, Q_Dhevap*Dhevap.au]) )

frame.loc['Tcond','Q'] = Q_mdotr*frame.loc['Tcond','mdotr']
frame.loc['Pdis_loss','Q'] = Q_mdotr*frame.loc['Pdis_loss','mdotr']
frame.loc['Tdis','Q'] = Q_mdotr*frame.loc['Tdis','mdotr']
frame.loc['Tevap','Q'] = (Q_mdotr*mdotr_Dhcomp*Dhcomp_hsuc* \
       hsuc_Psuc*Psuc_Pevap +
Q_Dhevap*(Dhevap_hgas*hgas_Pgas*Pgas_Pevap + Dhevap_hid_liq* \
       hid_liq_Pid_liq*Pid_liq_Pevap))*Pevap_Tevap
frame.loc['Psuc_loss','Q'] = Q_mdotr*frame.loc['Psuc_loss','mdotr']
frame.loc['Tsuc','Q'] = Q_mdotr*frame.loc['Tsuc','mdotr']
frame.loc['alpha','Q'] = Q_mdotr*frame.loc['alpha','mdotr']
frame.loc['V','Q'] = Q_mdotr*frame.loc['V','mdotr']
frame.loc['Ifan','Q'] = Q_mdotr*frame.loc['Ifan','mdotr']
frame.loc['Iod','Q'] = Q_mdotr*frame.loc['Iod','mdotr']
frame.loc['Tid_liq','Q'] = Q_Dhevap*Dhevap_hid_liq*hid_liq_Tid_liq
frame.loc['PXV','Q'] = Q_Dhevap*Dhevap_hid_liq*hid_liq_Pid_liq* \
       Pid_liq_PXV
```

```
frame.loc['Tgas','Q'] = Q_Dhevap*Dhevap_hgas*hgas_Tgas
frame.loc['Pgas_loss','Q'] = Q_Dhevap*Dhevap_hgas*hgas_Pgas* \
       Pgas_Pgas_loss

## COP
COP_Q = 1/Wcomp.v
COP_Wcomp = -Q.v/(Wcomp.v**2)
COP.set_a( # (Btu/hr)/kW
Q.v/Wcomp.v,
np.linalg.norm([COP_Q*Q.au, COP_Wcomp*Wcomp.au]))
COP.set_a(COP.v/3412.142, COP.au/3412.142)
COP_Q *= 1/3412.142
COP_Wcomp *= 1/3412.142

frame.loc['Tcond','COP'] = COP_Q*frame.loc['Tcond','Q']
frame.loc['Pdis_loss','COP'] = COP_Q*frame.loc['Pdis_loss','Q']
frame.loc['Tdis','COP'] = COP_Q*frame.loc['Tdis','Q']
frame.loc['Tevap','COP'] = COP_Q*frame.loc['Tevap','Q']
frame.loc['Psuc_loss','COP'] = COP_Q*frame.loc['Psuc_loss','Q']
frame.loc['Tsuc','COP']  = COP_Q*frame.loc['Tsuc','Q']
frame.loc['alpha','COP'] = COP_Q*frame.loc['alpha','Q']
frame.loc['Tid_liq','COP'] = COP_Q*frame.loc['Tid_liq','Q']
frame.loc['PXV','COP'] = COP_Q*frame.loc['PXV','Q']
frame.loc['Tgas','COP'] = COP_Q*frame.loc['Tgas','Q']
frame.loc['Pgas_loss','COP'] = COP_Q*frame.loc['Pgas_loss','Q']
frame.loc['V','COP'] = COP_Q*frame.loc['V','Q'] + COP_Wcomp*Wcomp_V
frame.loc['Ifan','COP'] = COP_Q*frame.loc['Ifan','Q'] + \
       COP_Wcomp*Wcomp_Icomp*Icomp_Ifan
frame.loc['Iod','COP'] = COP_Q*frame.loc['Iod','Q'] + \
       COP_Wcomp*Wcomp_Icomp*Icomp_Iod

# Farenheit
c = [77.345+0.0057*5/9*459.67, 0.0057*5/9, -7235*9/5, 459.67,
         np.power(5/9,-8.2)]
f1 = lambda x: (c[4]*np.exp(c[0] + c[1]*x + c[2]/(x+c[3]))* \
         np.power(x+c[3],-8.2))/6894.76
df1 = lambda x: (c[4]*(c[1]-c[2]*np.power(x+c[3],-2))*np.exp(c[0] \
       + c[1]*x + c[2]/(x+c[3]))*np.power(x+c[3],-8.2) \
       - 8.2*c[4]*np.exp(c[0] + c[1]*x + c[2]/(x+c[3]))* \
       np.power(x+c[3],-9.2))/6894.76
f2 = lambda x,y: np.min([x,y])

temp1_Tret = 1
temp1_Tsup = -1
temp1.set_a(
Tret.v - Tsup.v,
np.linalg.norm([temp1_Tret*Tret.au, temp1_Tsup*Tsup.au]) )

Dhair_dry_cp_air = temp1.v
Dhair_dry_temp1 = cp_air.v
Dhair_dry.set_a(
cp_air.v*temp1.v,
np.linalg.norm([Dhair_dry_cp_air*cp_air.au,
                  Dhair_dry_temp1*temp1.au]) )

Psat_ret_Tret = df1(Tret.v)
Psat_ret.set_a(
f1(Tret.v),
np.linalg.norm([Psat_ret_Tret*Tret.au ]) )

Psat_sup_Tsup = df1(Tsup.v)
Psat_sup.set_a(
f1(Tsup.v),
np.linalg.norm([Psat_sup_Tsup*Tsup.au ]) )

```

```
temp1_phi_ret = Psat_ret.v
temp1_Psat_ret = phi_ret.v
temp1.set_a(
phi_ret.v*Psat_ret.v,
np.linalg.norm([temp1_Psat_ret*Psat_ret.au,
                  temp1_phi_ret*phi_ret.au]) )

temp2_Psys = 1
temp2_temp1 = -1
temp2.set_a(
Psys.v - temp1.v,
np.linalg.norm([temp2_Psys*Psys.au, temp2_temp1*temp1.au]) )

temp3_temp1 = 1/temp2.v
temp3_temp2 = -temp1.v/temp2.v**2
temp3.set_a(
temp1.v/temp2.v,
np.linalg.norm([temp3_temp1*temp1.au, temp3_temp2*temp2.au]) )

om_ret_Mratio = temp3.v
om_ret_temp3 = Mratio.v
om_ret.set_a(
Mratio.v*temp3.v,
np.linalg.norm([om_ret_Mratio*Mratio.au, om_ret_temp3*temp3.au]) )

# intermdiate
om_ret_Mratio = om_ret_Mratio
om_ret_Psys = om_ret_temp3*temp3_temp2*temp2_Psys
om_ret_phi_ret = om_ret_temp3*(temp3_temp2*temp2_temp1 + \
     temp3_temp1)*temp1_phi_ret
om_ret_Tret = om_ret_temp3*(temp3_temp2*temp2_temp1 + \
     temp3_temp1)*temp1_Psat_ret*Psat_ret_Tret

temp1_Psys = 1
temp1_Psat_sup = -1
temp1.set_a(
Psys.v - Psat_sup.v,
np.linalg.norm([temp1_Psys*Psys.au, temp1_Psat_sup*Psat_sup.au]) )

temp2_Psat_sup = 1/temp1.v
temp2_temp1 = -Psat_sup.v/temp1.v**2
temp2.set_a(
Psat_sup.v/temp1.v,
np.linalg.norm([temp2_Psat_sup*Psat_sup.au,
                  temp2_temp1*temp1.au]) )

temp3_Mratio = temp2.v
temp3_temp2 = Mratio.v
temp3.set_a(
Mratio.v*temp2.v,
np.linalg.norm([temp3_Mratio*Mratio.au, temp3_temp2*temp2.au]) )

om_sup_temp3 = [1, 0][np.argmin([temp3.v, om_ret.v])]
om_sup_om_ret = [0, 1][np.argmin([temp3.v, om_ret.v])]
om_sup.set_a(
np.min([temp3.v, om_ret.v]),
[temp3.au, om_ret.au][np.argmin([temp3.v, om_ret.v])] )

#intermediate
om_sup_Mratio = om_sup_om_ret*om_ret_Mratio + \
     om_sup_temp3*temp3_Mratio
om_sup_Psys = om_sup_om_ret*om_ret_Psys + om_sup_temp3 \
     *temp3_temp2*temp2_temp1*temp1_Psys
om_sup_phi_ret = om_sup_om_ret*om_ret_phi_ret
om_sup_Tret = om_sup_om_ret*om_ret_Tret
```

```
om_sup_Tsup = om_sup_temp3*temp3_temp2*(temp2_temp1* \
     temp1_Psat_sup + temp2_Psat_sup)*Psat_sup_Tsup

temp1_om_ret = 1
temp1_om_sup = -1
temp1.set_a(
om_ret.v - om_sup.v,
np.linalg.norm([temp1_om_ret*om_ret.au, temp1_om_sup*om_sup.au]) )

temp2_hwe = temp1.v
temp2_temp1 = hwe.v
temp2.set_a(
hwe.v*temp1.v,
np.linalg.norm([temp2_hwe*hwe.au, temp2_temp1*temp1.au]) )

Dhair_wet_Dhair_dry = 1
Dhair_wet_temp2 = 1
Dhair_wet.set_a( # Btu/lbm
Dhair_dry.v + temp2.v,
np.linalg.norm([Dhair_wet_Dhair_dry*Dhair_dry.au,
                 Dhair_wet_temp2*temp2.au]) )

mdot_air_Q = 1/Dhair_wet.v
mdot_air_Dhair_wet = -Q.v/(Dhair_wet.v**2)
mdot_air.set_a( # lbm/hr
Q.v/Dhair_wet.v,
np.linalg.norm([mdot_air_Q*Q.au,
                 mdot_air_Dhair_wet*Dhair_wet.au]) )

Vdot_air_mdot_air = 1/rho_air.v
Vdot_air_rho_air = -mdot_air.v/(rho_air.v**2)
Vdot_air.set_a( # ft/hr
mdot_air.v/rho_air.v,
np.linalg.norm([Vdot_air_mdot_air*mdot_air.au,
                 Vdot_air_rho_air*rho_air.au]) )
Vdot_air.set_a(Vdot_air.v/60, Vdot_air.au/60) # cfm
Vdot_air_rho_air *= 1/60
Vdot_air_mdot_air *= 1/60

frame.loc['Tcond','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Tcond','Q']
frame.loc['Pdis_loss','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Pdis_loss','Q']
frame.loc['Tdis','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Tdis','Q']
frame.loc['Tevap','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Tevap','Q']
frame.loc['Psuc_loss','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Psuc_loss','Q']
frame.loc['Tsuc','Vdot_air']  = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Tsuc','Q']
frame.loc['alpha','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['alpha','Q']
frame.loc['V','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['V','Q']
frame.loc['Ifan','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Ifan','Q']
frame.loc['Iod','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Iod','Vdot_air']
frame.loc['Tid_liq','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Tid_liq','Q']
frame.loc['PXV','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['PXV','Q']
frame.loc['Tgas','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Tgas','Q']
frame.loc['Pgas_loss','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Q* \
     frame.loc['Pgas_loss','Q']
frame.loc['rho_air','Vdot_air'] = Vdot_air_rho_air
frame.loc['hwe','Vdot_air'] = Vdot_air_mdot_air* \
         mdot_air_Dhair_wet*Dhair_wet_temp2*temp2_hwe
frame.loc['Mratio','Vdot_air'] = Vdot_air_mdot_air* \
     mdot_air_Dhair_wet*Dhair_wet_temp2*temp2_temp1*( \
     temp1_om_sup*om_sup_Mratio + temp1_om_ret*om_ret_Mratio)
frame.loc['Psys','Vdot_air'] = Vdot_air_mdot_air* \
         mdot_air_Dhair_wet*Dhair_wet_temp2*temp2_temp1* \
         (temp1_om_sup*om_sup_Psys + temp1_om_ret*om_ret_Psys)
frame.loc['phi_ret','Vdot_air'] = Vdot_air_mdot_air* \
         mdot_air_Dhair_wet*Dhair_wet_temp2*temp2_temp1*( \
         temp1_om_sup*om_sup_phi_ret + temp1_om_ret*om_ret_phi_ret)
frame.loc['Tret','Vdot_air'] = Vdot_air_mdot_air* \
         mdot_air_Dhair_wet*(Dhair_wet_temp2*temp2_temp1* \
         (temp1_om_sup*om_sup_Tret + temp1_om_ret*om_ret_Tret) \
         + Dhair_wet_Dhair_dry*Dhair_dry_temp1*temp1_Tret)
frame.loc['Tsup','Vdot_air'] = Vdot_air_mdot_air* \
         mdot_air_Dhair_wet*(Dhair_wet_temp2*temp2_temp1* \
         temp1_om_sup*om_sup_Tsup + \
         Dhair_wet_Dhair_dry*Dhair_dry_temp1*temp1_Tsup)
frame.loc['cp_air','Vdot_air'] = Vdot_air_mdot_air* \
         mdot_air_Dhair_wet*Dhair_wet_Dhair_dry*Dhair_dry_cp_air

for i,r in frame.iterrows():
  frame.loc[i,'v'] = eval(i).v
  frame.loc[i,'au'] = eval(i).au
  frame.loc[i,'pu'] = eval(i).pu

for c in frame.columns.drop(['v','pu','au']):
  frame[c] = frame[c]*frame.v.abs()/np.abs(eval(c).v)

fields = {'Pdis_loss':      u'{0:.0f}\u00B1{1:.0f}%',
          'Tdis':           u'{0:.0f}\u00B1{1:.0f}%',
          'Tcond':          u'{0:.0f}\u00B1{1:.0f}%',
          'Pcond':          u'{0:.1f}\u00B1{1:.1f}%',
          'Pdis':           u'{0:.1f}\u00B1{1:.1f}%',
          'hdis':           u'{0:.1f}\u00B1{1:.2f}%',
          'Tid_liq':        u'{0:.0f}\u00B1{1:.0f}%',
          'hid_liq':        u'{0:.1f}\u00B1{1:.2f}%',
          'Pid_liq':        u'{0:.1f}\u00B1{1:.1f}%',
          'Tgas':           u'{0:.0f}\u00B1{1:.0f}%',
          'Dhevap':         u'{0:.1f}\u00B1{1:.2f}%',
          'Pgas':           u'{0:.1f}\u00B1{1:.1f}%',
          'hgas':           u'{0:.1f}\u00B1{1:.2f}%',
          'Pgas_loss':      u'{0:.0f}\u00B1{1:.0f}%',
          'Psuc_loss':      u'{0:.0f}\u00B1{1:.0f}%',
          'Tsuc':           u'{0:.0f}\u00B1{1:.0f}%',
          'Tevap':          u'{0:.0f}\u00B1{1:.0f}%',
          'Pevap':          u'{0:.1f}\u00B1{1:.1f}%',
          'Psuc':           u'{0:.1f}\u00B1{1:.1f}%',
          'hsuc':           u'{0:.1f}\u00B1{1:.2f}%',
          'Dhcomp':         u'{0:.1f}\u00B1{1:.1f}%',
          'PXV':            u'{0:.0f}\u00B1{1:.0f}%',
          'Ifan':           u'{0:.2f}\u00B1{1:.0f}%',
          'V':              u'{0:.0f}\u00B1{1:.0f}%',
          'mdotr':          u'{0:.1f}\u00B1{1:.1f}%',
          'Iod':            u'{0:.2f}\u00B1{1:.0f}%',
          'Icomp':          u'{0:.2f}\u00B1{1:.1f}%',
          'Wcomp':          u'{0:.2f}\u00B1{1:.1f}%',
          'Qloss':          u'{0:.2f}\u00B1{1:.1f}%',
          'alpha':          u'{0:.2f}\u00B1{1:.0f}%',
          'Q':              u'{0:.0f}\u00B1{1:.1f}%',
          'cp_air':         u'{0:.2f}\u00B1{1:.0f}%',
          'COP':            u'{0:.2f}\u00B1{1:.1f}%',
          'Dhair_dry':      u'{0:.2f}\u00B1{1:.1f}%',
          'Dhair_wet':      u'{0:.2f}\u00B1{1:.1f}%',
          'mdot_air':       u'{0:.1f}\u00B1{1:.1f}%',
          'Vdot_air':       u'{0:.0f}\u00B1{1:.1f}%',
          'Tsup':           u'{0:.0f}\u00B1{1:.0f}%',
          'Psat_sup':       u'{0:.3f}\u00B1{1:.1f}%',
          'om_sup':         u'{0:.5f}\u00B1{1:.1f}%',
          'rho_air':        u'{0:.3f}\u00B1{1:.0f}%',
          'phi_ret':        u'{0:.2f}\u00B1{1:.0f}%',
          'Psys':           u'{0:.1f}\u00B1{1:.0f}%',
          'Mratio':         u'{0:.3f}\u00B1{1:.0f}%',
          'hwe':            u'{0:.0f}\u00B1{1:.0f}%',
          'Tret':           u'{0:.0f}\u00B1{1:.0f}%',
          'Psat_ret':       u'{0:.3f}\u00B1{1:.1f}%',
          'om_ret':         u'{0:.5f}\u00B1{1:.1f}%'}

template = 'vsm1.pdf'
output = 'vsm1_filled.pdf'
pdf = pdfrw.PdfReader(template)
annots = pdf.pages[0]['/Annots']
for annot in annots:
  subtype = annot['/Subtype']
  if subtype == '/Widget':
    key = annot['/T'][1:-1]
    if key in fields:
      text = fields[key].format(eval(key).v, eval(key).pu)
      annot.update(pdfrw.PdfDict(V='{}'.format(text)))
      annot.update(pdfrw.PdfDict(AP=''))
pdfrw.PdfWriter().write(output, pdf)
```

B.2.2 Analysis of the Air Sensing Method

```
import sys
import importlib

deeptrane_path = "C:/Users/arogers9/Syncplicity Folders/" \
                    "GitHub/TraneProject"
refprop_path = "C:/Users/arogers9/Syncplicity Folders/" \
               "GitHub/TraneProject/austin/refprop"

if deeptrane_path not in sys.path:
  sys.path.append(deeptrane_path)
if refprop_path not in sys.path:
  sys.path.append(refprop_path)
if 'deeptrane_python' not in sys.modules:
  import deeptrane_python
else:
  importlib.reload(deeptrane_python)
if 'load_refprop' not in sys.modules:
  import load_refprop
else:
  importlib.reload(load_refprop)

import numpy as np
import pdfrw
import pandas as pd

class vu(): # uncertainty stored as absolute
  def __init__(self):
    self.v=[]
    self.au=[]
  def set_p(self,v,u): # input the  uncertainty as Percentage
```

```
    self.v = v # value
    self.au = (u/100)*v # percent/100 * value
    self.pu = u
  def set_a(self,v,u): # input uncertainty as absolute
    self.v = v
    self.au = u
    self.pu = (u/v)*100 # (absolute/value)*100

class uncertainty():
  def __init__(self):
    self.basic_functions()

  def basic_functions(self):
    self.mult = lambda x,a,y,b: np.power(float(y),float(b))*float(a
        )*np.power(float(x),float(a)-1) # d/dx (x^a * y^b)

ref = load_refprop.r410a()

frame = pd.DataFrame(
          columns = ['v','au','pu','Vdot_air', 'Q', 'COP','mdotr'],
          index=['Tret_heat','Tret_cool','Tsup_heat','Tsup_cool',
                 'phi_ret','Tevap','Tgas','Tid_liq','Iod','Qh',
                 'cp_air','rho_air','Psys','Mratio','hwe','PXV',
                 'Pgas_loss','Ifan','V'],
          data = 0)

temp1 = vu()
temp2 = vu()
temp3 = vu()

Tevap = vu()
Pevap = vu()
Wcomp = vu()
mdotr = vu()
Pgas_loss = vu()
Pgas = vu()
Tgas = vu()
Tid_liq = vu()
Pid_liq = vu()
PXV = vu()
hgas = vu()
hid_liq = vu()
Dhevap = vu()
Q = vu()
Iod = vu()
Ifan = vu()
Icomp = vu()
V = vu()
Tret_cool = vu()
Tsup_cool = vu()
Tret_heat = vu()
Tsup_heat = vu()
cp_air = vu()
Dhair_dry_cool = vu()
Dhair_dry_heat = vu()
Psat_ret = vu()
Psat_sup = vu()
phi_ret = vu()
Psys = vu()
Mratio = vu()
om_ret = vu()
om_sup = vu()
hwe = vu()
Dhair_wet = vu()
mdot_air = vu()
```

```
rho_air = vu()
Vdot_air = vu()
COP = vu()
Qh = vu()

Tevap.set_p(50,2) # F
Tgas.set_p(50,2) # F
Tid_liq.set_p(75,2) # F
Pgas_loss.set_p(5,60) # psi
PXV.set_p(250, 50) # psi
Iod.set_p(13.5,5) # amps
Ifan.set_p(0.5,50) # amps
V.set_p(240,5) # volts
phi_ret.set_p(0.7,5) # unitless
cp_air.set_p(0.24,1) # Btu/lb/F
hwe.set_p(1061,1) # Btu/lb
Mratio.set_p(0.622,1) # unitless
Psys.set_p(14.7, 2)
rho_air.set_p(0.0765,3) # lbm/ft3
Qh.set_p(10.46,5) # kW
Tret_heat.set_p(72,2) # F
Tsup_heat.set_p(100.6,2) # F
Tret_cool.set_p(72,2) # F
Tsup_cool.set_p(52,2) # F

# Evaporator Enthalpy
Pevap_Tevap = ref.dPsat_dT_func(Tevap.v)
Pevap.set_a( # psi
     ref.Psat_func(Tevap.v),
     np.linalg.norm([Pevap_Tevap*Tevap.au]) )

Pid_liq_Pevap = 1
Pid_liq_PXV = 1
Pid_liq.set_a( # psi
     Pevap.v + PXV.v,
     np.linalg.norm([Pid_liq_Pevap*Pevap.au,
                      Pid_liq_PXV*PXV.au]) )

Pgas_Pevap = 1
Pgas_Pgas_loss = -1
Pgas.set_a( # psi
    Pevap.v - Pgas_loss.v,
    np.linalg.norm([Pgas_Pevap*Pevap.au,
                      Pgas_Pgas_loss*Pgas_loss.au]) )

hid_liq_Tid_liq = ref.dHl_dTl_P_func(Tid_liq.v,Pid_liq.v)
hid_liq_Pid_liq = ref.dHl_dP_Tl_func(Tid_liq.v,Pid_liq.v)
hid_liq.set_a( # Btu/lbm
     ref.Hl_func(Tid_liq.v, Pid_liq.v)[0],
     np.linalg.norm([hid_liq_Tid_liq*Tid_liq.au,
                      hid_liq_Pid_liq*Pid_liq.au]) )

hgas_Tgas = ref.dHv_dTv_P_func(Tgas.v,Pgas.v)
hgas_Pgas = ref.dHv_dP_Tv_func(Tgas.v,Pgas.v)
hgas.set_a( # Btu/lbm
     ref.Hv_func(Tgas.v, Pgas.v)[0],
     np.linalg.norm([hgas_Tgas*Tgas.au,
                      hgas_Pgas*Pgas.au]) )

Dhevap_hgas = 1
Dhevap_hid_liq = -1
Dhevap.set_a( # Btu/lbm
     hgas.v - hid_liq.v,
     np.linalg.norm([Dhevap_hgas*hgas.au,
                      Dhevap_hid_liq*hid_liq.au]) )
```

```

# Compressor Power
Icomp_Iod = 1
Icomp_Ifan = -1
Icomp.set_a( # amps
     Iod.v - Ifan.v,
     np.linalg.norm([Icomp_Iod*Iod.au,
                     Icomp_Ifan*Ifan.au]) )

Wcomp_Icomp = V.v
Wcomp_V = Icomp.v
Wcomp.set_a( # Watts
     Icomp.v*V.v,
     np.linalg.norm([Wcomp_V*V.au,
                     Wcomp_Icomp*Icomp.au]) )
Wcomp.set_a(Wcomp.v/1000, Wcomp.au/1000) # kiloWatts
Wcomp_Icomp *= 1/1000
Wcomp_V *= 1/1000

Wcomp_Iod = Wcomp_Icomp*Icomp_Iod
Wcomp_Ifan = Wcomp_Icomp*Icomp_Ifan

# Air-side Heating
temp1_Tsup_heat = 1
temp1_Tret_heat = -1
temp1.set_a( # F
     Tsup_heat.v - Tret_heat.v,
     np.linalg.norm([temp1_Tsup_heat*Tsup_heat.au,
                     temp1_Tret_heat*Tret_heat.au]) )

Dhair_dry_heat_cp_air = temp1.v
Dhair_dry_heat_temp1 = cp_air.v
Dhair_dry_heat.set_a( # Btu/lbm
     cp_air.v*temp1.v,
     np.linalg.norm([Dhair_dry_heat_cp_air*cp_air.au,
                     Dhair_dry_heat_temp1*temp1.au]) )

mdot_air_Qh = 1/Dhair_dry_heat.v
mdot_air_Dhair_dry_heat = -Qh.v/Dhair_dry_heat.v**2
mdot_air.set_a( # (lbm/s)*(kJ/Btu)
     Qh.v/Dhair_dry_heat.v,
     np.linalg.norm([mdot_air_Qh*Qh.au,
                    mdot_air_Dhair_dry_heat*\
                    Dhair_dry_heat.au]) )
mdot_air.set_a( # lbm/hr
     mdot_air.v*3600/1.05506,
     mdot_air.au*3600/1.05506)
mdot_air_Qh *= 3600/1.05506
mdot_air_Dhair_dry_heat *= 3600/1.05506

Vdot_air_mdot_air = 1/rho_air.v
Vdot_air_rho_air = -mdot_air.v/rho_air.v**2
Vdot_air.set_a( # ft3/hr
     mdot_air.v/rho_air.v,
     np.linalg.norm([Vdot_air_mdot_air*mdot_air.au,
                     Vdot_air_rho_air*rho_air.au]) )
Vdot_air.set_a( # cfm
     Vdot_air.v/60,
     Vdot_air.au/60)
Vdot_air_rho_air *= 1/60
Vdot_air_mdot_air *= 1/60

frame.loc['rho_air','Vdot_air'] = Vdot_air_rho_air
frame.loc['Qh','Vdot_air'] = Vdot_air_mdot_air*mdot_air_Qh
frame.loc['cp_air','Vdot_air'] = Vdot_air_mdot_air* \
     mdot_air_Dhair_dry_heat*Dhair_dry_heat_cp_air
frame.loc['Tsup_heat','Vdot_air'] = Vdot_air_mdot_air* \
     mdot_air_Dhair_dry_heat*Dhair_dry_heat_temp1*temp1_Tsup_heat
frame.loc['Tret_heat','Vdot_air'] = Vdot_air_mdot_air* \
     mdot_air_Dhair_dry_heat*Dhair_dry_heat_temp1*temp1_Tret_heat

# Air-side Cooling
# Farenheit
c = [77.345+0.0057*5/9*459.67, 0.0057*5/9, -7235*9/5, 459.67,
     np.power(5/9,-8.2)]
f1 = lambda x: (c[4]*np.exp(c[0] + c[1]*x + c[2]/(x+c[3]))* \
     np.power(x+c[3],-8.2))/6894.76
df1 = lambda x: (c[4]*(c[1]-c[2]*np.power(x+c[3],-2))* \
     np.exp(c[0] + c[1]*x + c[2]/(x+c[3]))* \
     np.power(x+c[3],-8.2) - 8.2*c[4]*np.exp(c[0] + c[1]*x + \
     c[2]/(x+c[3]))*np.power(x+c[3],-9.2))/6894.76
f2 = lambda x,y: np.min([x,y])

temp1_Tret_cool = 1
temp1_Tsup_cool = -1
temp1.set_a(
     Tret_cool.v - Tsup_cool.v,
     np.linalg.norm([temp1_Tret_cool*Tret_cool.au,
                     temp1_Tsup_cool*Tsup_cool.au]) )

Dhair_dry_cool_cp_air = temp1.v
Dhair_dry_cool_temp1 = cp_air.v
Dhair_dry_cool.set_a(
     cp_air.v*temp1.v,
     np.linalg.norm([Dhair_dry_cool_cp_air*cp_air.au,
                     Dhair_dry_cool_temp1*temp1.au]) )

Psat_ret_Tret_cool = df1(Tret_cool.v)
Psat_ret.set_a(
     f1(Tret_cool.v),
     np.linalg.norm([Psat_ret_Tret_cool*Tret_cool.au ]) )

Psat_sup_Tsup_cool = df1(Tsup_cool.v)
Psat_sup.set_a(
     f1(Tsup_cool.v),
     np.linalg.norm([Psat_sup_Tsup_cool*Tsup_cool.au ]) )

temp1_phi_ret = Psat_ret.v
temp1_Psat_ret = phi_ret.v
temp1.set_a(
     phi_ret.v*Psat_ret.v,
     np.linalg.norm([temp1_Psat_ret*Psat_ret.au,
                     temp1_phi_ret*phi_ret.au]) )

temp2_Psys = 1
temp2_temp1 = -1
temp2.set_a(
     Psys.v - temp1.v,
     np.linalg.norm([temp2_Psys*Psys.au,
                     temp2_temp1*temp1.au]) )

temp3_temp1 = 1/temp2.v
temp3_temp2 = -temp1.v/temp2.v**2
temp3.set_a(
     temp1.v/temp2.v,
     np.linalg.norm([temp3_temp1*temp1.au,
                     temp3_temp2*temp2.au]) )

om_ret_Mratio = temp3.v
om_ret_temp3 = Mratio.v
```

```
om_ret.set_a(
     Mratio.v*temp3.v,
     np.linalg.norm([om_ret_Mratio*Mratio.au,
                        om_ret_temp3*temp3.au]) )

# intermdiate
om_ret_Mratio = om_ret_Mratio
om_ret_Psys = om_ret_temp3*temp3_temp2*temp2_Psys
om_ret_phi_ret = om_ret_temp3*(temp3_temp2*temp2_temp1 + \
     temp3_temp1)*temp1_phi_ret
om_ret_Tret_cool = om_ret_temp3*(temp3_temp2*temp2_temp1 + \
     temp3_temp1)*temp1_Psat_ret*Psat_ret_Tret_cool

temp1_Psys = 1
temp1_Psat_sup = -1
temp1.set_a(
     Psys.v - Psat_sup.v,
     np.linalg.norm([temp1_Psys*Psys.au,
                        temp1_Psat_sup*Psat_sup.au]) )

temp2_Psat_sup = 1/temp1.v
temp2_temp1 = -Psat_sup.v/temp1.v**2
temp2.set_a(
     Psat_sup.v/temp1.v,
     np.linalg.norm([temp2_Psat_sup*Psat_sup.au,
                        temp2_temp1*temp1.au]) )

temp3_Mratio = temp2.v
temp3_temp2 = Mratio.v
temp3.set_a(
     Mratio.v*temp2.v,
     np.linalg.norm([temp3_Mratio*Mratio.au,
                        temp3_temp2*temp2.au]) )

om_sup_temp3 = [1, 0][np.argmin([temp3.v, om_ret.v])]
om_sup_om_ret = [0, 1][np.argmin([temp3.v, om_ret.v])]
om_sup.set_a(
     np.min([temp3.v, om_ret.v]),
     [temp3.au, om_ret.au][np.argmin([temp3.v, om_ret.v])] )

#intermediate
om_sup_Mratio = om_sup_om_ret*om_ret_Mratio + \
     om_sup_temp3*temp3_Mratio
om_sup_Psys = om_sup_om_ret*om_ret_Psys + \
     om_sup_temp3*temp3_temp2*temp2_temp1*temp1_Psys
om_sup_phi_ret = om_sup_om_ret*om_ret_phi_ret
om_sup_Tret_cool = om_sup_om_ret*om_ret_Tret_cool
om_sup_Tsup_cool = om_sup_temp3*temp3_temp2*(temp2_temp1* \
     temp1_Psat_sup + temp2_Psat_sup)*Psat_sup_Tsup_cool

temp1_om_ret = 1
temp1_om_sup = -1
temp1.set_a(
     om_ret.v - om_sup.v,
     np.linalg.norm([temp1_om_ret*om_ret.au,
                        temp1_om_sup*om_sup.au]) )

temp2_hwe = temp1.v
temp2_temp1 = hwe.v
temp2.set_a(
     hwe.v*temp1.v,
     np.linalg.norm([temp2_hwe*hwe.au,
                        temp2_temp1*temp1.au]) )

Dhair_wet_Dhair_dry_cool = 1
```

```
Dhair_wet_temp2 = 1
Dhair_wet.set_a( # Btu/lbm
     Dhair_dry_cool.v + temp2.v,
     np.linalg.norm([Dhair_wet_Dhair_dry_cool*Dhair_dry_cool.au,
                       Dhair_wet_temp2*temp2.au]) )

Q_Dhair_wet = mdot_air.v
Q_mdot_air = Dhair_wet.v
Q.set_a( # Btu/hr
     Dhair_wet.v*mdot_air.v,
     np.linalg.norm([Q_Dhair_wet*Dhair_wet.au,
                       Q_mdot_air*mdot_air.au]) )

frame.loc['Qh','Q'] = Q_mdot_air*mdot_air_Qh
frame.loc['cp_air','Q'] = Q_Dhair_wet*Dhair_wet_Dhair_dry_cool* \
     Dhair_dry_cool_cp_air + Q_mdot_air*mdot_air_Dhair_dry_heat* \
     Dhair_dry_heat_cp_air
frame.loc['Tsup_cool','Q'] = Q_Dhair_wet*(Dhair_wet_temp2* \
     temp2_temp1*temp1_om_sup*om_sup_Tsup_cool +
Dhair_wet_Dhair_dry_cool*Dhair_dry_cool_temp1*temp1_Tsup_cool)
frame.loc['Tret_cool','Q'] = Q_Dhair_wet*(Dhair_wet_temp2* \
     temp2_temp1*(temp1_om_sup*om_sup_Tret_cool + temp1_om_ret* \
     om_ret_Tret_cool ) + Dhair_wet_Dhair_dry_cool* \
     Dhair_dry_cool_temp1*temp1_Tret_cool)
frame.loc['phi_ret','Q'] = Q_Dhair_wet*Dhair_wet_temp2* \
     temp2_temp1*(temp1_om_sup*om_sup_phi_ret + temp1_om_ret* \
     om_ret_phi_ret )
frame.loc['Psys','Q'] = Q_Dhair_wet*Dhair_wet_temp2*temp2_temp1* \
     (temp1_om_sup*om_sup_Psys + temp1_om_ret*om_ret_Psys )
frame.loc['Mratio','Q'] = Q_Dhair_wet*Dhair_wet_temp2* \
     temp2_temp1*(temp1_om_sup*om_sup_Mratio + \
     temp1_om_ret*om_ret_Mratio )
frame.loc['hwe','Q'] = Q_Dhair_wet*Dhair_wet_temp2*temp2_hwe
frame.loc['Tret_heat','Q'] = Q_mdot_air*mdot_air_Dhair_dry_heat* \
     Dhair_dry_heat_temp1*temp1_Tret_heat
frame.loc['Tsup_heat','Q'] = Q_mdot_air*mdot_air_Dhair_dry_heat* \
     Dhair_dry_heat_temp1*temp1_Tsup_heat

# COP
COP_Q = 1/Wcomp.v
COP_Wcomp = -Q.v/Wcomp.v**2
COP.set_a(
Q.v/Wcomp.v, # (Btu/hr)/kW
np.linalg.norm([COP_Q*Q.au, COP_Wcomp*Wcomp.au]) )
COP.set_a(COP.v/3412.142, COP.au/3412.142) # unitless
COP_Q *= 1/3412.142
COP_Wcomp *= 1/3412.142

frame.loc['V','COP'] = COP_Wcomp*Wcomp_V
frame.loc['Ifan','COP'] = COP_Wcomp*Wcomp_Icomp*Icomp_Ifan
frame.loc['Iod','COP'] = COP_Wcomp*Wcomp_Icomp*Icomp_Iod
frame.loc['Qh','COP'] = COP_Q*frame.loc['Qh','Q']
frame.loc['cp_air','COP'] = COP_Q*frame.loc['cp_air','Q']
frame.loc['Tsup_cool','COP'] = COP_Q*frame.loc['Tsup_cool','Q']
frame.loc['Tret_cool','COP'] = COP_Q*frame.loc['Tret_cool','Q']
frame.loc['phi_ret','COP'] = COP_Q*frame.loc['phi_ret','Q']
frame.loc['Psys','COP'] = COP_Q*frame.loc['Psys','Q']
frame.loc['Mratio','COP'] = COP_Q*frame.loc['Mratio','Q']
frame.loc['hwe','COP'] = COP_Q*frame.loc['hwe','Q']
frame.loc['Tret_heat','COP'] = COP_Q*frame.loc['Tret_heat','Q']
frame.loc['Tsup_heat','COP'] = COP_Q*frame.loc['Tsup_heat','Q']


# mass flow
mdotr_Q = 1/Dhevap.v
```

```
mdotr_Dhevap = -Q.v/Dhevap.v**2
mdotr.set_a( # lbm/hr
     Q.v/Dhevap.v,
     np.linalg.norm([mdotr_Q*Q.au,
                     mdotr_Dhevap*Dhevap.au]) )

mdotr_Tgas = mdotr_Dhevap*Dhevap_hgas*hgas_Tgas
mdotr_Pgas_loss = mdotr_Dhevap*Dhevap_hgas*hgas_Pgas*Pgas_Pgas_loss
mdotr_Tid_liq = mdotr_Dhevap*Dhevap_hid_liq*hid_liq_Tid_liq
mdotr_PXV = mdotr_Dhevap*Dhevap_hid_liq*hid_liq_Pid_liq*Pid_liq_PXV
mdotr_Tevap = mdotr_Dhevap*(Dhevap_hgas*hgas_Pgas*Pgas_Pevap +
     Dhevap_hid_liq*hid_liq_Pid_liq*Pid_liq_Pevap)*Pevap_Tevap

frame.loc['Qh','mdotr'] = mdotr_Q*frame.loc['Qh','Q']
frame.loc['cp_air','mdotr'] = mdotr_Q*frame.loc['cp_air','Q']
frame.loc['Tsup_cool','mdotr'] = mdotr_Q*frame.loc['Tsup_cool','Q']
frame.loc['Tret_cool','mdotr'] = mdotr_Q*frame.loc['Tret_cool','Q']
frame.loc['phi_ret','mdotr'] = mdotr_Q*frame.loc['phi_ret','Q']
frame.loc['Psys','mdotr'] = mdotr_Q*frame.loc['Psys','Q']
frame.loc['Mratio','mdotr'] = mdotr_Q*frame.loc['Mratio','Q']
frame.loc['hwe','mdotr'] = mdotr_Q*frame.loc['hwe','Q']
frame.loc['Tret_heat','mdotr'] = mdotr_Q*frame.loc['Tret_heat','Q']
frame.loc['Tsup_heat','mdotr'] = mdotr_Q*frame.loc['Tsup_heat','Q']
frame.loc['Tgas','mdotr'] = mdotr_Dhevap*Dhevap_hgas*hgas_Tgas
frame.loc['Tid_liq','mdotr'] = mdotr_Dhevap*Dhevap_hid_liq* \
     hid_liq_Tid_liq
frame.loc['Pgas_loss','mdotr'] = mdotr_Dhevap*Dhevap_hgas* \
     hgas_Pgas*Pgas_Pgas_loss
frame.loc['PXV','mdotr'] = mdotr_Dhevap*Dhevap_hid_liq* \
     hid_liq_Pid_liq*Pid_liq_PXV
frame.loc['Tevap','mdotr'] = mdotr_Dhevap*(Dhevap_hgas* \
     hgas_Pgas*Pgas_Pevap + Dhevap_hid_liq*hid_liq_Pid_liq* \
     Pid_liq_Pevap)*Pevap_Tevap


for i,r in frame.iterrows():
  frame.loc[i,'v'] = eval(i).v
  frame.loc[i,'au'] = eval(i).au
  frame.loc[i,'pu'] = eval(i).pu

for c in frame.columns.drop(['v','pu','au']):
  frame[c] = frame[c]*frame.v.abs()/np.abs(eval(c).v)


fields = {'Tid_liq':        u'{0:.0f}\u00B1{1:.0f}%',
          'PXV':            u'{0:.0f}\u00B1{1:.0f}%',
          'Pid_liq':        u'{0:.1f}\u00B1{1:.1f}%',
          'hid_liq':        u'{0:.1f}\u00B1{1:.2f}%',
          'Tevap':          u'{0:.0f}\u00B1{1:.0f}%',
          'Pevap':          u'{0:.1f}\u00B1{1:.1f}%',
          'Dhevap':         u'{0:.1f}\u00B1{1:.2f}%',
          'mdotr':          u'{0:.1f}\u00B1{1:.1f}%',
          'Qh':             u'{0:.2f}\u00B1{1:.0f}%',
          'Pgas_loss':      u'{0:.0f}\u00B1{1:.0f}%',
          'Pgas':           u'{0:.1f}\u00B1{1:.1f}%',
          'hgas':           u'{0:.1f}\u00B1{1:.2f}%',
          'Tgas':           u'{0:.0f}\u00B1{1:.0f}%',
          'rho_air':        u'{0:.3f}\u00B1{1:.0f}%',
          'mdot_air':       u'{0:.0f}\u00B1{1:.1f}%',
          'Vdot_air':       u'{0:.0f}\u00B1{1:.1f}%',
          'cp_air':         u'{0:.2f}\u00B1{1:.0f}%',
          'Dhair_dry_cool': u'{0:.2f}\u00B1{1:.1f}%',
          'Dhair_dry_heat': u'{0:.2f}\u00B1{1:.1f}%',
          'Dhair_wet':      u'{0:.2f}\u00B1{1:.1f}%',
          'Q':              u'{0:.0f}\u00B1{1:.1f}%',
          'Tsup_cool':      u'{0:.0f}\u00B1{1:.0f}%',
          'Tsup_heat':      u'{0:.0f}\u00B1{1:.0f}%',
          'Psat_sup':       u'{0:.3f}\u00B1{1:.1f}%',
          'om_sup':         u'{0:.4f}\u00B1{1:.1f}%',
          'Ifan':           u'{0:.2f}\u00B1{1:.0f}%',
          'V':              u'{0:.0f}\u00B1{1:.0f}%',
          'phi_ret':        u'{0:.2f}\u00B1{1:.0f}%',
          'Psys':           u'{0:.1f}\u00B1{1:.0f}%',
          'Mratio':         u'{0:.3f}\u00B1{1:.0f}%',
          'hwe':            u'{0:.0f}\u00B1{1:.0f}%',
          'Iod':            u'{0:.2f}\u00B1{1:.0f}%',
          'Icomp':          u'{0:.2f}\u00B1{1:.1f}%',
          'Wcomp':          u'{0:.2f}\u00B1{1:.1f}%',
          'COP':            u'{0:.2f}\u00B1{1:.1f}%',
          'Tret_cool':      u'{0:.0f}\u00B1{1:.0f}%',
          'Tret_heat':      u'{0:.0f}\u00B1{1:.0f}%',
          'Psat_ret':       u'{0:.3f}\u00B1{1:.1f}%',
          'om_ret':         u'{0:.4f}\u00B1{1:.1f}%'}
template = 'vsm2.pdf'
output = 'vsm2_filled.pdf'
pdf = pdfrw.PdfReader(template)
annots = pdf.pages[0]['/Annots']
for annot in annots:
  subtype = annot['/Subtype']
  if subtype == '/Widget':
    key = annot['/T'][1:-1]
    if key in fields:
      text = fields[key].format(eval(key).v, eval(key).pu)
      annot.update(pdfrw.PdfDict(V='{}'.format(text)))
      annot.update(pdfrw.PdfDict(AP=''))
pdfrw.PdfWriter().write(output, pdf)
```

B.3 The Maximum t-Statistic (MTS) Algorithm

This section includes elements of the code used to develop the MTS Algorithm.

B.3.1 Generating the MTS Distribution

The following script generates 20000 datasets for each of 12 different lengths. In other words, 20000 datasets are generated with length 10, 20000 are generated with length 15, 20000 are generated with length 20, and so on. The lengths include [10,15,20,30,40,50,75,100,125,150,175,200]. Each randomly dataset is randomly drawn from a Gaussian population with mean of 1 and standard deviation of 1. The MTS for each dataset is determined and stored within dataframe `B`.

```
import numpy as np
import pandas as pd

Sigma = 1
Mu = 1
NN = 20000
B = pd.DataFrame(index = np.arange(0,NN))
for N in [10,15,20,30,40,50,75,100,125,150,175,200]:
  print(N)
  for i in range(0,NN):
    b = pd.DataFrame(index = np.arange(1,N+1))
    b['x'] = Sigma*np.random.randn(N) + Mu
    b['w'] = 1

    b['sum_w_p'] = b.w.cumsum() - b.w
    b['sum_w_f'] = b.w.sum() - b.w.cumsum() + b.w
    b['wx'] = b.w*b.x
    b['sum_wx_p'] = b.wx.cumsum() - b.wx
    b['sum_wx_f'] = b.wx.sum() - b.wx.cumsum() + b.wx
    b['wx2'] = b.w*b.x**2
    b['sum_wx2_p'] = b.wx2.cumsum() - b.wx2
    b['sum_wx2_f'] = b.wx2.sum() - b.wx2.cumsum() + b.wx2

    b['xbar_p'] = b.sum_wx_p/b.sum_w_p
    b['xbar_f'] = b.sum_wx_f/b.sum_w_f
    b['var_p'] = (b.sum_wx2_p + b.xbar_p**2 * b.sum_w_p - \
        2*b.xbar_p*b.sum_wx_p)/b.sum_w_p
    b['var_f'] = (b.sum_wx2_f + b.xbar_f**2 * b.sum_w_f - \
        2*b.xbar_f*b.sum_wx_f)/b.sum_w_f
    b['t'] = np.abs(b.xbar_f - b.xbar_p) / np.sqrt(( \
        (b.sum_w_f - 1)*b.var_f + (b.sum_w_p-1)*b.var_p) \
        / (b.sum_w_f + b.sum_w_p - 2) * (1/b.sum_w_f + \
        1/b.sum_w_p))
    B.loc[i,N] = b.t.max()
    B.to_pickle('B.pkl')
```

B.3.2 Plotting the MTS Distributions

The following script plots the MTS distribution for lengths [10,20,30,150,200].

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rc

plt.close('all')
font = {'family' : 'serif',
        'weight' : 'normal',
        'serif'  : 'Times New Roman',
        'size'   : 8}
rc('font', **font)

B = pd.read_pickle('B.pkl')

fig = plt.figure(figsize=[3,2.5])
ax = fig.add_subplot(111)

for n in [10,20,30,150,200]:
  B[n].plot.density(label='n = {x}'.format(x=n))
ax.set_xlim([0,5])
ax.set_ylim([0,1])
ax.legend(loc='upper right', edgecolor='none')
ax.set_xlabel('Maximum $t$-statistic (MTS)')
fig.tight_layout()
```

B.3.3 Analyzing the Algorithm Using Monte Carlo Simulations

The following script includes code for generating 1000 datasets for each of lengths [25,50,100,200]. Each dataset as a change of 0.5σ in the center of the dataset. For each dataset, the MTS algorithm was implemented recursively and the script determines the delay between the change point and the time at which the change point was detected. If the change point was never detected, then the delay is set to -2 which indicates a Type II error. This script was run for several different magnitudes in change points in order to evaluate how the length of data and the signal-to-noise ratio affect the algorithm's success. The script is written to account for Type I errors (detecting a change when there is not one), but these do not occur because the change point is always within the time period analyzed for the parameters chosen in the script.

```
import numpy as np
import pandas as pd
from numpy import cumsum
from numpy import sum
from numpy import sqrt

Sigma = 1
Mu = 1
NNN = 1000
result = pd.DataFrame(index = np.arange(0,NNN))
for j in np.arange(0,NNN):
  for N in [25,50,100,200]:
    print(j,N)
    change = N
    NN = 2*N
    B = pd.read_pickle('B.pkl')

    thresh = np.interp(N,B.columns, B.quantile(0.999))

    df = pd.DataFrame(index = np.arange(0,NN))
    df['x'] = Sigma*np.random.randn(NN) + Mu
    idx = df.index[df.index>=change]
    df.loc[idx,'x'] = df.loc[idx,'x'] + 0.5*Sigma
    df['w'] = 1

    for i in range(N,NN):

      x = df.loc[np.arange(i-N+1,i+1),'x'].values
      w = df.loc[np.arange(i-N+1,i+1),'w'].values
      n_X = f(w)
      mean_X = f(w*x) / f(w)
      var_X = (f(w*x**2) + f(w)*mean_X**2 - 2*f(w*x)*mean_X) / f(w)
      n_Y = g(w)
      mean_Y = g(w*x) / g(w)
      var_Y = (g(w*x**2) + g(w)*mean_Y**2 - 2*g(w*x)*mean_Y) / g(w)
      sp = ((n_X - 1)*var_X + (n_Y-1)*var_Y) / (n_X + n_Y - 2)
      t = abs(mean_X - mean_Y) / sqrt(sp*(1/n_X + 1/n_Y) )
      df.loc[i,'MTS'] = np.nanmax(t)
      df.loc[i,'n'] = sum(w)
      df.loc[i,'thresh'] = np.interp(sum(w),
                                 B.columns, B.quantile(0.999))

    df.loc[df.n<10,['MTS','thresh']] = np.nan
    detect = df.loc[df.MTS>thresh].index
    if len(detect)==0:
      result.loc[j,N] = -2 # type 2 error
    else:
      detect = detect[0]
      if detect<change:
        result.loc[j,N] = -1 # type 1 error
      else:
        result.loc[j,N] = detect - change

result.to_pickle('result05.pkl')
```

B.3.4 Accounting for Multiple Change Points

The following section analyzes a randomly generated dataset with 4 change points. Each change has magnitude 1.5σ. The results are then plotted.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rc

plt.close('all')
font = {'family' : 'serif',
        'weight' : 'normal',
        'serif'  : 'Times New Roman',
        'size'   : 8}
rc('font', **font)

B = pd.read_pickle('B.pkl')

def f(x): return np.cumsum(x)
def g(x): return (np.sum(x) - np.cumsum(x))

Sigma = 1
Mu = 1
N = 200
NN = int(3.5*N)
changes = [N, 1.5*N, 2.5*N, 2.75*N]
B = pd.read_pickle('B.pkl')

s2n = 1.5*Sigma
df = pd.DataFrame(index = np.arange(0,NN))
df['x'] = Sigma*np.random.randn(NN) + Mu
for change in changes:
  idx = df.index[df.index>=change]
  df.loc[idx,'x'] = df.loc[idx,'x'] + s2n
df['w'] = 1

i0 = [0] # location of last change
for i in range(0,NN):
  print(i)
  idx = df.index[np.arange(max(i0[-1],i-N),i)]
  x = df.loc[idx,'x'] # from the last change OR
  w = df.loc[idx,'w']
  if len(x)>=10:
    n_X = f(w)
    mean_X = f(w*x) / f(w)
    var_X = (f(w*x**2) + f(w)*mean_X**2 - 2*f(w*x)*mean_X) / f(w)
    n_Y = g(w)
    mean_Y = g(w*x) / g(w)
    var_Y = (g(w*x**2) + g(w)*mean_Y**2 - 2*g(w*x)*mean_Y) / g(w)
    sp = ((n_X - 1)*var_X + (n_Y-1)*var_Y) / (n_X + n_Y - 2)
    t = abs(mean_X - mean_Y) / np.sqrt(sp*(1/n_X + 1/n_Y) )
    df.loc[i,'MTS'] = t.max()
    df.loc[i,'thresh'] = np.interp(np.sum(w), B.columns,
                                   B.quantile(0.999))
    if df.loc[i,'MTS'] >=  df.loc[i,'thresh']:
      i0.append(t.idxmax())
  if (len(i0)>1) & (i <= i0[-1]+N/2):
    idx = df.index[np.arange(max(i0[-2],i-N),i)]
    x = df.loc[idx,'x']
    w = df.loc[idx,'w']
    if len(x)>=10:
      n_X = f(w)
      mean_X = f(w*x) / f(w)
      var_X = (f(w*x**2) + f(w)*mean_X**2 - 2*f(w*x)*mean_X) / f(w)
      n_Y = g(w)
      mean_Y = g(w*x) / g(w)
      var_Y = (g(w*x**2) + g(w)*mean_Y**2 - 2*g(w*x)*mean_Y) / g(w)
      sp = ((n_X - 1)*var_X + (n_Y-1)*var_Y) / (n_X + n_Y - 2)
      t = abs(mean_X - mean_Y) / np.sqrt(sp*(1/n_X + 1/n_Y) )
      df.loc[i,'MTS_2'] = t.max()
      df.loc[i,'thresh_2'] = np.interp(np.sum(w), B.columns,
                                       B.quantile(0.999))
      i0[-1] = t.idxmax()

fig = plt.figure(figsize=[6,2.5])
ax2 = fig.add_subplot(111)
ln3 = ax2.plot(df.x,color='gray',marker='.', markersize=3,
               linestyle='none', label='Data')
ax2.set_yticks([])
ax1 = plt.twinx(ax2)
Idx = np.append(np.append(0,df.index[df.MTS>df.thresh].values),
                df.index[-1])
for i in range(0,len(Idx)-1):
  idx = df.index[(df.index>=Idx[i]) & (df.index<=Idx[i+1])]
  ln1 = ax1.plot(df.loc[idx,'MTS'],
                 label='Primary MTS calculation',
                 color='C0')
  ln2 = ax1.plot(df.loc[idx,'MTS_2'],
                 label='Secondary MTS calculation',
                 color='C1')
  ax1.plot(df.loc[idx,'thresh'],color='red', linestyle=':')

for change in changes:
  ax1.axvline(change,color='black', linestyle='--', linewidth=1)

ax1.set_ylim([0,14])
ax1.set_ylabel('Maximum $t$-Statistic (MTS)')

ax2.set_xlim([0,700])
ax2.set_xlabel('Time')

lns = ln1+ln2+ln3
labs = [l.get_label() for l in lns]
ax1.legend(lns, labs, loc='upper left', edgecolor='gray')

ax1.yaxis.tick_left()
ax1.yaxis.set_label_position('left')
ax2.yaxis.tick_right()

ax1.text(x=100, y=5.2, s='Threshold',
         horizontalalignment='left', verticalalignment='bottom',
         bbox=dict(edgecolor='None', facecolor='white', alpha=1))

ax1.annotate(s='', xy=(500,11), xytext=(450,11),
             arrowprops=dict(arrowstyle='->'))

ax1.text(x=450, y=11, s='Change Points',
         horizontalalignment='right', verticalalignment='center',
         bbox=dict(edgecolor='None', facecolor='white', alpha=1))
```

```
fig.tight_layout()
```

B.4 Predicting an Equilibrium Point

The following script includes a Monte Carlo analysis of the exponential regression accuracy and the Type II error rate.

```
import time

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rc

from sklearn.linear_model import LinearRegression

plt.close('all')
font = {'family' : 'serif',
        'weight' : 'normal',
        'serif'  : 'Times New Roman',
        'size'   : 8}
rc('font', **font)

reg1 = LinearRegression(fit_intercept=False)
def exp_regr(x,y):
  int1 = (y[1:] + y[0:-1])/2 * (x[1:] - x[0:-1])
  S = np.cumsum(int1)
  x_x1 = x[1:] - x[0]
  y1_y = y[0] - y[1:]
  reg1.fit(np.stack((x_x1,y1_y),axis=1), S)
  tau1 = reg1.coef_[1]
  yf1 = reg1.coef_[0]
  return tau1, yf1
def exp_func(x, tau, y0, yf): return yf - (yf-y0)*np.exp(-x/tau)
def r2_func(y,yhat): return 1 - \
      np.sum((y-yhat)**2)/np.sum((y-np.mean(y))**2)

Noise = np.round(np.arange(0.01, 0.155, 0.005),4)
Tau = np.round(np.arange(0.1,0.61,0.01),4)
dfy = pd.DataFrame(index = Noise, columns=Tau)
dft = pd.DataFrame(index = Noise, columns=Tau)
dff = pd.DataFrame(index = Noise, columns=Tau)
if 0:
  t0 = time.time()
  for noise in Noise:
    for tau in Tau:
      print(noise, tau)
      Y,T,F = [],[],[]
      for j in np.arange(0,1000):
        x = np.arange(0,1,0.05)
        y0 = 0
        yf = 1
        y = yf - (yf-y0)*np.exp(-x/tau)
        y = y + noise*np.random.randn(np.size(x))*(yf-y0)
        tau1, yf1 = exp_regr(x,y)
        if tau1>0:
          Y.append(np.abs(yf1 - yf)/yf)
          T.append(np.abs(tau1 - tau)/tau)
        else:
          F.append(1)
      dfy.loc[noise,tau] = np.mean(Y)
      dft.loc[noise,tau] = np.mean(T)
      dff.loc[noise,tau] = np.sum(F)
  tf = time.time()
  total_time = tf - t0
  print(total_time)
#  pd.to_pickle(dfy,'dfy.pkl')
#  pd.to_pickle(dft,'dft.pkl')
#  pd.to_pickle(dff,'dff.pkl')
else:
  dfy = pd.read_pickle('dfy.pkl')
  dft = pd.read_pickle('dft.pkl')
  dff = pd.read_pickle('dff.pkl')

## Figure 1 - Error in yf
fig = plt.figure(figsize=[3,2])
ax = fig.add_subplot(111)
errors = [0.025, 0.05, 0.075, 0.1, 0.125]
for e in errors:
  ax.plot(1/dfy.columns, dfy.loc[e,:], color='black')
ax.set_xlabel('Length of Time Series\n' \
              '(relative to time constant $\\tau$)')
ax.set_xlim([1.6, 5])
xticks = [2,3,4,5]
ax.set_xticks(xticks)
ax.set_xticklabels(['{x}$\\tau$'.format(x=round(a)) \
                    for a in xticks])

ax.set_ylabel('Absolute Error in $y_f$')
ax.set_ylim([0,0.5])
yticks = ax.get_yticks()
ax.set_yticklabels(['{x}%'.format(x=int(100*a)) for a in yticks])

ax.annotate(s='', xy=(2.75,0.25),
            xytext=(3.25,0.35),
            arrowprops=dict(arrowstyle='<-'))
ax.text(x=3.1, y=.3,
        s='Increasing Noise:\n  2%, 5%, 7.5%,\n  10%, 12.5%',
        horizontalalignment='left', verticalalignment='top',
        bbox=dict(edgecolor='None', facecolor='White', alpha=0))
fig.tight_layout()


## Figure 2 - Error in Tau
fig = plt.figure(figsize=[3,2])
ax = fig.add_subplot(111)
errors = [0.025, 0.05, 0.075, 0.1, 0.125]
for e in errors:
  ax.plot(1/dft.columns, dft.loc[e,:], color='black')
ax.set_xlabel('Length of Time Series\n' \
              '(relative to time constant $\\tau$)')
ax.set_xlim([1.6, 5])
xticks = [2,3,4,5]
ax.set_xticks(xticks)
ax.set_xticklabels(['{x}$\\tau$'.format(x=round(a)) \
                    for a in xticks])

ax.set_ylabel('Absolute Error in $\\tau$')
ax.set_ylim([0,0.8])
yticks = ax.get_yticks()
ax.set_yticklabels(['{x}%'.format(x=int(100*a)) for a in yticks])

ax.annotate(s='', xy=(2.75,0.55),
            xytext=(3.25,0.7),
            arrowprops=dict(arrowstyle='<-'))
ax.text(x=3.1, y=.6,
        s='Increasing Noise:\n  2%, 5%, 7.5%,\n  10%, 12.5%',
        horizontalalignment='left', verticalalignment='top',
        bbox=dict(edgecolor='None', facecolor='White', alpha=0))
fig.tight_layout()


## Figure 3 - Failure Rate
fig = plt.figure(figsize=[3,2])
ax = fig.add_subplot(111)
errors = [0.075, 0.1, 0.125]
for e in errors:
  ax.plot(1/dff.columns, dff.loc[e,:]/1000, color='black')
ax.set_xlabel('Length of Time Series\n' \
              '(relative to time constant $\\tau$)')
ax.set_xlim([1.6, 5])
xticks = [2,3,4,5]
ax.set_xticks(xticks)
ax.set_xticklabels(['{x}$\\tau$'.format(x=round(a)) \
                    for a in xticks])

ax.set_ylabel('Failure Rate')
ax.set_ylim([0,.1])
yticks = ax.get_yticks()
ax.set_yticklabels(['{x}%'.format(x=int(100*a)) for a in yticks])

ax.annotate(s='', xy=(2.3,0.034),
            xytext=(2.7,0.05),
            arrowprops=dict(arrowstyle='<-'))
ax.text(x=2.6, y=.04,
        s='Increasing Noise:\n  7.5%, 10%, 12.5%',
        horizontalalignment='left', verticalalignment='top',
        bbox=dict(edgecolor='None', facecolor='White', alpha=0))
fig.tight_layout()
```