7-5-2012

Monitoring and anomaly detection in solar thermal systems using adaptive resonance theory neural networks

Hongbo He

Hongbo He

Candidate

Mechanical Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Andrea A. Mammoli , Chairperson

Arsalan Razani

Peter Vorobieff

Thomas P. Caudell

Monitoring and anomaly detection in solar thermal systems using adaptive resonance theory neural networks

by

Hongbo He

B.S. Thermal and Power Engineering Xi'an Jiaotong University, 2003
M.S. Thermal and Power Engineering Xi'an Jiaotong University, 2006

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Engineering

The University of New Mexico

Albuquerque, New Mexico

May, 2012

Dedication

I would like to dedicate this dissertation to my mother Zulian Li, my wife Zhen Yang and my daughter Cassiah K. He. Also, I dedicate this to the sweet memory of my father Jianming He.

Acknowledgments

I would like to thank Dr. Andrea A. Mammoli, my advisor, and Dr. Thomas P. Caudell for their support, patience, inspiration and for introducing me to this interesting research.

I would like to thank my other dissertation committee members Dr. Arsalan Razani and Dr. Peter Vorobieff for their valuable insights and assistance in my studies. Many thanks go to David Menicucci for his advisement in the process of my research. I also wish to thank the graduate students Mario Leroy Ortiz and Donald Lincoln for their help and support. Thanks also to Mike Edgar and Jeremy Sment for help with the data acquisition system.

I gratefully acknowledge the financial support of this research project from Sandia National Laboratories.

Monitoring and anomaly detection in solar thermal systems using adaptive resonance theory neural networks

by

Hongbo He

B.S. Thermal and Power Engineering Xi'an Jiaotong University, 2003
M.S. Thermal and Power Engineering Xi'an Jiaotong University, 2006

Ph.D, Engineering, University of New Mexico, 2012

Abstract

SHW systems are generally expected to last for at least 20 years with little or no maintenance. However, in many cases failures occur far sooner due to a variety of problems, many of which are undetected or detected long after the system has failed because the backup heater silently assumes the heating load. Some of the failures may cause the system to run inefficiently or even damage other system components, such as when a system loses fluid in the solar loop and the pump runs dry, eventually destroying itself.

In recent years there has been an increasing demand for SHW systems to become economic and reliable. Fault Detection and Diagnosis (FDD) in SHW systems is an important part of maintaining proper performance, reducing power consumption and unnecessary peak electricity demand. The aim of the current work is to develop anomaly detection system that can reliably detect both anticipated and unforeseen

faults and can be implemented in commercial SHW systems without any additional sensors to the ones commonly needed for ordinary system control.

Adaptive Resonance Theory (ART)-based neural networks are chosen to perform this task, because the ART-based neural networks are fast, efficient learners and retain memory while learning new patterns. In particular, the ART networks can be incorporated into SHW system controller without any extra sensors and have the capability of an early detection of performance degradation faults. Other benefits of ART-based neural networks are on-line fault detection for its high computational efficiency and no involvement of faulty data for the training process.

A testbed for SHW system reliability is developed for the purposes of investigating the fault detection system. The input patterns of the fault detection system are generated from two sensors: collector plate temperature and water tank heat exchanger outlet temperature, which are normally installed in residential SHW systems installed by commercial operators. One of the strengths of the system is that only few data points are needed, meaning that it will not be necessary to instrument SHW systems with additional sensors, something which would not be acceptable in an aggressively competitive industry where reducing costs is paramount.

The training data for the fault detection system are generated from a verified SHW system TRNSYS (Transient Systems Simulation) model. The simulation and experimental results show that the ART-based anomaly detection has the capability to accurately and efficiently detect degradation and failure. Faults are detected at various levels depending on their severity. The ART-based anomaly detection can be used for SHW real-time reliability monitoring, as well as, eventually, in larger, more complex systems such as commercial building HVAC systems or subsystems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The market for solar hot water (SHW) systems has grown rapidly over the past five years, following tax incentives and utility rebates. Once considered to be an inconsequential generation source, the cumulative total energy production potential of these systems is beginning to capture the attention of utility operators, particularly load planners.

As shown in Figure 1.1, the residential and commercial buildings account for 42% of total U.S. energy consumption, and 18% of the residential energy consumption is from water heating (DOE and EIA, 2011). In many places, solar hot water systems can provide up to 85% of the residential water heating load. The compounded annual growth rate of total U.S. shipments of solar thermal collectors between 2000 and 2009 is about 5% (Figure 1.2). The total U.S. market value in 2009 is around 900 MWth(DOE and EIA, 2010). Due to attractive federal and state incentives, the SHW market is currently keeping growing. In 2007, sales in the U.S. only accounted for 0.5% of the solar water heating market, with the U.S. solar water heating market

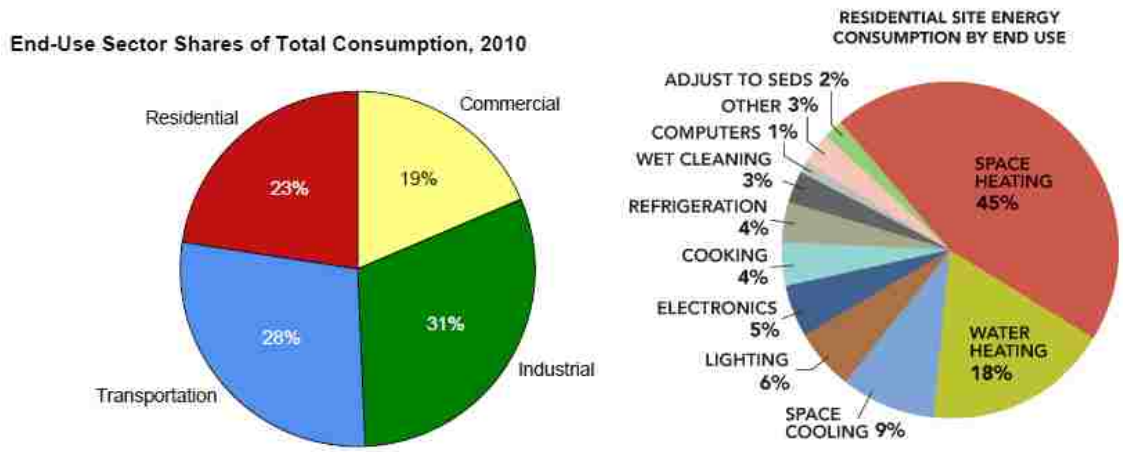

Figure 1.1: End-Use Sector Shares of Total Consumption, 2010 (left) and residential energy consumption by end use (right)

size only reaching $800 million in 2009. However, recent legislative changes provide regulatory initiatives that could create a large market in the U.S. According to a study done by Frantzis and Goffri (2010), 35% of all new U.S. solar domestic water heating systems are installed in Hawaii. China overwhelmingly led 2008 installations with 75% of global installations, but Europe is the largest solar water heating market (in terms of revenue) with nearly half of global solar water heating market value. The 2008 global solar water heating market was $12.4 billion and industry experts expect that this market will experience a second, high growth phase as demand for energy efficient methods grows.

Two concerns are emerging. First, many utilities are paying rebates to customers who install these systems based on this assumption that the systems will operate flawlessly for their expected lifetimes, typically 20-30 years. This assumption is almost certainly false because these systems typically contain a variety of mechanical components whose lifetimes are less than 20 years. Second, there are few data to define the true reliability of these fielded systems. If these systems fail in the field, the utility must be capable of supplying a corresponding amount of energy for domestic

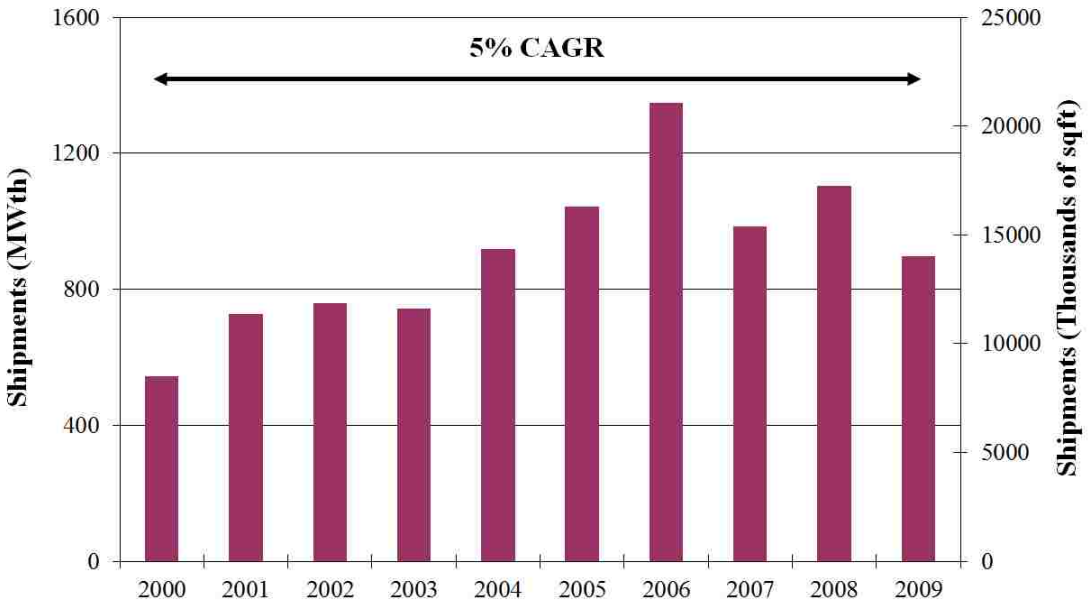

Figure 1.2: Total U.S. shipments of solar thermal collectors. Since 2000, there has been a 5% compounded annual growth rate (CAGR). Data reported in 1000's of sq.ft. MWth is calculated based upon an internationally agreed upon conversion factor of 0.7 kWth/m^2. Source: U.S. Energy Information Administration, Form EIA-63A, "Annual Solar Thermal Collector Manufacturers Survey".

water heating. This is because one of the basic problems with SHW systems is that when failures occur in installed systems, there are few negative consequences because the backup water heating system silently assumes the load. Unless system owners are regularly monitoring their systems, they do not notice when the solar components are offline due to failures. Most SHW controllers have no capability to recognize a failure in the system or to notify the owner that a problem exists.

Thus, new tools are needed to detect failures in systems. Even more desirable would be tools and techniques that could predict impending failures. If these predictive capabilities could be integrated into existing SHW controllers at low additional cost, the unscheduled down time that systems experience could be substantially reduced. New tools also can be applied to other distributed energy resources of similar

nature, and other systems where reliability and performance is of concern. In this study, the first objective is to develop and validate SHW simulation model. This model is used to collect normal operation data. The second objective is to build a testbed to validate the SHW simulation model. The third object is to develop a practical fault detection technique for SHW system that:

1. can be developed based only on simulation fault-free data or historical fault-free measurement data, and no faulty training data are required;
2. does not require the installation of extra sensors;
3. is robust under different operation conditions;
4. has high computational efficiency and can be implemented in on-line system monitoring;
5. has the capability to detect complete or degradation faults.

1.2 Literature review on fault detection and diagnosis

A fault detection and diagnosis system (FDD) consists of the following three tasks:

1. Fault detection: detection of the fault occurrence.
2. Fault isolation: localization of the fault, and classification of the fault.
3. Fault identification: determination of the size and cause of the fault.

Research on FDD has been active over several decades. Methods for FDD as part of the control process are receiving an increased interest due to increasing demands on

reliability and safety of various types of plants, such as HVAC systems, SHW systems, FDD in these systems are an important part of maintaining proper performance, reducing power consumption, unnecessary peak electricity demand, reducing wear on various equipment and the CO_2 emissions. FDD also plays an important role to meet the stringent safety requirements in some disciplines, such as nuclear engineering, chemical engineering, and aerospace engineering.

Methods of FDD can be broadly classified into two major categories: model based methods and process history based methods. Model based methods use various kinds of models to diagnose the cause of failures. Model based methods can be divided into qualitative model based methods and quantitative model based methods. In quantitative models the relationships between the inputs and outputs are expressed by mathematical functions. In qualitative models these relations are expressed by qualitative terms. For example, a qualitative model predicts whether the collector outlet temperature increases or decreases with increasing the flow rate. Process history based methods are derived from historical process data. A detailed classification of FDD methods is shown in Figure 1.3 (Venkatasubramanian et al., 2003a,b,c).

Today, FDD methods are fully integrated into chemical plants, aerospace applications, HVAC systems, heat pumps, chillers, air handling units (AHU) system. In this section, a review of prior work in the area of FDD methods in energy applications with quantitative model-based methods, qualitative model-based models and process history based methods is presented.

1.2.1 Quantitative model-based methods in energy applications

Quantitative model-based methods for fault detection and diagnosis are based on a detailed model of the underlying physical process. The model compares the outputs

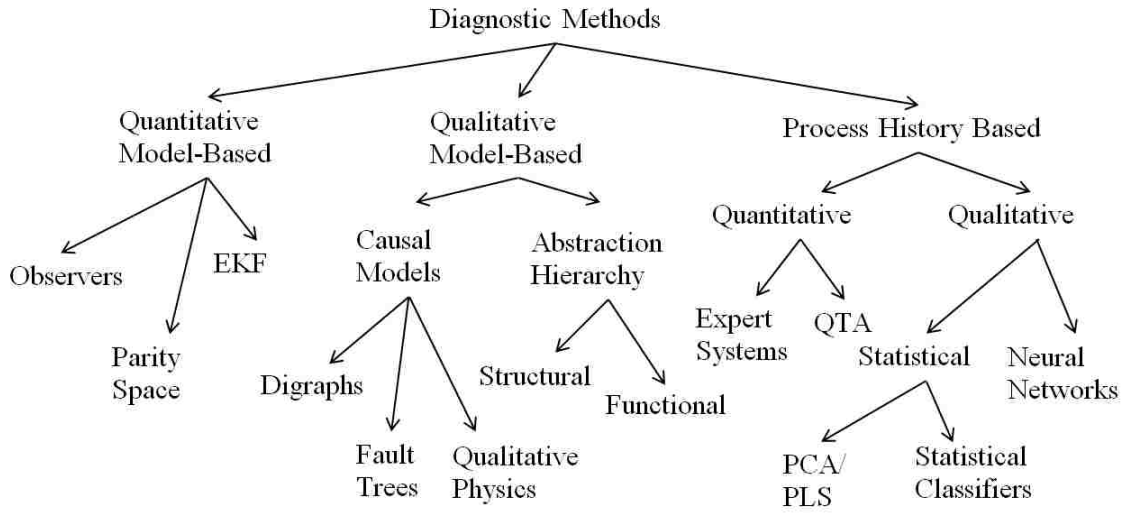

Figure 1.3: Classification of diagnostic algorithms (Venkatasubramanian et al., 2003a)

of the real system with the expected value to detect and diagnose faults. The study of quantitative model-based methods began in the early 1970s. Important reviews in the field of quantitative model-based methods include those by Chow and Willsky (1984), Isermann (1984), Basseville (1988), Gertler and Singer (1990), Frank (1990), Isermann (1997), Frank et al. (2000), Isermann (2005).

Using the dependencies between different measurable signals, faults can be detected in the processes, actuators and sensors. These dependencies are expressed by explicit mathematical functions. The basic structure of the model-based FDD method is shown in Figure 1.4. After collecting input signals $\mathbf{U}$, output signals $\mathbf{Y}$ and non-directly measurable variables $\mathbf{N}$, the detection methods create features including residuals $\mathbf{r}$, parameter estimates $\widehat{\mathbf{\Theta}}$ or state estimates $\widehat{\boldsymbol{x}}$. Then they analyze symptoms by comparing these features with expected features.

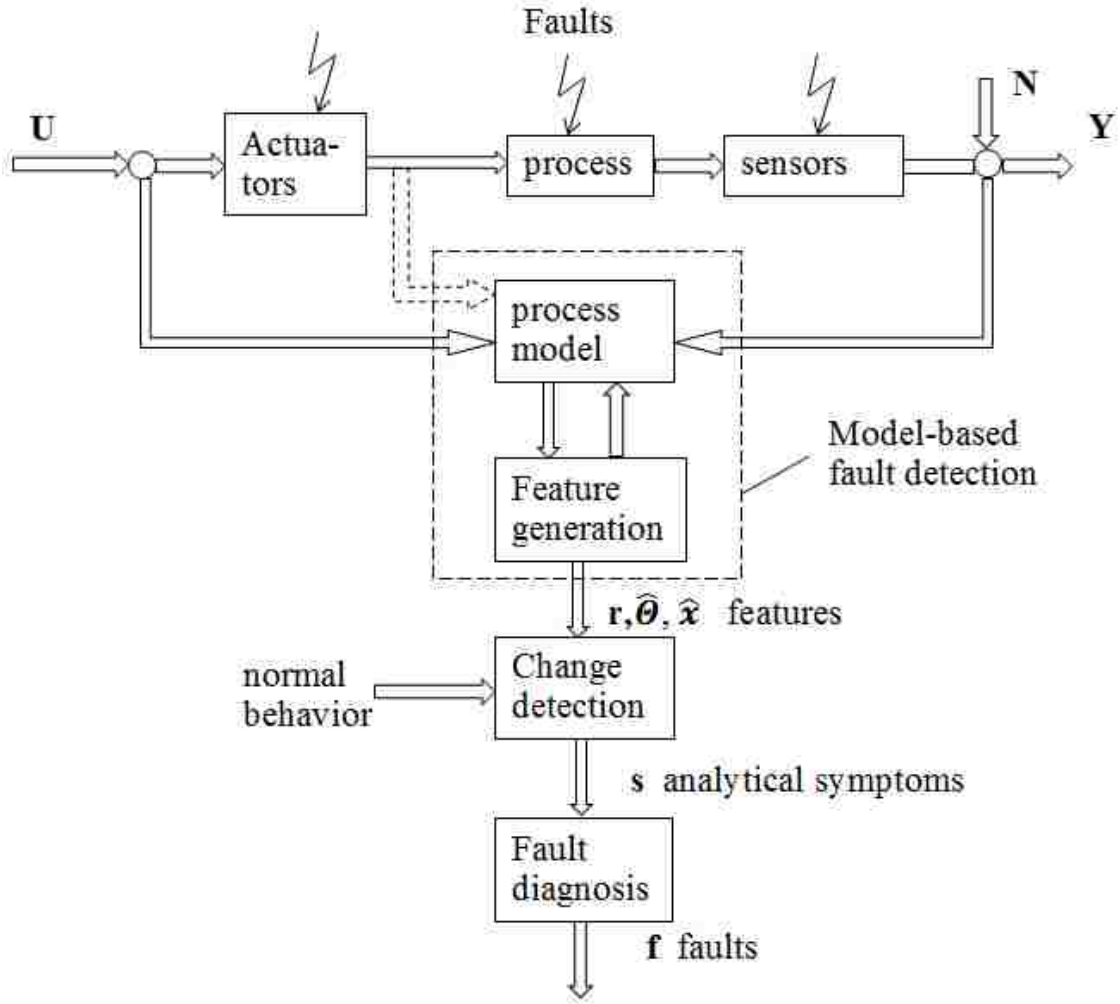

Figure 1.4: General scheme of model-based FDD method (Isermann, 2005)

Some quantitative model-based methods were applied in fields as diverse as of digital aircraft control and electrocardiography (Willsky, 1976). Clark (1978a) developed a simplified observer scheme which used only one observer driven by a single sensor output. Then other outputs were regenerated and compared with estimated outputs. The scheme was able to detect single sensor faults and verified by some tests with a jetfoil boat control system. Clark (1978b) also developed a dedicated observer scheme which used several observers for single sensor outputs. The multiple sensor faults could be detected by comparing the measured outputs with the esti-

mated outputs. The robustness of the dedicated observer scheme was demonstrated with respect to variations in two important physical parameters of a jetfoil boat control system.

FDD methods for heating, ventilation, air conditioning and refrigeration (HVAC &R) systems were first developed and applied in the late 1980s. Stallard (1989) and McKellar (1987) developed model-based FDD for household refrigerators. Wagner and Shoureshi (1992) presented two schemes: the limit and trend checking scheme, and innovation-based failure diagnostic strategy, and applied these to a residential heat pump system. The innovation-based fault detection scheme used a fourth-order nonlinear compressor and condenser model, including a variable structure system observer. The experimental results demonstrated that the fault diagnostics for the thermofluid systems could detect five heat pump major failures which included condenser fan motor failure, evaporator fan motor failure, capillary tube blockage, compressor piston leakage, and sealed system leakage.

Norford et al. (2002) developed two methods for variable-air-volume (VAV) air-handling-units (AHUs). One method was a first-principles-based model of system components, the other was based on semi-empirical correlations of sub-metered electrical power with flow rates or process control. First-principles-based models are considered as quantitative model types. Both methods detected nearly all of the faults which were in the air-mixing, filter-coil, and fan sections of the AHU. Compared to the first-principles-based method, the electrical power correlation method diagnosed more faults successfully. The first-principles-based models required many sensors and were sensitive to the occurrence of non-ideal system behavior. Gray-box (based on partial understanding of the system) electrical power models were based on data collected from normal system operation under closed loop control, which makes the models less sensitive to non-ideal system behavior and less likely to generate false alarms. Therefore, gray-box modeling methods are more robust than

first-principles-based methods Norford et al. (2002).

Bendapudi and Braun (2002) designed a dynamic centrifugal chiller FDD model. This model can be applied to other chillers, but the detailed physical characteristics of the system and subsystem are difficult to obtain, since other chillers should use similar controller and compressor. As pointed out by Bendapudi and Braun, first-principles-based models were not widely used probably due to the difficulty in creating the models and the complexity of implementing the models in real-time fault diagnostic systems.

Thumati et al. (2011) built an on-line model-based fault detection and isolation (FDI) scheme for HVAC systems. This was the first reported FDI scheme for HVAC systems which had the capability of online fault diagnostic. An observer consists of an on-line approximator in discrete-time (OLAD). The faults were detected by comparing the observer outputs and the HVAC system states. A simulation example demonstrated that this FDI scheme had the capability to detect expected faults including cooling coil degradation, leakage in thermal space or insulation degradation.

As proposed by these previous researchers, quantitative models are based on a detailed model derived from a theoretical analysis of the physical processes. Many sensors are needed to implement a system which can compare the model to measured values. These models are expensive to generate and maintain. As a result, the quantitative models are not widely used.

1.2.2 Qualitative model-based methods in energy applications

Qualitative model-based methods use qualitative modeling techniques to detect and diagnose faults. Qualitative model-based methods can be further divided into causal

models and abstraction hierarchies. The causal models are divided into three categories: signed digraphs (SDG), fault trees, and qualitative physics. The abstraction hierarchies include those based on functional abstraction hierarchy models as well as those based on functional abstraction hierarchy models. The bond graphs, case-based reasoning and rule-based systems are also considered as qualitative model types.

The signed digraphs (SDG) use graphical models to capture the root cause. A SDG is usually built by nodes and directed arcs, in which the nodes indicate state variables, failure origins and alarm conditions, and directed arcs show the relationship between these nodes. Shiozaki and Miyasaka (1999) developed a VAV AHU fault diagnosis tool using SDG. A real-time fault diagnosis system could be created using this tool. Maurya et al. (2003a,b) presented SDG methods which were used for safety analysis and fault diagnosis in chemical process systems.

Fault tree analysis (FTA) converts the physical system into a structured logic diagram in which the event symbols and logic symbols are used. FTA includes the following four steps: system definition, fault-tree construction, qualitative evaluation, quantitative evaluation (Fussell et al., 1974). Hessian et al. (1990) used FTA to diagnose faults for an existing control-room HVAC system. This logic-based methodology was incorporated into the operating system design to improve system reliability.

Rule-based systems consist of a set of in-then-else rules which contains all of the appropriate knowledge, a temporary working memory, and an inference engine which examines all the rule conditions to draw conclusions. Luskay et al. (2003) used rule-based models to diagnose the operating conditions of an AHU. However this kind of models can be difficult to use for real-time diagnosis. For example, in order to examine the outdoor-air temperature sensor, the outdoor-air damper was required to close completely, then when the system reached steady-state, compared the return-air temperature and the average mixed-air temperature. If the return-air temperature was equal to the average mixed-air temperature, then the outdoor-air

temperature sensor was classified as faulty.

House et al. (2001) presented an air-handling unit performance assessment rules (APAR) for fault detection in AHU. The APAR rules were based on qualitative models. The simulation results showed that a stuck outdoor air damper and a manual override of the control signal to the mixing box dampers were detected by the APAR. Furthermore, the APAR was used by Schein et al. (2006) for fault detection in AHUs. The expert rules in the APAR were derived from mass and energy balances. The simulation and experimental results demonstrated that the APAR could detect a variety of sensor, actuator, and control logic faults. It was possible to embed APAR in commercial building AHU controllers. Schein and Bushby (2006) built a system-level hierarchical FDD tool for HVAC systems. The advantage of this hierarchical FDD tool was that it has the capability to infer a higher-level fault from multiple lower-level fault reports. Other similar rule-based models for AHU and HVAC systems were described by Katipamula et al. (1999, 2003).

Compared to quantitative models, qualitative models are easy to develop and apply, and can identify faults because the models are based on cause-effect relationships. However, qualitative models are specific to a system and it is difficult to develop a complete set of rules for complex systems, and as a consequence, qualitative models are not widely used in complex systems.

1.2.3 Process history based methods in energy applications

Process history based methods are based on a large amount of historical process data and use *a priori* knowledge to extract features. According to the extraction process, the methods are divided into quantitative and qualitative methods. Methods that extract qualitative information include expert systems and qualitative trend analysis (QTA). Quantitative methods include neural networks and statistically de-

rived models. Statistical models include those based on principal component analysis (PCA)/partial least squares (PLS), and statistical classifiers.

Expert systems are computer programs which use a specialist's knowledge to solve specialized problems in a narrow field. Expert systems have been widely used for fault diagnosis in many fields. Early papers about applications of expert systems to fault diagnosis can be found in Nelson (1982); Kumamoto et al. (1984); PAU (1986); Cardozo and Talukdar (1988). Kilma (1990) developed an expert system for operational problems trouble-shooting in solar hot water systems. Based on an expert system, Kaldorf and Gruber (2002) built a diagnostic tool-performance audit tool (PAT) to supervise building performance. The faults, underperformance or abnormalities could be automatically detected by the PAT. Tassou and Grace (2005) described a fault diagnosis system based on an expert system and fuzzy Jave toolkit (a set of Jave classes which deal with fuzzy concepts and reasoning) for vapour compression refrigeration systems. The fault diagnosis system was able to detect gradual leakage and over charge conditions. Venkatasubramanian et al. (2003c) described the application of expert systems in process industries while Patel and Kamrani (1996) summarized the application of expert systems for diagnosis and maintenance.

PCA, an effective tool in multivariate data analysis, reduces the original set of correlated parameters to a reduced set of uncorrelated parameters (i.e. PCs). PCA has been widely used in the FDD field. Wang and Xiao (2004) described a FDD strategy based on PCA to detect degradation sensor faults in typical AHU. The PCA models were based on heat balance and pressure-flow balance. Li (2009) developed a automated fault detection and diagnosis (AFDD) methods based on PCA for AHU. He applied the two PCA methods which included wavelet-PCA and pattern matching-PCA to the HVAC system of the Iowa Energy Center Energy Resource Station.

Neural networks have been broadly used in many areas including fault diagnosis.

The most popular neural networks are the back-propagation neural networks and multi-layer perceptron neural networks (described in more detail later). Lee et al. (1996) described a fault diagnosis method for AHU using back-propagation neural networks. The training data include normal and faulty data collected from steady-state operating conditions. The method consisted of two steps. First, the value of the cooling coil valve control signal was predicted by back-propagation neural networks. Then seven residual symptoms were used to define eight kinds of faults by a type of if-then reasoning. This method was verified by experimental data and identified each fault. In this method all the faults detected, such as the failure of the supply fan or the return fan, were severe problems, but the system could not detect performance degradation at the early stages. This method was used for steady state and could not applied for transient state. In addition, if a new fault was added to the system, it is necessary to retrain the system by back-propagation and possibly to add new residual symptoms.

Lee et al. (1997) developed a fault diagnosis system for AHUs using a two-stage artificial neural network (ANN). The operation of the AHU was divided into various subcategories: normal, pressure control subsystem fault, flow control subsystem fault, cooling coil subsystem fault, mixing box damper subsystem, and an unknown type of fault. A fault occurring in a subsystem was classified by the stage one ANN, then the cause of the fault was identified by the stage two ANN. The system was able to identify eleven faults. If the failure of the supply air temperature sensor was detected, a regression equation was used to recover an estimate for the supply air temperature, then the cooling coil valve controller used this information to set the supply air temperature to the setpoint value. This method was based on steady state conditions and demonstrated by a simulation model of a laboratory-scale AHU.

Li et al. (1996) presented a fault diagnosis method for a heating system using two multi-layered feed-forward neural networks with a tan-sigmoid transfer function.

This method was designed to diagnose seven failures. The first ANN was used to discriminate the failure of heating curve (or an outside temperature reset, which controls the supply water temperature) from the other failures and the second ANN was able to classify the other six failures. Both ANNs were trained using an improved back-propagation algorithm. Simulation results demonstrated that this method was able to diagnose seven failures, with the exception that the first ANN could not discriminate the heating curve from the other failures. Li et al. (1997) also built another neural network prototype for fault detection and diagnosis of heating systems, in which ANN was also trained with an improved back-propagation algorithm. The training data were collected from a simulation model of a reference heating system. Then the prototype was tested on four other heating systems. The results demonstrated that the ANN has the capability to diagnose faults in heating systems.

Morisot and Marchio (1999) built a fault detection model for a variable air volume (VAV) system using multi-layer perceptron neural networks. The training data were collected from a dimensionless physical model which was specific for that VAV system. The simulation results demonstrated that this model could detect two kinds of faults: fouling and sensor deviation.

Lee et al. (2004) explored a subsystem level FDD scheme for AHU based on general regression neural networks (GRNN). The scheme had three steps: process estimation, residual generation and fault detection and diagnosis. No mathematical models were used to estimate the system and GRNN had a high computation efficiency. This method was suitable for real-time FDD. The simulation results demonstrated that the GRNN was able to diagnose several abrupt and performance degradation faults.

Du and Er (2004a,b) built a fault diagnosis method for AHU using dynamic fuzzy [1] neural networks (DFNNs). DFNN was based on the ellipsoidal basis function

[1] or fuzzy logic, which deals with reasoning that is approximate rather than fixed and

(EBF) neural network and had four layers: input pattern in layer 1, membership function in layer 2, possible IF-part for fuzzy rules in layer 3, and the THEN-part for fuzzy rules in layer 4. The normal and faulty data were collected form an AHU simulation model by MATLAB. The inputs were the residuals calculated at steady-state conditions. The simulation results demonstrated that the training and diagnosis processes were fast and the diagnosis rate was high.

Kalogirou et al. (2008) developed a fault diagnosis system for SHW systems based on multi-layer perceptron neural networks. This fault diagnosis system could detect two types of faults: collector faults and faults in insulation of the pipes. The training data for this system required many sensors which are not typically installed in residential SHW systems, such as a global radiation sensor, incidence angle sensor, wind speed sensor. Moreover, these models have four limitations. First, most models can not be used to extrapolate beyond the range of the training data. Second, a large amount of training data are needed to represent both normal and faulty operation. Third, the generalization of these models is very limited as they are specific to a system or process. Fourth, extra sensors must be installed to collect training and testing data.

Du et al. (2010) described a FDD strategy based on an Efficient Adaptive Fuzzy Neural Network (EAFNN) method for an AHU. The structure of EAFNN was based on the EBF which mentioned by Du and Er (2004a,b). The main advantage of the EAFNN was that the unneeded hidden units were removed by Error Reduction Ratio (ERR) method and a modified Optimal Brain Surgeon (OBS) technology. Comparing with other earlier works, such as back-propagation (BP) based method, radial basis function (RBF) based method and DFNN, the EAFNN had much lower root mean square error (RMSE) than other methods. Simulation results demonstrated that this method resulted in fast and efficient learning.

---

exact

Zhu et al. (2012) designed a FDD method for sensors in an AHU based on neural network pre-processed by wavelet and fractal (NNPWF). The normal, faulty and transitional operation conditions of the sensors data from the building automation systems were collected for the training. The data were decomposed by three-level wavelet analysis and the fractal dimensions information were extracted for the inputs of a three-layer neural networks. Normal, faulty and transition operation conditions could be identified by comparing the prediction with the objective vectors. Performance test results showed that the diagnosis efficiency of the NNPWF was high and this method could be used to diagnose other faults in HVAC systems.

1.3 Summary

Methods of fault detection and diagnostics (FDD) can be broadly classified into three major categories: quantitative model-based methods, qualitative model-based methods, and process history based methods (Venkatasubramanian et al., 2003a,b,c). The quantitative models are based on a detailed model derived from a theoretical analysis of the physical processes. Many sensors are needed to implement a system which can compare the model to measured values. These models are expensive to generate and maintain. Comparing to quantitative models, qualitative models are easy to develop and apply, and can identify the faults because the models are based on cause-effect relationships. However, qualitative models are specific to a system and it is difficult to develop a complete set of rules for complex systems. Various quantitative and qualitative model-based methods for FDD of building systems have been presented in the literature.

Process history based methods are based on a large amount of historical process data and use a priori knowledge to extract features. Neural networks are an important class of process history based methods. Neural networks are able to detect

and identify faults with high computation efficiency and can be embedded in SHW operation and control systems.

For the case of neural network-based FDD methods, it is important to note that in most of the cases described neural networks are trained to learn the steady-state relationship between the dominant symptoms and the faults, and fault diagnosis is based on the symptoms and dominant residuals. Only faults with the corresponding dominant symptoms can be diagnosed, but faults with new symptoms can not be detected. These methods are used for steady state and cannot applied for transient state. In addition, faults with dominant symptoms are severe abrupt failures or complete faults, and the methods are not effective for performance degradation faults. In many cases, it was necessary to install many extra sensors which are not normally installed in commercial SHW systems, such as a global radiation sensor, incidence angle sensor, wind speed sensor. The generalization of these models is generally very limited as it is specific to a system.

SHW systems are generally expected to last for at least 20 years with little or no maintenance. However, in many cases failures occur far sooner due to a variety of problems, many of which are undetected or detected long after the system has failed because the backup heater silently assumes the heating load. Some of the failures may cause the system to run inefficiently or even damage other system components, such as when a system loses fluid in the solar loop and the pump runs dry, eventually destroying itself. In recent years there has been an observed increasing demand for SHW systems to become economic and reliable. FDD in SHW systems is an important part of maintaining proper performance, reducing power consumption and unnecessary peak power demand. The aim of the current work is to built a FDD method for SHW systems which can be implemented in commercial SHW systems without any extra instruments and is able to detect the performance degradation faults at the early stage. ART-based neural networks are chosen, because the ART-

based neural networks can be incorporated into SHW systems without any extra sensors and have the capability of an early detection of performance degradation faults. Other benefits of ART-based neural networks are the possibility of on-line fault detection allowed by high computational efficiency, and the ability to detect fault conditions without exposure of "faulty" training data.

Chapter 2

Neural networks

Neural network consists of neurons, simulate human brain structures and information processing (Haykin, 1998). The topic of neural network is a interdisciplinary subject with application in artificial intelligence, pattern recognition, automatic control, cognitive science, neuroscience, nonlinear dynamical systems and meteorology, among many others.

2.1 Biological neuron model

There are about 100 billion neurons in the brain. Neurons are cells that process and transmit information by electro-chemical signals. As shown in Figure 2.1, a typical neuron consists of soma or cell body, axon, dendrites and synapse. The dendrites are tree-like receptive networks of nerve fibers that receive activation from other neurons to the cell body. The cell body processes and thresholds the incoming activation and convert it into output activation. The output activation was then carried out by the axon, a single long fiber, to other neurons. The cleft between an axon of one cell and a dendrite of another cell is called a synapse. The signal transmission

through a synapse is by diffusion of chemicals called neuro-transmitters. The function of a neural network is jointly determined by the neurons and the strengths of the synapses. An artificial neuron is a simple model of the biological neuron by a simplified mathematical function, as shown in Figure 2.2.

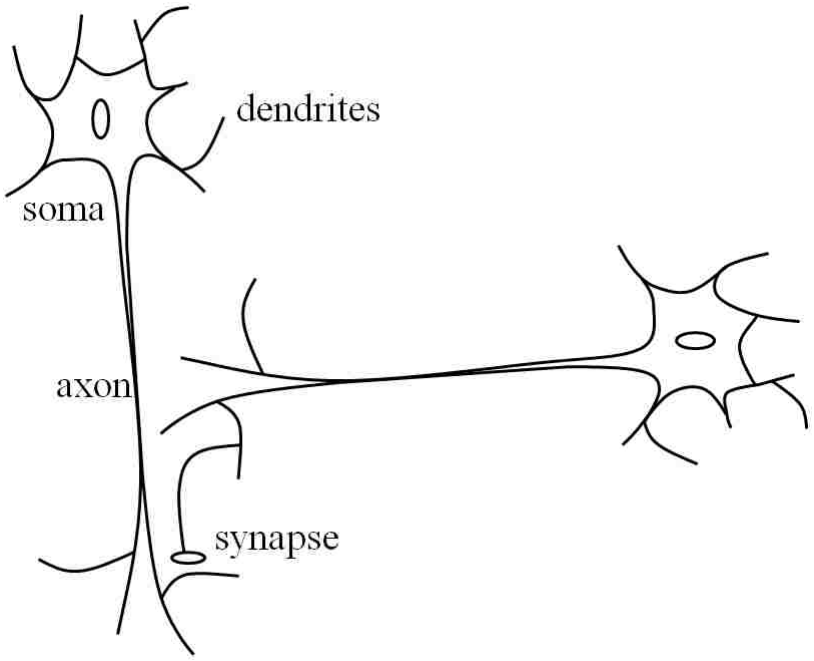

Figure 2.1: Schematic of biological neurons

2.2 The history of neural networks

2.2.1 The stage of formation

During the formative stage, from 1943 $\sim$ 1969, a number of researchers developed many neural network models and learning rules. The McCulloch-Pitts neuron model was developed by the psychiatrist McCulloch and mathematician Pitts in 1943, uncovering the prelude of correlatively international research about neural networks

(McCulloch and Pitts, 1943). The structural of McCulloch-Pitts model is described in section 2.3. In 1949 the Hebbian learning rule was proposed by neuropsychologist Hebb, which states that the synapse between tow neurons should be strengthened if the two neurons are active across the synapse (Hebb, 1949). In 1958 Rosenblatt proposed the perceptron, which is the first model for classification of patterns, but can only be applied in a convergent way if linearly separably (Rosenblatt, 1958). In 1960 Widrow and Hoff developed the Adeline neural network which is a adaptive linear neuron network. Adeline neural networks adjust their weights during the training process until the output is expected for the given input (Widrow and Hoff, 1960).

2.2.2 The stage of dormancy

In 1969 Minsky and Papert showed that there were many fundamental problems with perceptron-like networks. A perceptron is the simplest type of neural network and is also thought of as a binary classifier. For example, simple perceptrons can only solve linearly separable problem, and cannot learn the logical function of exclusive-or (XOR) (Minsky and Papert, 1969). These studies lead to the conclusion that although perceptrons were interesting to study, they were ultimitely a 'sterile' direction of research. Influenced by Minsky and Papert, in combination with computer hardware limitations, many researchers left this field.

Nevertheless, some researchers did some important work during the 1970s. In 1972 Kohonen developed a correlation matrix model for associative memory. The model was trained by Hebb rules and was able to learn an association between a set of inputs and outputs. The correlation matrix memories was able to classify data (Kohonen, 1972). At the same time, Anderson developed a linear associator model for associative memory, which was similar as Kohonen's mode (Anderson, 1972). Grossberg started to work on self-organizing networks (Grossberg, 1976a,b).

2.2.3 The stage of resurgence

Neural networks re-emerged in the early 1980s. In 1982 the Hopfield network was proposed by John Hopfield, which significantly contributed to the new era of neural networks (Hopfield, 1982). A Hopfield network is a type of recurrent network, which could solve complex computational problem by using an energy function. The now popular backward propagation of error learning method was first introduced by Bryson and Ho (1969). In 1986 the back-propagation algorithm was developed (Rumelhart et al., 1986). In the same year, summaries of the back-propagation algorithm and multilayer perceptrons were given by Rumelhart and McClelland (Rumelhart and McClelland, 1986a,b). Other researchers also developed new neural network models, such as adaptive resonance theory (ART). This neural architecture was developed by Grossberg and Carpenter (Carpenter and Grossberg, 1987; Carpenter et al., 1991b). The self-organizing map (SOM) was created by Kohonen (Kohonen, 1990), and the Boltzmann machine, a type of stochastic recurrent neural network, was invented by Hinton and Sejnowski (Ackley et al., 1985).

Two new concepts were primarily responsible for the resurgence of interest in neural networks. The first was the Hopfield network which is a recurrent network with symmetric synaptic connections. Hopfield explained the operation of this model with the use of statistical physics. This paved the way for future research about realistic models for neurobiological systems. The other important development was the back-propagation algorithm which was used to train multilayer perceptron networks. The back-propagation algorithm was the answer to the earlier critique of perceptrons (Minsky and Papert, 1969). These new concepts and the availability of powerful new computers promoted the innovation and development of the field of neural networks, and neural networks have since been applied in many disciplines.

2.3 Typical neural network model and ART based neural networks

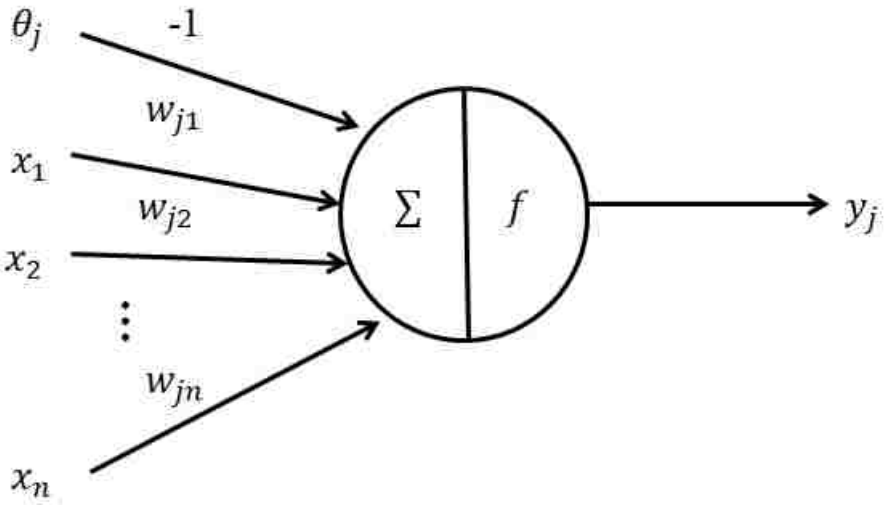

Figure 2.2: McCulloch-Pitts model

Neural network consists of neuron models. A typical McCulloch-Pitts model (McCulloch and Pitts, 1943) is shown in Figure 2.2. The j-th neuron has many inputs x_i. The inputs x_1,x_2,..., x_n are weighted by the weights w_{1j},w_{2j},..., w_{nj} respectively. The j-th neuron net input N_j is the summation of the weighted inputs:

$$N_j = \sum_i w_{ji} x_i. \tag{2.1}$$

Letting y_j be the j-th neuron output, the McCulloch-Pitts model is written as:

$$NET_j = \sum_i w_{ij} x_i - \theta_j \tag{2.2}$$

$$y_j = f(NET_j), \tag{2.3}$$

where the θ_j is the bias of j-th neuron, NET_j is the net input into the j-th neuron and the activation function f is a hard threshold defined by signum function. As shown in Figure 2.3, if the net input is larger than the bias, then the output y_j is 1, otherwise the output is -1.

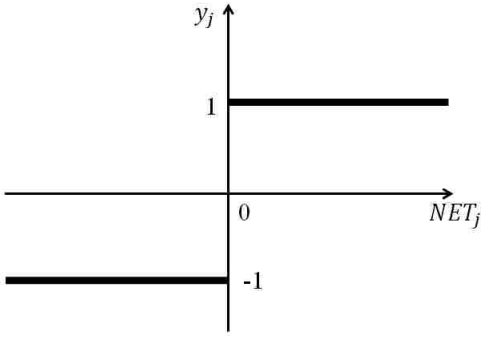

Figure 2.3: Signum function

Consideration of time delay (between input and output), leads to an output of the McCulloch-Pitts model which can be expressed as:

$$y_j(t+1) = f(\sum_i w_{ij}x_i(t) - \theta_j), \tag{2.4}$$

where t is the time step.

Multi-layer perceptrons consist of McCulloch-Pitts neurons, in which the activation functions are usually nonlinear functions, such as the Log-Sigmoid function shown in Figure 2.4.

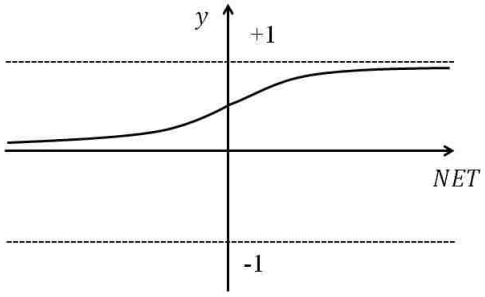

Figure 2.4: Log-Sigmoid function

The output range of the Log-Sigmoid function is from 0 to 1, and the Log-Sigmoid function is defined as:

$$y = \frac{1}{1 + e^{-NET}}. \tag{2.5}$$

The McCulloch-Pitts model is a single layer perceptron and can be trained for classification. But it can only solve linearly separable problem. The back-propagation algorithm is the most popular algorithm for the supervised training of multilayer perceptrons. It can be used for nonlinear classification problems. But the training process is time consuming, and risks being trapped in a local minimum. Once trained, the back-propagation network can't recognize new patterns appearing in a problem, or will give wrong predictions. In order to add the new patterns into the network, the network has to be retrained. The generalization ability of back-propagation network is limited.

2.3.1 ART1 neural network models

The objective of this research is to use adaptive resonance theory (ART) networks to build a fault detection system. ART networks are a type of ANN with many advantages compared to more commonly used neural network systems (such as multilayer perceptrons) that were used to monitor and diagnose SHW systems. ART is a well-established self-organizing neural technique for categorizing input patterns, characterized by rapid, stable learning and high computational efficiency. Several types of networks belong to the ART class. Here, the simplest of these, the ART1 architecture, is described briefly. The network has three layers, as depicted in Figure 2.5: input layer F0, the comparison layer F1 and the category layer F2. The input vectors are normalized and use complement coding which can represent both the presence and absence of features (Carpenter et al., 1991a). There are n neurons $(u_i, i = 1, 2, \ldots, n)$ in layer F1. Each of these has three inputs: one is from layer F0

(representing bottom-up sensory information), one is the feedback signal from the recognition layer F2 (representing top-down category expectation), while the third is a signal from the gain control G_i. Each neuron in layer F1 has binary output c_i defined by:

$$c_i = c_i(x_i, t_i, G_i) = \begin{cases} 1, \text{at least two inputs are 1} \\ 0, \text{otherwise} \end{cases}, \tag{2.6}$$

where x_i is the i-th component of the bottom-up weights, and t_i is the i-th component of the top-down weights.

There are m neurons $(u_j, j = 1, 2, \ldots, m)$ in the recognition layer F2. Each neuron in layer F1 is connected to all neurons in layer F2 through a bottom-up weight matrix $B = (b_{ji})_{n\times m}$, where b_{ji} represents the weight to neuron u_j in layer F2 from neuron u_i in layer F1. Conversely, the analog output of each neuron in layer F2 is connected to all neurons in layer F1 through a top-down weight matrix $T = (t_{ij})_{m\times n}$, where t_{ij} represents the weight from neuron u_j in layer F2 to neuron u_i in layer F1. The matrices B and T both perform long-term memory (LTM) functions.

The input to the j-th neuron in layer F2 from layer F1 is given by

$$\mu_j = x^T B_j = \sum_{i=1}^{n} b_{ji} u_i, \tag{2.7}$$

where B_j is the j-th column of the bottom-up weight matrix B.

The function μ_j (also known as the choice function) calculates a measure of similarity between the input pattern $(x_1, x_2, \ldots, x_n)^T$ and the LTM pattern stored in B_j. In the comparison process represented by equation 2.8, the largest μ_j is chosen.

$$\mu_{j^*} = \max_{1\le j\le m} \{\mu_j\}. \tag{2.8}$$

The corresponding neuron u_{j^*} activates while inhibiting all other neurons in F2 ('winner take all' competition rule). If there are more than one maximum values,

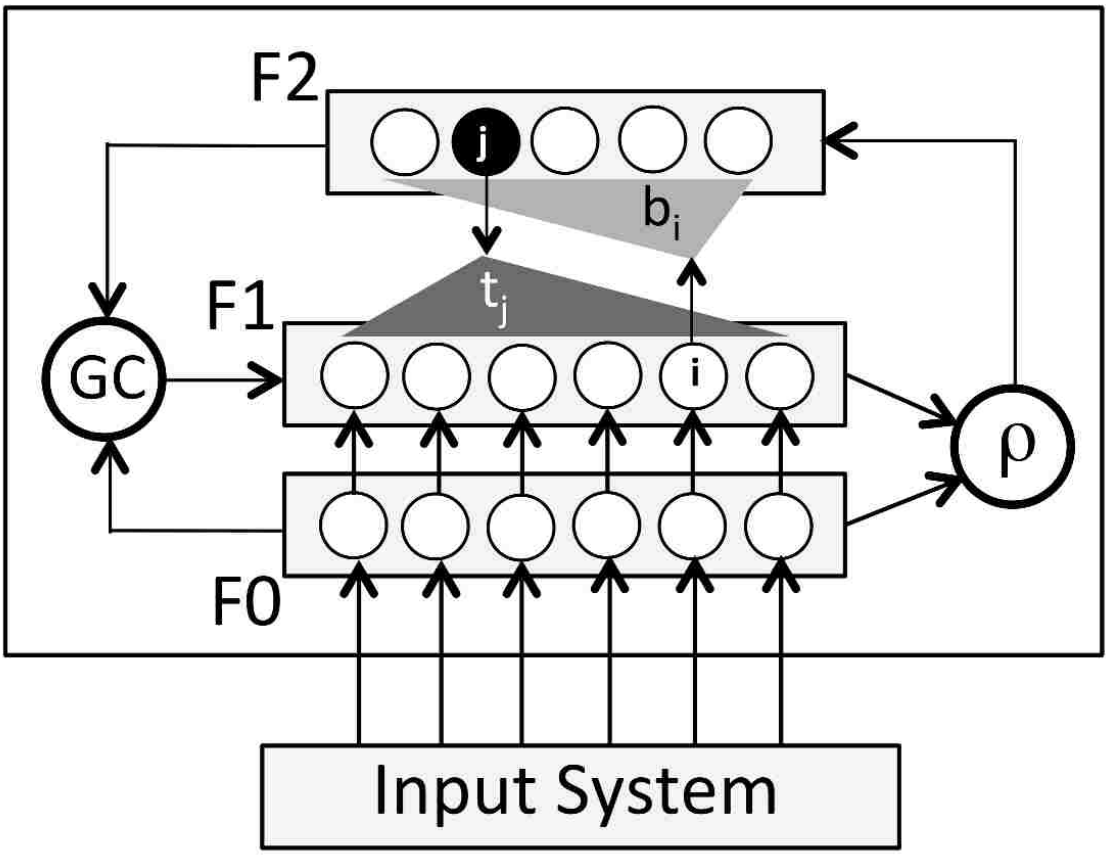

Figure 2.5: ART1 architectural diagram. A binary pattern enters the F0 layer of the neural network from the Input System. This pattern is processed through the F1 layer to from another pattern of inputs denoted b_i into the competitive F2 layer. After that competitive process, with the support of the gain control node labeled GC, an F2 winning node is chosen that in term will output a template pattern t_j back into the F1 layer. The node labeled with ρ controls the granularity of the choice process as described in the text.

the neuron j with the smallest index is chosen, resulting in the F2 output R, given by:

$$\begin{aligned} R &= \{r_1, \ldots, r_{j^*}, \ldots, r_m\}^T \\ &= \{0, \ldots, 1, \ldots, 0\}^T. \end{aligned} \tag{2.9}$$

The gain control G_1 sends excitatory signals to the neurons in layer F1 and is expressed by

$$G_1 = \begin{cases} 1, \ \bigcup_{i=1}^{n} x_i \neq 0 \ and \ \bigcup_{j=1}^{m} r_j = 0 \\ 0, \ otherwise \end{cases}, \tag{2.10}$$

where $\cup$ is the union operation, x_i is the i-th component of the bottom-up weights, and r_j is the j-th neuron output in layer F2. If there is an input vector x and output from layer F2 R= 0, then G_1=1. On the other hand, the gain control G_2 is defined by

$$G_2 = \begin{cases} 1, \ \bigcup_{i=1}^{n} x_i \neq 0 \\ 0, \ otherwise \end{cases}. \tag{2.11}$$

The gain control signal G_2 depends only on the input vector x. If there is an input vector, then the recognition is activated in layer F2.

The weight vector T_{j^*} which is the column in T associated with the winner j^*, representing feedback from the recognition process, is

$$t_{ij^*} = \sum_{j=1}^{m} t_{ij} R_j, \tag{2.12}$$

where the t_{ij} is binary valued. A vigilance subsystem operated by the reset controller in Figure 2.5 checks the appropriateness of the active F2 neuron, according to :

$$\gamma_{j^*} = \frac{|x \cap T_{j^*}|}{|x|} \geq \rho, \tag{2.13}$$

where $|x|$ is the 1-normal $|x| = \sum_{i=1}^{N} x_i, \cap$ is the intersection operation, and $\rho \in (0, 1]$ is the vigilance parameter.

If equation 2.13 is satisfied, then j^* remains the winning neuron in the layer F2. If not, the vigilance subsystem resets the current active neuron in layer F2 by forcing $\gamma_{j^*} = 0$. Another active neuron in layer F2 is chosen with maximum μ_j; this process continues until a winning neuron j^* in layer F2 satisfies the vigilance subsystem. If no neuron satisfies the vigilance subsystem, then a new F2 neuron is recruited (corresponding to a new class) and m is incremented by 1. This last mechanism is what allows an ART network to learn new classes of pattens without losing prior learning. High values of the vigilance parameter ρ result in fine-grained memory with many classes which are the categories created in the layer F2, while low values of ρ produce few broad classes. While in theory, learning could go on indefinitely by adding new neurons to F2, a maximum number is set in practice.

In summary, the dynamics of the ART1 neural network can be described in the sequence of instructions below.

1. Initialize:

$$m = 1 \tag{2.14}$$

$$b_{ji}(0) = \frac{1}{n+1} \tag{2.15}$$

$$t_{ji}(0) = 1 \tag{2.16}$$

2. Read Input Pattern: present a binary pattern $x = [x_1, \ldots, x_n]$, where $x_i \in \{0, 1\}$

3. Calculate the similarity μ_j (see Eq. 2.7)

4. Choose F2 node j^* that satisfies Eq. 2.8

5. Perform vigilance criterion check

$$\gamma_j \begin{cases} \geq \rho, & \text{go to step 7} \\ < \rho, & \text{go to step 6} \end{cases} \tag{2.17}$$

6. Mismatch reset:

   Neuron j^* in layer F2 is inhibited by forcing $\gamma_{j^*} = 0$. Go to step 3, find the maximum μ_j from the remaining neurons. If none of the μ_j satisfies the vigilance criterion, then go to step 8.

7. Update weights

$$t_{ij^*}(p+1) = t_{ij^*}(p)x_i \tag{2.18}$$
$$b_{j^*i}(p+1) = \frac{t_{ij^*}(p)x_i}{\alpha + \sum_i t_{ij^*}(p)x_i} \tag{2.19}$$
$$t_{ij}(p+1) = t_{ij}(p) \tag{2.20}$$
$$b_{ji}(p+1) = b_{ji}(p), j \neq j^* \tag{2.21}$$

   where p is the index of the current timestep, and α is the choice parameter. Go to step 2.

8. Create new F2 neuron, corresponding to new category. Set:

$$m = m + 1 \tag{2.22}$$
$$t_{im} = 1 \tag{2.23}$$
$$b_{mi} = \frac{1}{n+1} \tag{2.24}$$

   Go to step 2.

Note that the long-term memory is updated in step 7. Short-term memory (STM) operations, on the other hand, occur in the comparison and the recognition processes.

In some cases, for example when supervising the operation of thermo-mechanical processes, inputs to the supervisory system are analog rather than binary. With minor modifications, the ART1 network architecture can be used for this purpose. This type of network, Fuzzy ART, has the same structure as the ART1 system, with three main differences:

1. The input patterns of Fuzzy ART can be analog (real valued), with $x_i \in [0, 1]$, or binary valued, with $x_i \in \{0, 1\}$.

2. Top-down weight vectors and bottom-up weight vectors are the same, i.e. $B = T^T = W$.

3. The intersection operation in the μ_j, in the learning rule and in the vigilance criterion, is replaced by the fuzzy MIN operator $\wedge$:

$$(x \wedge y)_i = min(x_i, y_i). \tag{2.25}$$

4. The weight vector w_J is updated according to the equation:

$$w_J^{(new)} = \beta(I \wedge w_J^{(old)}) + (1 - \beta)w_J^{(old)}, \tag{2.26}$$

where β is the learning rate, usually 1 for "fast learning" and $\beta < 1$ for "slow learning".

Compared with other neural networks, ART networks have the following advantages when applied to monitoring and automatic detection for SHW systems:

1. An ART network is an unsupervised neural network, and is able to learn new classes without weakening previously learned classes. The previously learned knowledge is stored in the long-term memory. For most of the neural network algorithms (such as multi-layer perceptron, back-propagation, radial basis function), when the training process is finished, these neural networks can be used when the problems have stationary domain. If new patterns appear at their inputs, these neural networks can not recognize these new patterns or will give wrong predictions. In order to add the new patterns into the networks, both the new and previous classes have to be retrained. These neural networks are not suitable for the systems which operate in possibly noisy and unstationary environments, such as SHW systems which are severely affected by weather conditions and residential usage.

2. An ART network has high computational efficiency and the training process is much faster compared with multi-layer perceptron, back-propagation, radial basis function. An ART network is suitable for the implementation of on-line monitoring and automatic detection.

3. The monitoring and automatic detection based on ART network can be implemented in larger, more complex systems such as commercial building HVAC systems or subsystems.

2.3.2 The hierarchical ART neural network

By varying the vigilance parameter, it is possible to set the classification strategy of an ART (binary or fuzzy) network from very coarse to very fine-grained. An excessively fine-grained classification could result in many false alarms, while an excessively coarse classification could miss important signals of a developing failure. To overcome this dilemma, it is possible to utilize a series of ART networks, which are connected in a hierarchical structure (Caudell et al., 1994). In these, an initial coarse-grained classification (i.e. with low vigilance parameter) is followed by subsequent finer-grained ones (with successively higher vigilance parameter).

A hierarchical ART (HART) network is illustrated in Figure 2.6, and an input pattern under examination traverses from bottom to top. At the lowest level, the pattern is either classified into an existing class, or a new class is created if the pattern is novel. At this level, the vigilance parameter ρ_{00} is low, and the number of classes is small. Novelty only arises if the pattern is substantially different from any of the existing ones, such as would be the case for the catastrophic failure of an important system component. Accordingly, creation of the new class would generally be associated with a 'high-severity' alarm. Following classification or novelty detection (new classes which are created in the category layer), the input pattern

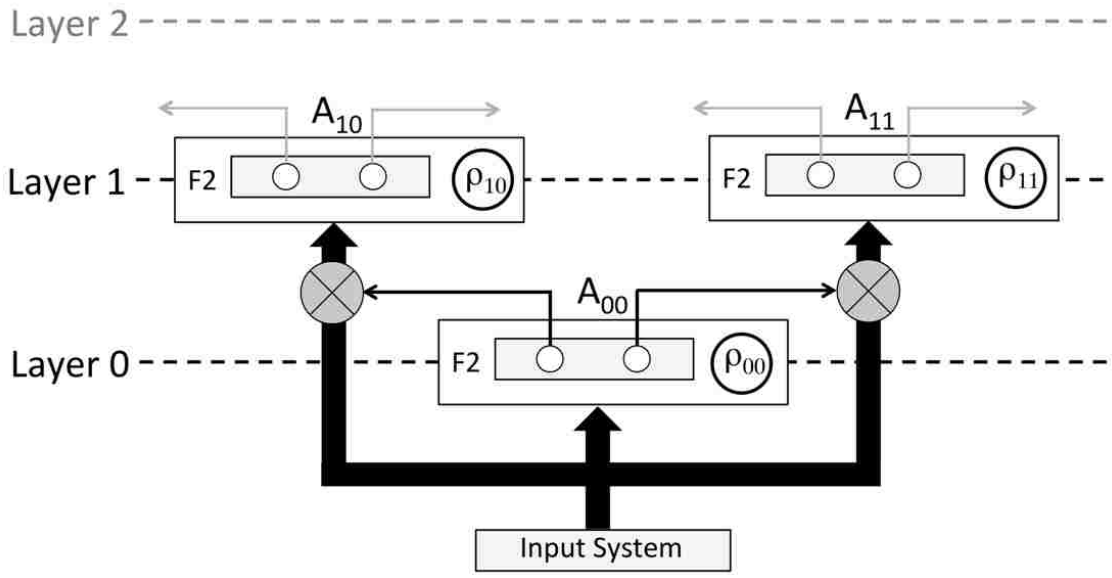

Figure 2.6: Hierarchical ART system architecture. The figure illustrates the first and second layer of a possibly multi-layered tree structure. The internal structure of the ART networks, labeled with A_{kl}, has been simplified for clarity. As described in the text, the system is trained with multiple presentation epochs using a training data set. During the first epoch, the Layer 0 ART learns to classify the data with a relatively low ρ value, generating a coarse partitioning of the data. During the second epoch, the training data is gated into Layer 1 ARTs according to this partitioning for finer grain classification.

is routed to an ART network at the next level up, that is uniquely associated with the class just chosen. All ART networks at this new level are characterized by a vigilance parameter $\rho_{1i} > \rho_{00}$. Note that in principle each ρ_{1i} could take different values, although in the present case a single vigilance parameter (ρ_k) for each level k is adopted. The input pattern is again classified, and either matched with an existing class, or, if the pattern is novel, a new class is created. Novelty at this level may result from a less severe failure, from progressive component degradation, or from hitherto unseen, but normal operating conditions, a fairly common occurrence in renewable energy systems. An alarm would still be issued when the novelty is detected, but with reduced severity (Caudell and Newman, 1993). The input pattern is then

passed up the tree to the ART network at the next level until the penultimate level is reached.

For the specific case of a cascade of Fuzzy ART modules (Caudell et al., 1994), the specific steps for hierarchical classification of an input pattern are:

1. Initialize:
   Set the number of layers, $L+1$,
   Set the vigilance parameter, $\rho_0 < \rho_1 < \cdots < \rho_L$,
   Set the initial weights, $w_{k:ij} = 1$

2. Read Input Pattern:
   Present an analog pattern $x = \{x_1, \ldots, x_n\}^T$, where $x_i \in [0, 1]$

3. Bottom-up and top-down learning process:
   Input pattern for layer k is x^k, $(0 \leq k \leq L)$
   Input pattern for layer $k+1 (k \leq L)$ is $x^{k+1} = x^k$
   In layer $F1^K$, if the class j of $F2^k$ is active and Fuzzy ART module $k-1$ is in resonance, $y1^k = x^k \wedge w_{k:j}$; else $y1^k = x^k$.
   In layer $F2^k$, if the class j of $F2^k$ is active and Fuzzy ART module $k-1$ is in resonance, $y2_j^k = 1$; else $y2_j^k = 0$.
   If the module $k-1$ is in resonance, μ_j^k is calculated by

$$\mu_j^k = \frac{|x^k \wedge w_{k:j}|}{\alpha + |w_{k:j}|}, \tag{2.27}$$

   where α is the choice parameter. Note that α should be set to a small positive value for single pass convergence with Fuzzy ART.
   The vigilance criterion for layer k is

$$\frac{|w_{k:J} \wedge x^k|}{x^k} \geq \rho_k \tag{2.28}$$

where the index J corresponds to the maximum value of μ_j^k.

4. Update weights:

   If the active class in layer $F2^k$ is J and inequality 2.28 is true, then update the weights:

$$w_{k:J}^{new} = \beta(x^k \wedge w_{k:J}^{old}) + (1 - \beta)w_{k:J}^{old} \tag{2.29}$$

5. Go to step 2 until no new class is created and the weights are stable.

The development of the ART networks used in this work is described in this chapter. The training data for the ART network can be collected from historical fault-free measurement or simulation models. In this study, training data for the ART networks are generated by simulation. The SHW simulation model and verification of the ART-based fault detection system by experiment and simulation are described in the following chapters.

Chapter 3

Development of the SHW testbed

To demonstrate the capacity of ART-based neural networks to detect and predict faults on a residential SHW system, a replica of a typical installation was constructed - the solar hot water reliability testbed (SHWRT)(Menicucci et al., 2011). It has the capability of being configured to represent two kinds of SHW systems, a closed-loop active one (usually found in colder climates, with glycol-water antifreeze mixtures as the heat medium, see Figure 3.1) and a drainback one (prevalent in warm climates, with distilled water as the heat medium, see Figure 3.2).

In closed-loop active system, the system consists of collectors, pipes, pump, automatic valves, a storage tank, an expansion tank and heat exchanger. The pump is controlled by a differential temperature controller. One temperature sensor is in the solar collector, and one temperature sensor is in the water tank. When the temperature of the collector is higher than the temperature in the water tank, the pump will start to run, continuing until the collector cools. In this model, the heat-transfer fluid is pumped from the tank to the collectors by a pump and back by gravity to the storage tank. Compared to the closed-loop active system, this system has no check valves and no expansion tank, and requires less maintenance.

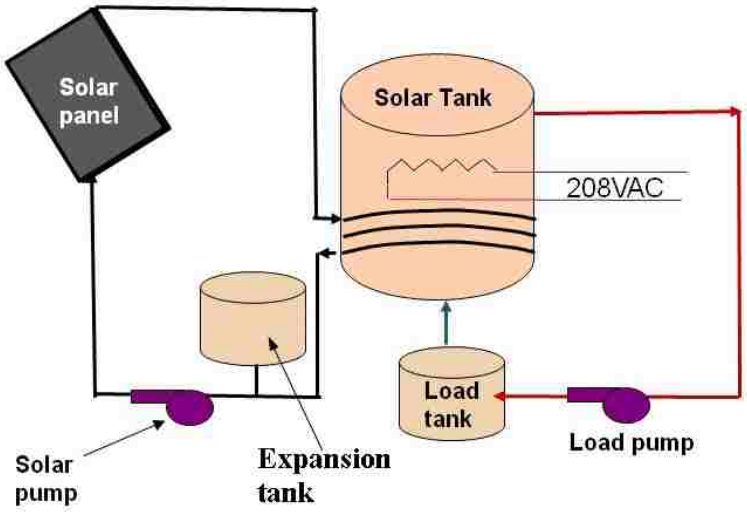

Figure 3.1: SHWRT in pressure mode. In this model, the heat-transfer fluid circulation is controlled by the electric pumps, valves, and controllers.

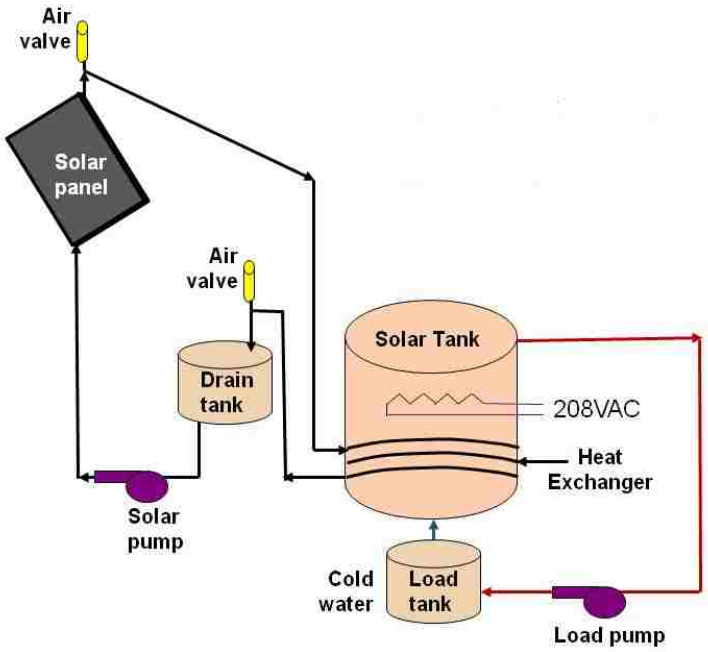

Figure 3.2: SHWRT in drainback mode. In this model, the heat-transfer fluid is pumped from the tank to the collectors by a pump and back by gravity to the storage tank.

3.1 SHWRT configuration and operation

Two principal features distinguish the SHWRT from a typical installation (Menicucci et al., 2011):

1. The SHWRT is much more highly instrumented than a typical residential system, so that every detail of its operation is known, for the purposes of post-failure diagnosis, discovery of important monitoring parameters, etc.

2. While in a residential system the load results from the use of domestic hot water, in the SHWRT the load is provided by a separate cooling system with storage, which can be programmed to mimic arbitrary load profiles.

Table 3.1: Components of the SHWRT

Component	Part Specification
Collectors	Lennox LSC-18
Storage tank	SunEarth SU80-HE-1
Load tank	55 gal drum; insulated
Load tank chiller	Neslab RTE-8
Pumps	B & G
Plumbing	Copper, Type L and M
Instrumentation (pressure, flow, etc)	Mostly Omega
TC acquisition	Agilent 34970A
TCs	Type K, Type T, welded in-house
Instrumentation controller	National Instruments
Electrical device controller	Custom designed, built with solid state

The principal components of the SHWRT are shown schematically in Figure 3.3 and listed in Table 3.1 in detail. These are: solar collector array, solar pump, load

pump, solar storage tank with integral heat exchanger, cooling tank, chiller, drainback tank, expansion tank and computer/data acquisition system. In the SHWRT, the real load (resulting from typical domestic activities such as bathing & showering, washing dishes etc.) is replaced by a simulated load, provided by a load tank and a load pump, under computer control. The load tank is stratified, with the bottom layer containing water at a temperature similar that found in a municipal water supply. To simulate the load, cold water drawn from the bottom of the load tank is pumped to the bottom of the storage tank in precisely metered amounts. Because the hot storage (solar) tank is full, the same volume of water flows from the top of the solar tank to the top of the load tank. A temperature-controlled circulating bath is used to remove heat from the load tank accumulated as a result of the load simulation. Hot water is drawn from the top of the load tank, cooled in the bath, and returned to the bottom of the tank at a set temperature. Aside from the obvious practical advantages, using a simulated load allows its consistent operation, so that the variability of a real load is removed from the experimental parameter space, thereby isolating the effects of weather and solar irradiance variability. For the purposes of this experiment, a scaled version of the standard load profile published by the Florida Solar Energy Center is used (Fairey and Parker, 2004). The load profile used here is shown in Figure 3.4.

The solar loop can function in two modes: closed loop and drainback, by operating valves that either isolate the drainback tank or the expansion tank. When in drainback mode, the system is not pressurized. When instructed to do so by the system controller, the solar pump sends heat medium (a 50% propylene glycol / water mixture in this case) to the collectors, where it is heated and returned to the solar tank heat exchanger. From the heat exchanger, heat medium either flows back to the pump (when in closed loop mode) or into the drainback tank (when in drainback mode).

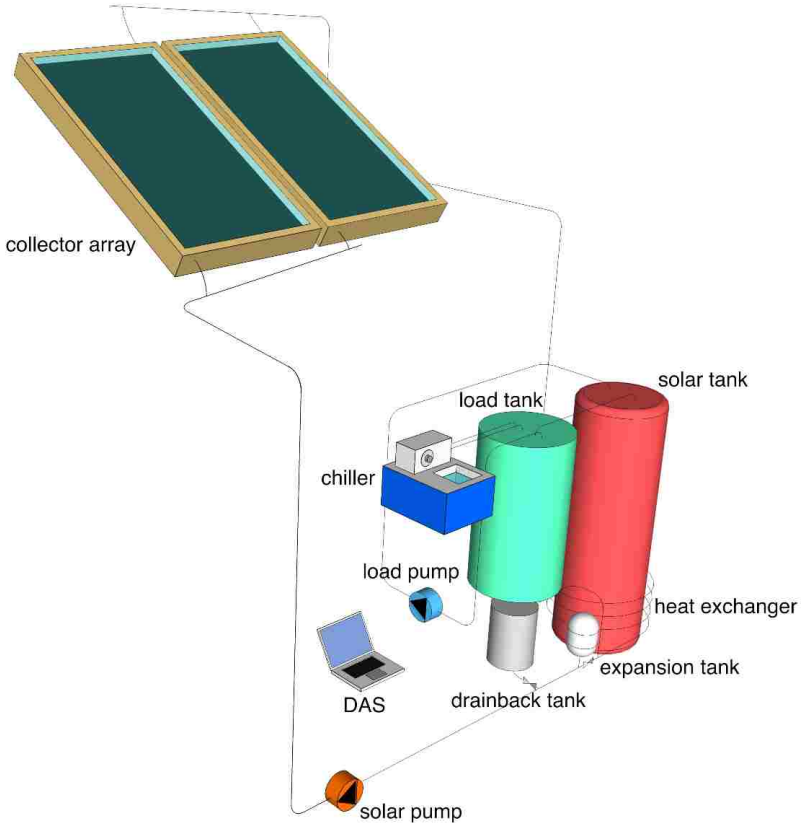

Figure 3.3: Principal components of the SHWRT: the solar loop is composed of the solar collector, the solar pump, the heat exchanger and either the expansion tank (for closed loop operation) or the drainback tank (for drainback operation). Heat is stored in a stratified hot water tank. The load loop is composed of cold tank, load pump and chiller. Control and data logging are operated by a computer.

The solar tank heat exchanger consists of a copper coil wrapped around the lower quarter of the exterior of the solar storage tank. Good thermal contact between the copper cols and the tank wall is ensured by a silicon-based heat conduction compound. An electrical resistance heater (3 kW) in the tank heats the water when solar heat is not available. The location of the heating element, close to the top of the tank, ensures that solar heating, if available, is the default heating mechanism, and that electric heating only occurs when the solar tank is almost completely discharged.

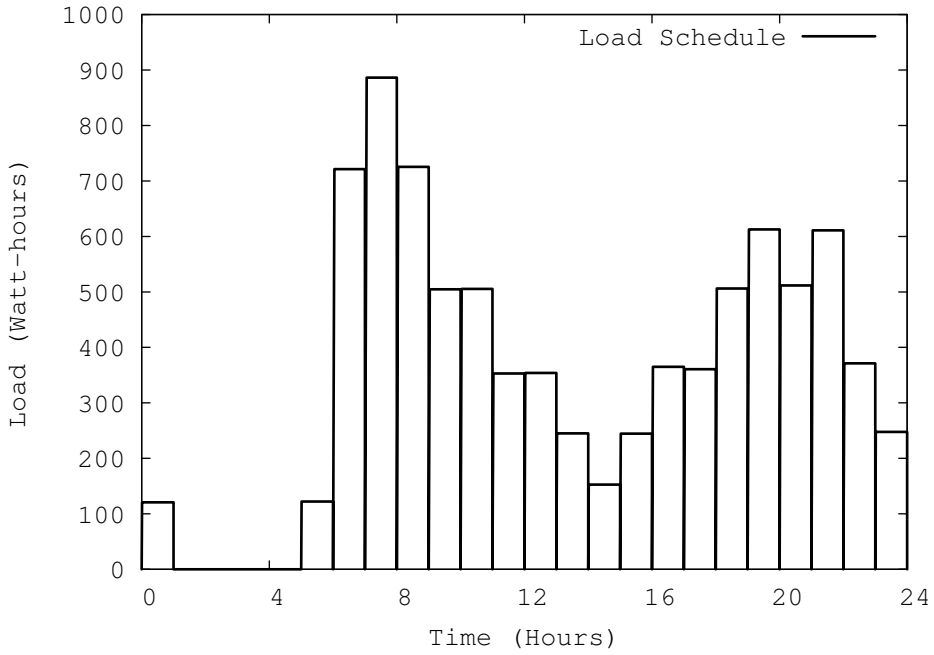

Figure 3.4: Hourly loads used for simulating typical residential usage. A load pump operates while the control system calculates the energy draw in real time. The load pump stops when the load is met.

The control system monitors the temperature of the solar collector plate (the fin temperature), and the temperature of the heat exchanger outlet. When the difference between the fin temperature and the heat exchanger outlet temperature exceeds a specified value (7°C in this case), the solar pump is activated. The solar pump remains on until the temperature difference becomes negative. If the bottom temperature is greater than a set threshold value (60°C in the present case), or the fin temperature is above a set limit (95°C in the present case) then the tank is saturated and can not absorb additional heat, so the pump is turned off.

The SHWRT system employs two Lennox LSC-18 collectors, with low-iron double glazing and black chrome absorber, as shown in Figure 3.5 (Ortiz, 2008). The collector was manufactured in the early 1980s and was certified by Solar Rating and Certification Corporation (SRCC).

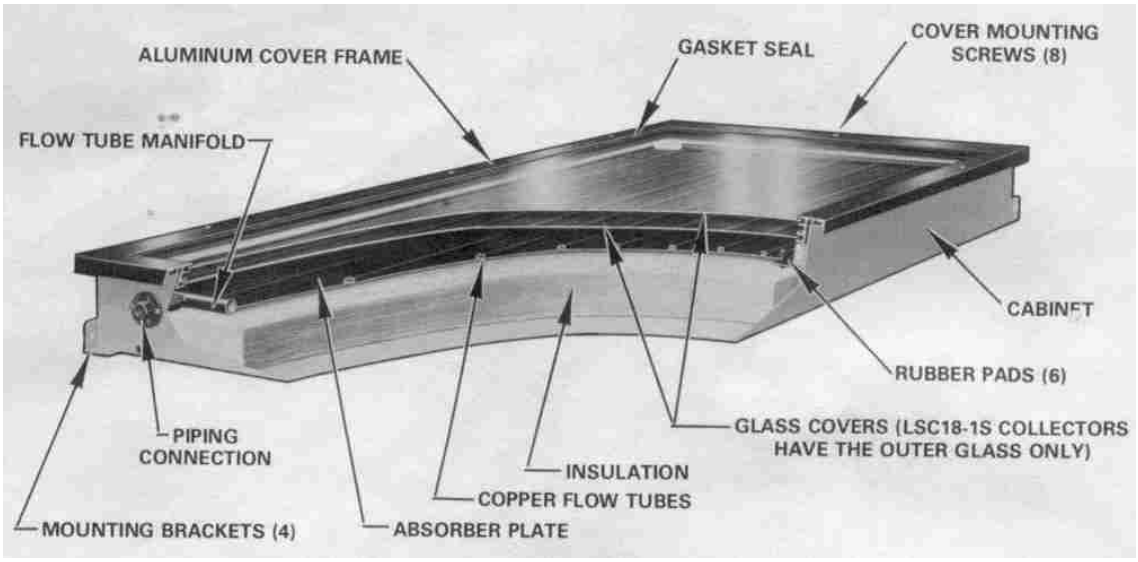

Figure 3.5: Lennox LSC-18 collector

3.2 SHWRT Instrumentation

The SHWRT is a testbed and as such it is much more highly instrumented than a commercial system. For example, in a typical residential SHW system, there are normally only two temperature sensors. One is located on the outlet of the collector and the other is on the supply line, at the outlet of the solar storage tank. The temperature difference between these two sensors is used by a commercial controller to turn on and off the solar loop pump. The SHWRT, however, contains many more sensors.

The solar tank and the load tank were instrumented with type T thermocouples. A thermocouple tree consisting of eight thermocouples located along a CPVC pipe was installed in approximately the center of the tank. The plastic pipe is used only to hold each thermocouple in place, approximately equidistant from one another along the vertical axis of the tank.

Similarly, thermocouples were placed along the outside skin of the metal tank under the insulation in approximately the same vertical locations as those on the internal tree. To install them the exterior skin of the tank was carefully cut and the insulation was removed. The exterior metal surface of the water-bearing tank was cleaned and the thermocouples were glued in place using thermal epoxy. Figure 3.6 shows graphically how the thermocouple trees are installed in the solar tank. Figure 3.7 shows diagrammatically the array of sensors located on the system.

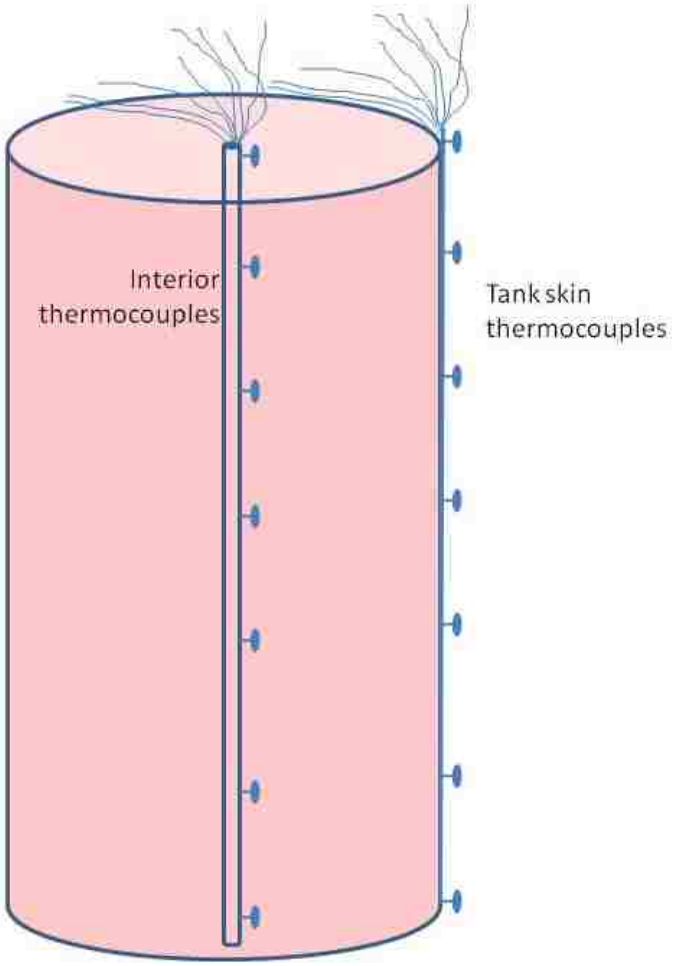

Figure 3.6: Thermocouple tree in the solar tank, thermocouples were placed in approximately vertical locations for both the outside skin and the internal tree.

Because this SHWRT system contained many more features than a commercial system, its control system was much more sophisticated. As an example of this complexity, Figure 3.8 shows the logic diagram for controlling the solar pump. Only a small portion of this logic would be implemented in a commercial controller.

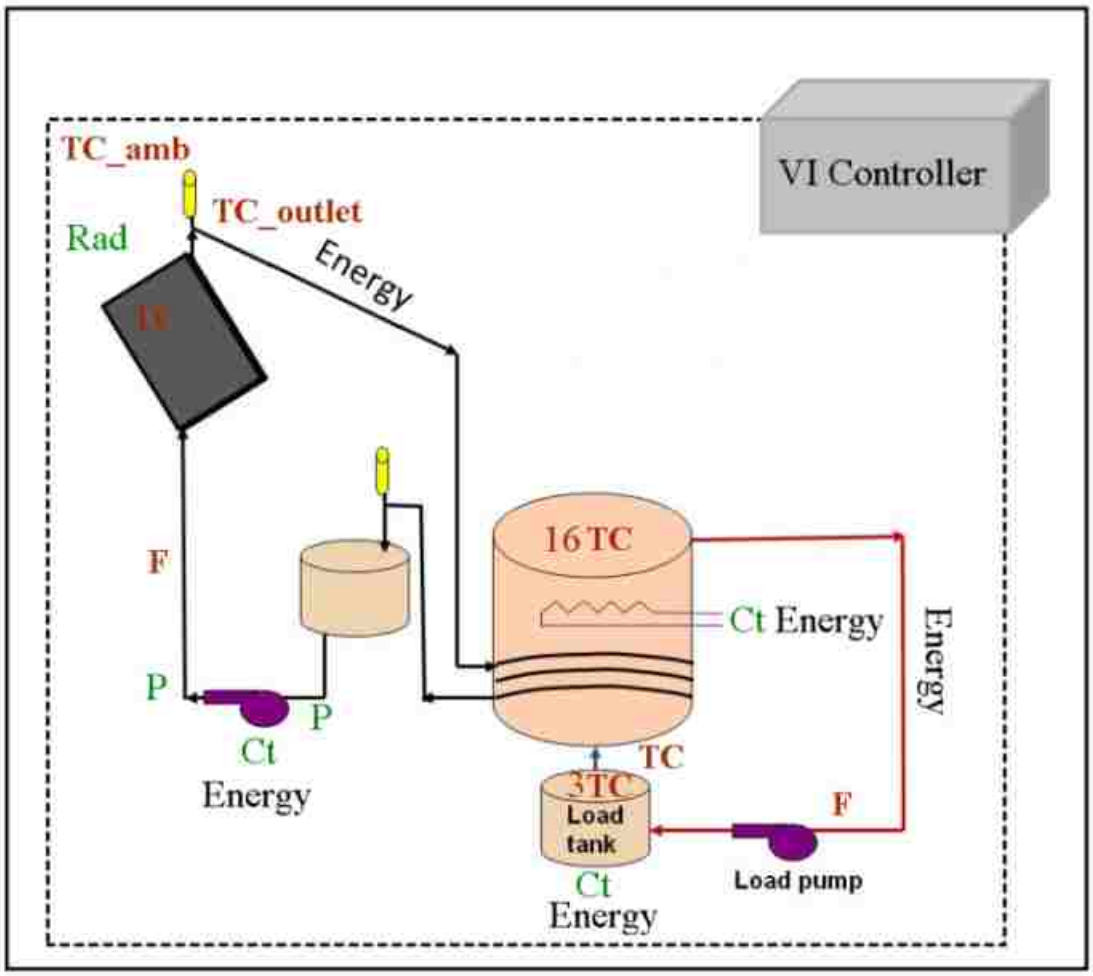

Figure 3.7: Schematic illustration of the sensors installed in the SHWRT: thermocouples (TC), pressure sensors (P), flow meters (F), solar radiation sensors (Rad), current transducers (Ct). 'Energy' represents a point in which the energy generated in the loop is computed.

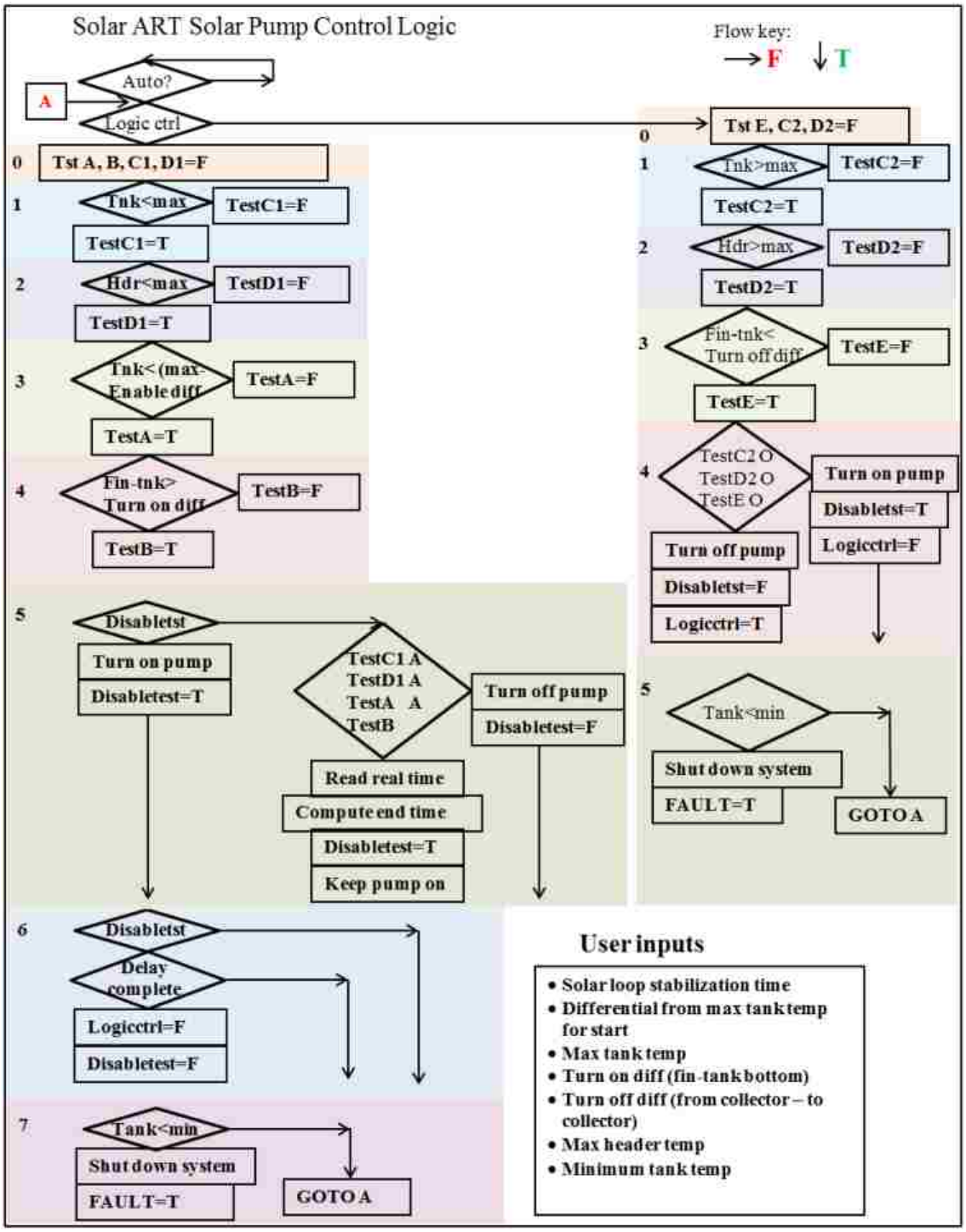

Figure 3.8: Solar pump control logic

Chapter 4

TRNSYS models

Despite its speed in learning compared to other classes of ANNs, the number of possible combinations of operating conditions and weather / solar irradiance conditions is so large that several years of training are needed to allow the ART network to observe and categorize the majority of 'normal' operating states it is likely to encounter. Clearly, doing this experimentally or in the field is impractical. Model-based training, on the other hand, is an attractive option, with the caveat that the model must be an accurate representation of the real system, and must be accompanied by accurate weather and irradiance date.

4.1 Development of the SHWRT TRNSYS model

The model of the SHWRT is implemented in TRNSYS, a module-based code specifically designed for transient simulation of energy systems. In TRNSYS, a wide array of standard components can be selected from module libraries. Tunable parameters associated with each component allow the user to develop accurate representations

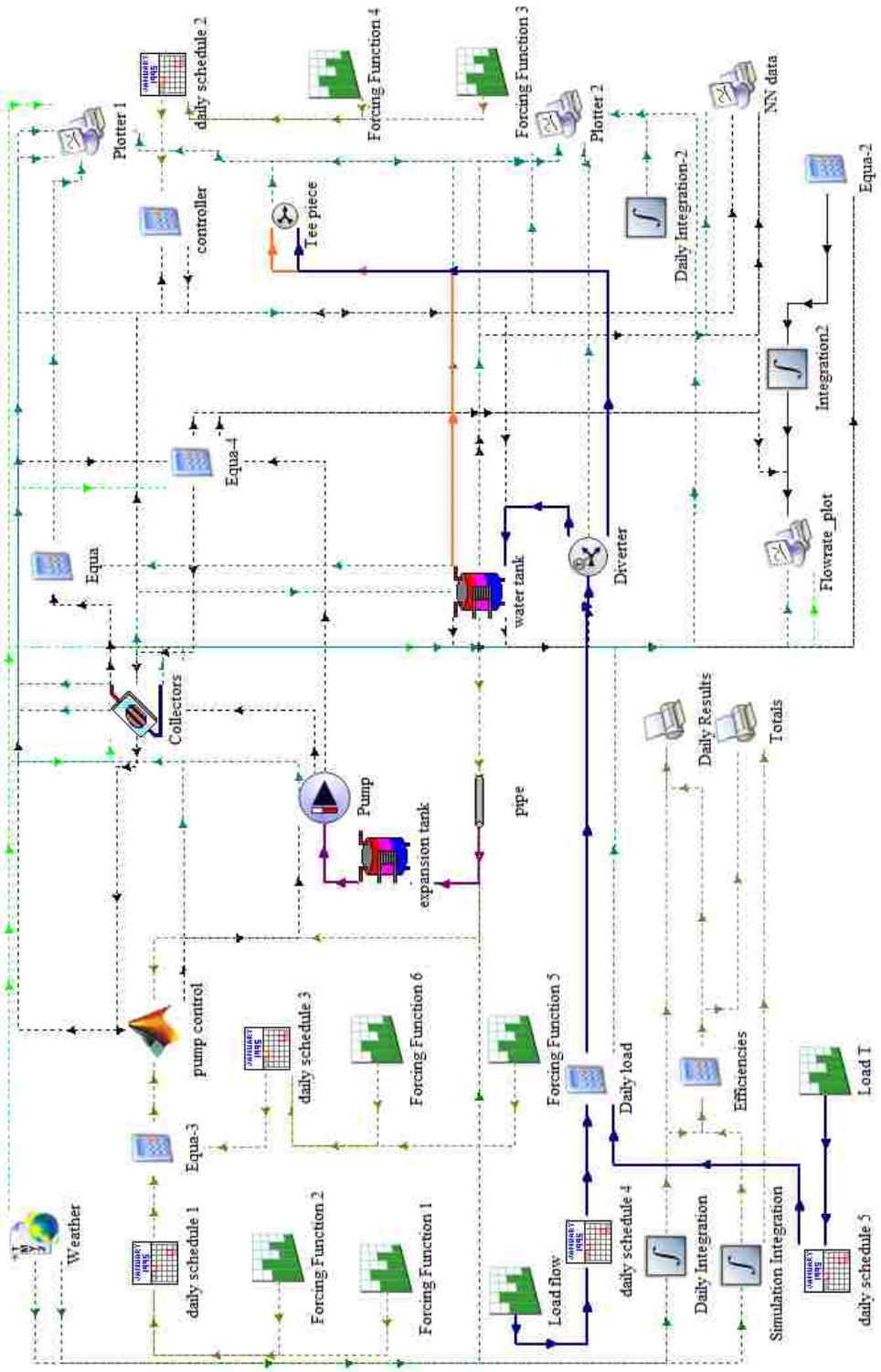

Figure 4.1: TRNSYS model of the SHWRT. Parameters for collectors, storage tanks, pumps and heat exchanger are direct representation of their real counterparts. Weather and irradiance data are from historical TMY2 data bases, or from direct measurement. A Matlab module is used to implement of the controller logic. Various output devices collect data for off-line training of the ART algorithms.

of real system components. Within TRNSYS, individual components are represented by a set of differential or algebraic equations that closely approximate the system physics. The modules are connected to each other via information or material flow lines. The layout of the TRNSYS model of the SHWRT is shown in Figure 4.1.

Table 4.1: Collector type 564 properties

Parameter	**Value**	**Units**
Collector length	1.71	m
Collector width	1.61	m
Absorber plate thickness	0.0012	m
Conductivity of absorber material	194	$kJ/hr \cdot m \cdot K$
Number of tubes	20	
Inner tube diameter	0.0044	m
Outer tube diameter	0.0064	m
Bond resistance	0.05	$h \cdot m^2 \cdot K/kJ$
Fluid specific heat	4.19	$kJ/kg \cdot K$
Absorptance of the absorber plate	0.88	Fraction
Emissivity of the absorber plate	0.9	Fraction
Top loss mode	1	
Number of identical covers	2	
Index of refraction of cover material	1.53	
Extinction coefficient, thickness product	0.005	
Emissivity of the glass	0.9	Fraction
Plate spacing	0.025	m
Glass spacing	0.01	m

The principal components of the TRNSYS model are the solar collectors, the storage tank, and the control system. The collector module from the TRNSYS Thermal

Energy System Specialists (TESS) library that best matches the hardware installed in the SHWRT is the type 564. A number of input parameters must be defined so that the modeled collector is an accurate representation of the real one. Some of these parameters, such as physical dimension, were measured directly. Others, such as the bond thermal resistance between the riser tubes and the collector fins, are not known and were estimated based on engineering judgment. The parameters used for the type 564 module are listed in Table 4.1.

Ideally the model should be an exact representation of the real system. In reality, it was impossible to match the model to the system because some components in the system had no corresponding component within TRNSYS 16. For example, water tank type 534 from the TRNSYS library mateched the testbed's tank most closely, however the heat exchanger is different. As shown in Figure 4.2, the type 534 heat

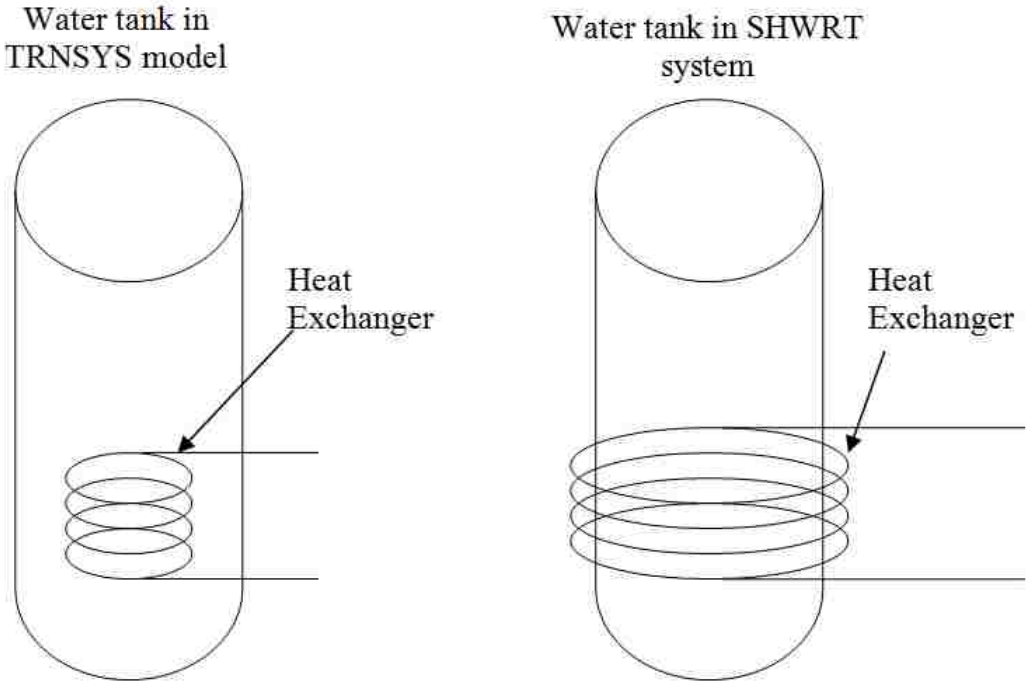

Figure 4.2: Diagrams of the modeled (left) and actual (right) water tank in the SHW system

exchanger is immersed inside the water tank, in direct contact with the water, while the heat exchanger in the testbed's tank consists of a copper tube coiled around the outside of the tank, and thus not in direct contact with the water. Some of the important parameters used to characterize tank type 534 are shown in Table 4.2.

Table 4.2: Properties of tank Type 534 used for the SHWRT model

Parameter	**Value**	**Units**
Number of tank nodes	8	
Number of ports	1	
Number of immersed heat exchangers	1	
Inversion mixing flow rate	-10	$kJ/hr \cdot m \cdot K$
Tank Volume	0.297	m^3
Tank Height	1.4	m
Top loss coefficient	15	$kJ/hr \cdot m^2 \cdot K$
Bottom loss coefficient	5	$kJ/hr \cdot m^2 \cdot K$
Edge loss node # 1	5	$kJ/hr \cdot m^2 \cdot K$
Edge loss node # 2	5	$kJ/hr \cdot m^2 \cdot K$
Edge loss node # 3	2	$kJ/hr \cdot m^2 \cdot K$
Edge loss node # 4	2	$kJ/hr \cdot m^2 \cdot K$
Edge loss node # 5	15	$kJ/hr \cdot m^2 \cdot K$
Edge loss node # 6	20	$kJ/hr \cdot m^2 \cdot K$
Edge loss node # 7	20	$kJ/hr \cdot m^2 \cdot K$
Edge loss node # 8	10	$kJ/hr \cdot m^2 \cdot K$
Top loss temperature	22	°C
Bottom loss temperature	22	°C
Flue loss temperature	22	°C
Exit node	1	
Number of miscellaneous heat flows	0	

The input parameters for the type 534 tank model within the TRNSYS were adjusted based on trial and error comparisons until its predicted behavior mirrored the real system sufficiently.

There are eight temperature sensors in the water tank from bottom to top. In the TRNSYS model the water tank is divided into eight zones, each representing an equal portion of the water tank from the bottom to the top. The location of the nodes corresponds to the placement of the thermocouples in the water tank.

The upper four zones in the TRNSYS model perform very similarly to the real zones, but the temperatures of the lower four zones are slightly different because the heat exchanger is located in the lower part of the water tank. The wrap-around heat exchanger displaced insulation in the bottom portion of the tank, resulting in more heat loss in that part of the tank than in the upper half. These nuances were incorporated into the model by adjusting the input parameters until the modeled behavior of the tank reasonably represented the actual behavior.

The controller is implemented using a Matlab module (listed in full in Appendix A), based on the logic described previously. The weather data are typical meteorological year (TMY) data sets which are derived from the 1991-2005 National Solar Radiation Data Base (NSRDB) archives (Wilcox and Marion, 2008).

4.2 Verification of the SHWRT TRNSYS model

To ensure that the TRNSYS model is an accurate representation of the physical system, several tests were performed on both the SHWRT and the model under identical inputs (i.e. weather and solar irradiance conditions recorded via the SHWRT during the experiment were used as inputs for the model). The validation experiment was conducted from 4PM on December 13, 2010 (hour 8318) to 9AM on December 16,

2010 (hour 8385).

The TRNSYS model produced a set of predicted behaviors for the solar system's various components, such as the temperatures in the tank, and the collector temperature. These modeled values were then compared to the actual measurements for each parameter.

After initial bug fixes, the overall error between the model's predictions and the measured values was reduced to an acceptable level, less than around 5%.

Some of the comparisons are shown in the following figures. As shown in Figure 4.3, the TRNSYS model predicts the collector plate temperature very accurately during the day, both for cloudy and clear conditions. On the other hand, the model predicts the night-time temperature accurately for clear conditions (nights after December 13 and 14), but underpredicts temperature for overcast conditions (the night after December 15). The reason for the discrepancy is simple - the night of December 15 was cloudy, and the sky temperature model used in TRNSYS, described by equation 4.1, is obtained from an empirical equation based on ambient temperature (Swinbank, 1963):

$$T_{sky} = 0.0552 T_a^{-1.5}, \tag{4.1}$$

where T_{sky} = sky temperature (°C) and T_a = ambient temperature (°C). This does not account for humidity and cloud cover by overpredicting the radiant heat loss to the sky, and underpredicting the sky temperature substantially under cloudy conditions.

The predicted and measured temperatures of tank nodes 1, 3 and 8, located at the top of the tank, at the heater coil, and near the bottom of the tank respectively, are shown in Figure 4.5. The hourly water draws are clearly visible in the experimental traces. For each hourly draw, the node temperature drops sharply until the lower limit of the temperature control deadband is reached. At this point, the heater

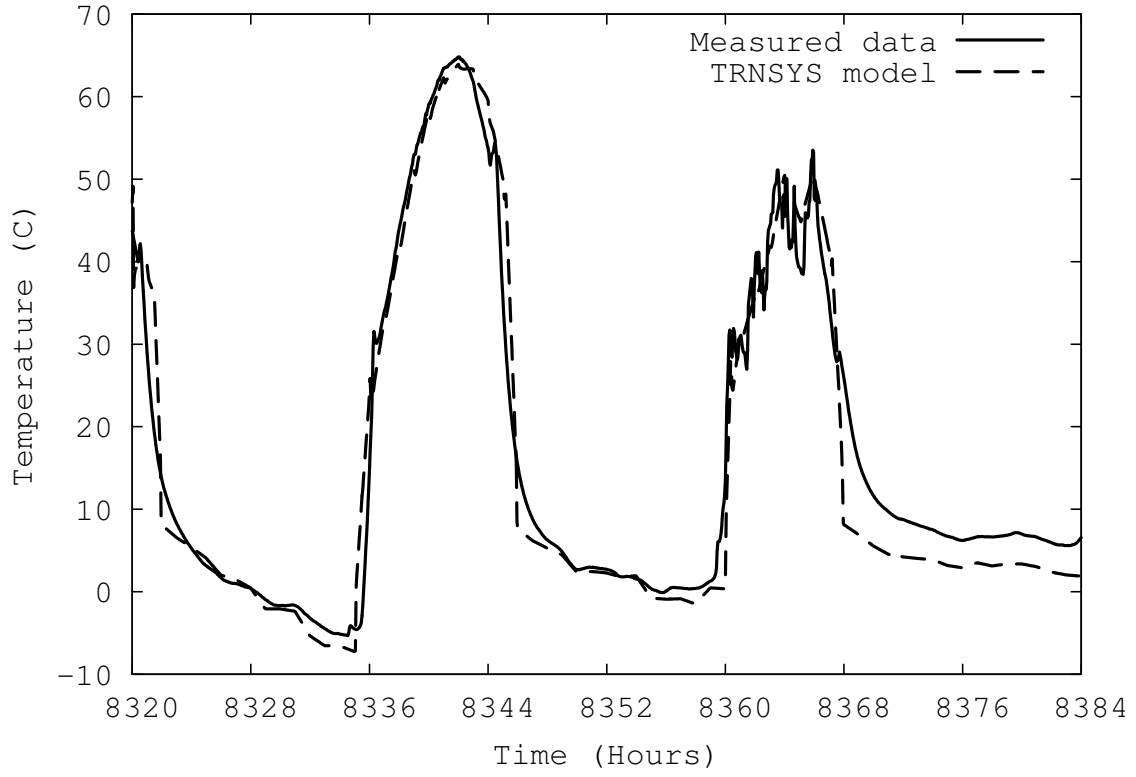

Figure 4.3: Comparison of collector plate temperatures, predicted by the TRNSYS model and measured. The measurement was made using a calibrated T-type thermocouple.

is activated, and remains on until the temperature reaches the upper limit in the deadband.

Figure 4.4 presents a comparison of the predicted and measured useful energy gain by the solar collectors. As can be seen, the overall trends are reasonably accurately predicted with a slight time lag. Because the TRNSYS runs according to the solar time, and there is about a half hour difference between the solar time and our local time. In general, the difference between the predicted and measured total energy, based on the integral under the curves, is around 1%.

Small discrepancies in the temperature predictions can also be attributed to com-

plex three-dimensional flows occurring in the tank, which can not be described by the one-dimensional differential equations in TRNSYS. However, the model is sufficiently accurate for initial training of the network, which can be further refined in the field.

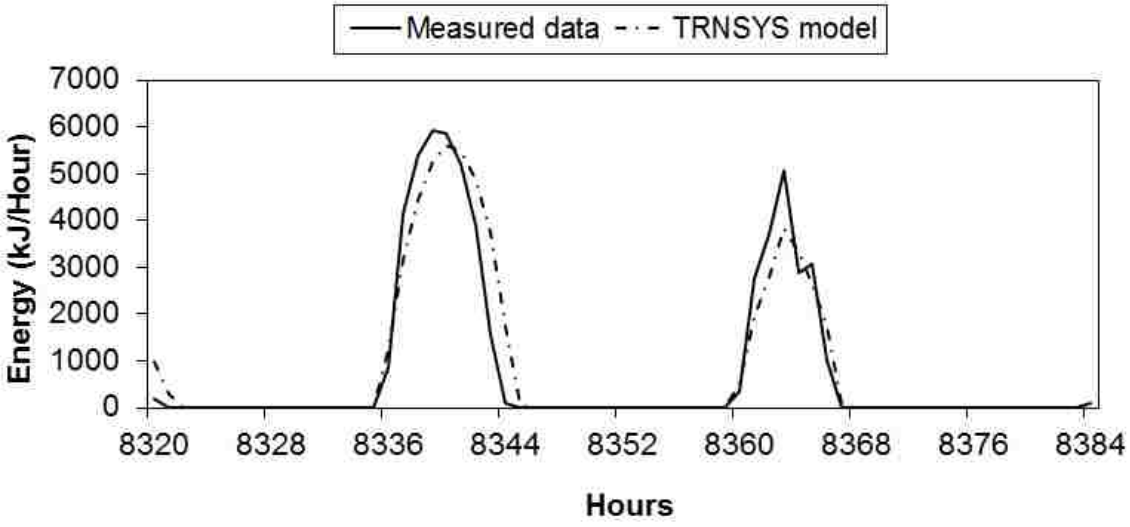

Figure 4.4: Useful energy gain by the solar collectors

4.3 Additional verification of the TRNSYS simulation model

The ART network relies on the TRNSYS simulation model to provide the training data. The accuracy of the predictions of the TRNSYS model is crucial to the correct training ART network. This section presents an additional test to validate the predictions of a TRNSY glazing model, namely, the prediction of collector glazing temperature.

Knowledge of the collector glazing temperature is valuable, since ultimately the glazing is the major source of heat loss in a solar collector. The ability to predict

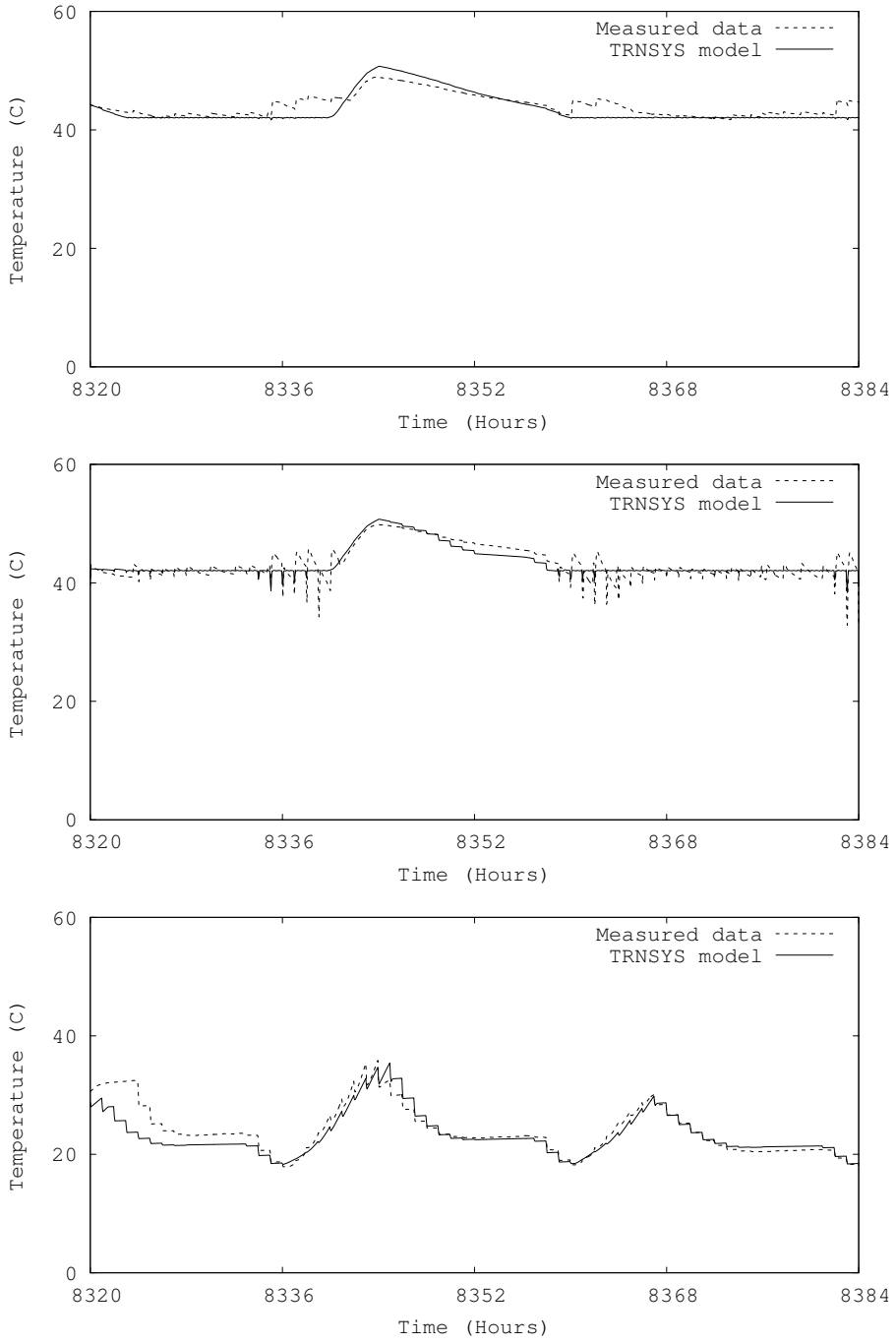

Figure 4.5: Predicted and measured temperatures of the tank at node 1, at the top of the tank, node 3, near the electric heater, and at node 8, near the tank bottom.

glazing temperature for a variety of solar collector types (e.g. single glazed vs. double glazed, painted vs. selective surface) provides an indication of the adaptability of this fault detection system to other configurations. Calculation of glazing temperature is complex, because the heat transfer mechanisms between the absorber plate and the exterior of the collector are themselves complex. Both convective and radiative processes are present in the calculation. Accurate estimates of material/surface parameters may be needed to obtain accurate predictions. The sensitivity of the predictions to knowledge of these parameters is also of interest.

The glazing model was incorporated into the SHWRT testbed model described previously to validate its accuracy. The standard model for flat plate collectors was replaced by a user defined one, based on the following theoretical considerations.

In the SHW testbed, two sensors, the outside temperature sensor and global incident solar radiation, comprise weather information relative to the collectors. Typical weather data for the TRNSYS model come from TMY weather data, which include wind velocity, relative humidity, total radiation on horizontal, horizontal diffuse radiation, solar azimuth, etc. Some of these, such as wind speed, can affect the glazing temperature.

Thus, using the glazed collector components from TRNSYS Thermal Energy System Specialists (TESS) library without some important weather information, some error was expected. For example, TYPE 944 is a double glazed collector component in TRNSYS 17. Using this model with user defined inputs to match the testbed collector and with TMY weather data, the simulation model runs properly. However, if the weather data are limited to only those recorded by the testbed and inputted to the TRNSYS model, run-type errors were encountered. Thus, a glazing collector model needed to be developed that would execute with the available weather data. This also implied that some estimates of certain parameters were needed.

The TRNSYS model of the chrome double and single glazed collectors (model: LSC-18 & LSC-18S, by Lennox Industries, Inc.) is the same as the one used in the previous SHWRT tests, except that the collector component for the chrome double glazed collector is replaced by a user defined collector TYPE 242 and the chrome single glazed collector component is replaced by a user defined collector TYPE 244. The FORTRAN code of collector TYPE 242 is listed fully in Appendix C.

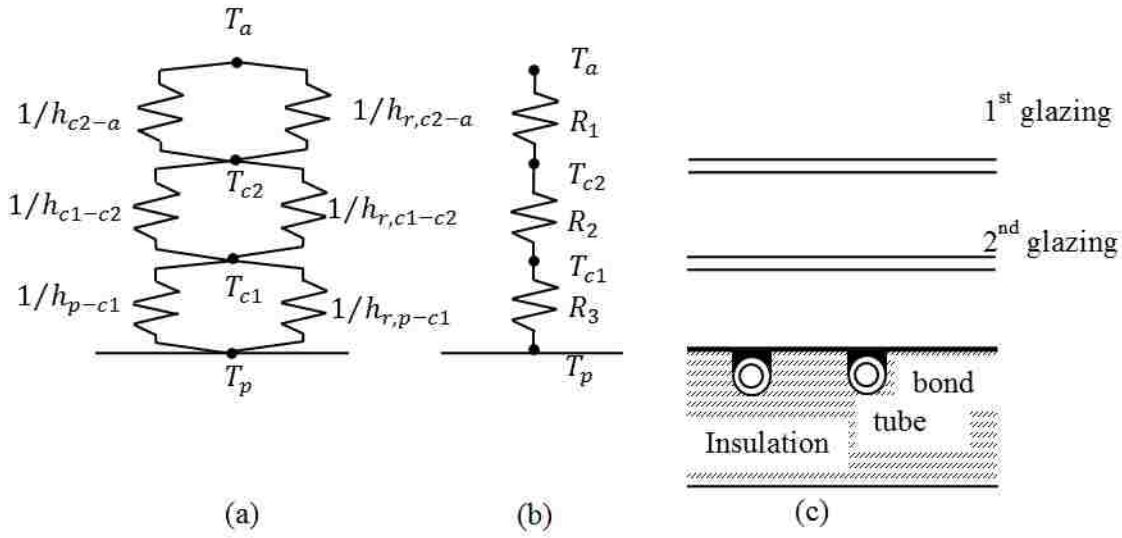

Figure 4.6: Thermal network for a double glazed collector. (a) in terms of conduction, convention and radiation resistances, (b) in terms of combined effective resistances between plates, (c) schematic for double glazing flat plate collector.

The thermal network for a double-cover collector is shown in Figure 4.6. The collector energy top loss is the result of convection and radiation between parallel plates. The energy transfer between the plate at T_p and the first cover at T_{c1} is the same as between any other two adjacent covers and is also equal to the energy loss to the surroundings from the top cover (Duffie and Beckman, 1991):

$$q_{loss,top} = h_{p-c1}(T_p - T_{c1}) + \frac{\sigma(T_p^4 - T_{c1}^4)}{\frac{1}{\epsilon_p} + \frac{1}{\epsilon_c} - 1}, \tag{4.2}$$

where h_{p-c1} is the heat transfer coefficient between collector plate and the first cover.

Equation 4.2 can be written as:

$$q_{loss,top} = (h_{p-c1} + h_{r,p-c1})(T_p - T_{c1}), \tag{4.3}$$

where

$$h_{r,p-c1} = \frac{\sigma(T_p + T_{c1})(T_p^2 + T_{c1}^2)}{\frac{1}{\epsilon_p} + \frac{1}{\epsilon_c} - 1}. \tag{4.4}$$

The thermal resistances R_1, R_2 and R_3 can be expressed as:

$$R_1 = \frac{1}{h_{c2-a} + h_{r,c2-a}} \tag{4.5}$$

$$R_2 = \frac{1}{h_{c1-c2} + h_{r,c1-c2}} \tag{4.6}$$

$$R_3 = \frac{1}{h_{p-c1} + h_{r,p-c1}} \tag{4.7}$$

For the double glazed collector, the top loss coefficient from the collector plate to the ambient is:

$$U_t = \frac{1}{R_1 + R_2 + R_3} \tag{4.8}$$

The new temperature of plate j can be expressed in terms of the temperature of plate i as:

$$T_j = T_i - \frac{U_t(T_p - T_a)}{h_{i-j} + h_{r,i-i}} \tag{4.9}$$

For the single glazed collector, similar expressions can be derived. The parameters used for the chrome single glazed collector TYPE 244 module are listed in Table 4.3, and the parameters used for the chrome double glazed collector TYPE 242 module are listed in Table 4.4.

Table 4.3: User-supplied input parameters for the chrome single glazed collector TYPE 244

Parameter	**Value**	**Units**
Collector length	1.71	m
Collector width	0.805	m
Absorber plate thickness	0.0012	m
Conductivity of absorber material	194	$kJ/hr \cdot m \cdot K$
Number of tubes	10	
Inner tube diameter	0.0044	m
Outer tube diameter	0.0064	m
Bond resistance	0.05	$h \cdot m^2 \cdot k/kJ$
Fluid specific heat	3.747	$kJ/kg \cdot k$
Absorptance of the absorber plate	0.5	Fraction
Emissivity of the absorber plate	0.2	Fraction
Number of identical covers	1	
Index of refraction of cover material	1.53	
Extinction coefficient, thickness product	0.005	
Emissivity of the glass	0.2	Fraction
Plate spacing	0.035	m

The validation experiment was conducted from 2pm on November 15, 2011 (hour 7646) to 9am on November 19, 2011 (hour 7737). As shown in Figure 4.7 and Figure 4.8, the collector plate temperatures of the TRNSYS model is consistent with our measured data, except at night. While in the TRNSYS model the sky temperature is calculated from equation 4.1 (which only relates to the outside ambient temperature), in reality it is also influenced by other factors, such as relate humidity, sunny day or cloudy day, and so on. The calculations performanced here can assist

in determining whether a more accurate sky temperature model is necessary for the purposes of ART training.

Table 4.4: User-supplied input parameters for the chrome double glazed collector TYPE 242

Parameter	**Value**	**Units**
Collector length	1.71	m
Collector width	0.805	m
Absorber plate thickness	0.0012	m
Conductivity of absorber material	194	$kJ/hr \cdot m \cdot K$
Number of tubes	10	
Inner tube diameter	0.0044	m
Outer tube diameter	0.0064	m
Bond resistance	0.05	$h \cdot m^2 \cdot k/kJ$
Absorptance of the absorber plate	0.55	Fraction
Emissivity of the absorber plate	0.5	Fraction
Number of identical covers	2	
Index of refraction of cover material	1.53	
Extinction coefficient, thickness product	0.005	
Emissivity of the glass	0.15	Fraction
Plate spacing	0.025	m
Glass spacing	0.01	m

Figure 4.9 and Figure 4.10 present comparisons of the predicted and measured glazing temperatures of the single and double glazed collectors. As can be seen, the nighttime sky temperature estimation errors produce some errors that are not of consequence to this application. The reason for the discrepancy may be that the wind velocity sensor is not installed in the testbed and in the TRNSYS simulation

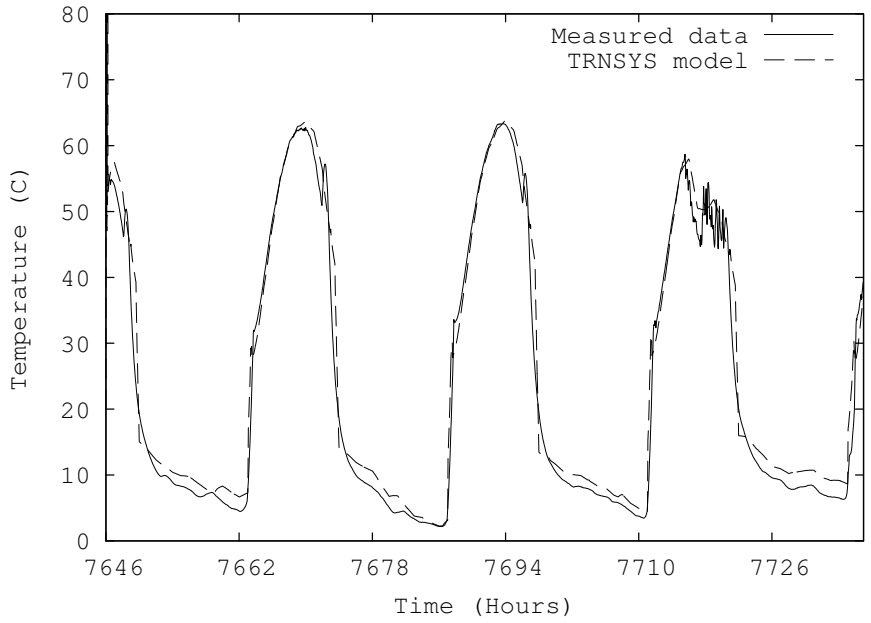

Figure 4.7: Double glazed collector plate temperature for chrome double glazed with black chrome coated absorber surface.

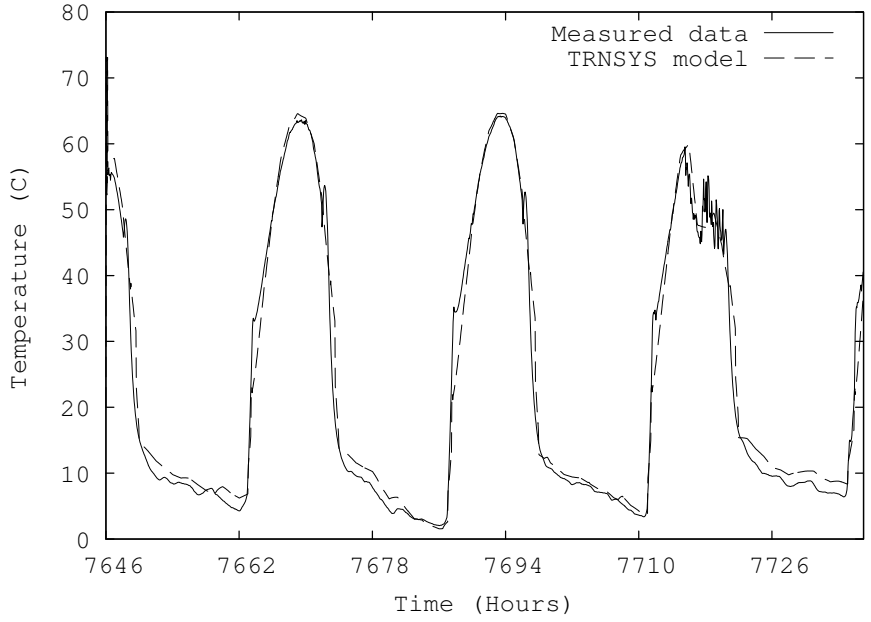

Figure 4.8: Single glazed collector plate temperature for chrome single glazed with black chrome coated absorber surface.

model the information on the wind velocity is collected from TMY weather data. As can be seen, the overall trends are reasonably accurately predicted with a slight time lag. Because the TRNSYS runs according to the solar time, and there is about a half hour difference between the solar time and our local time.

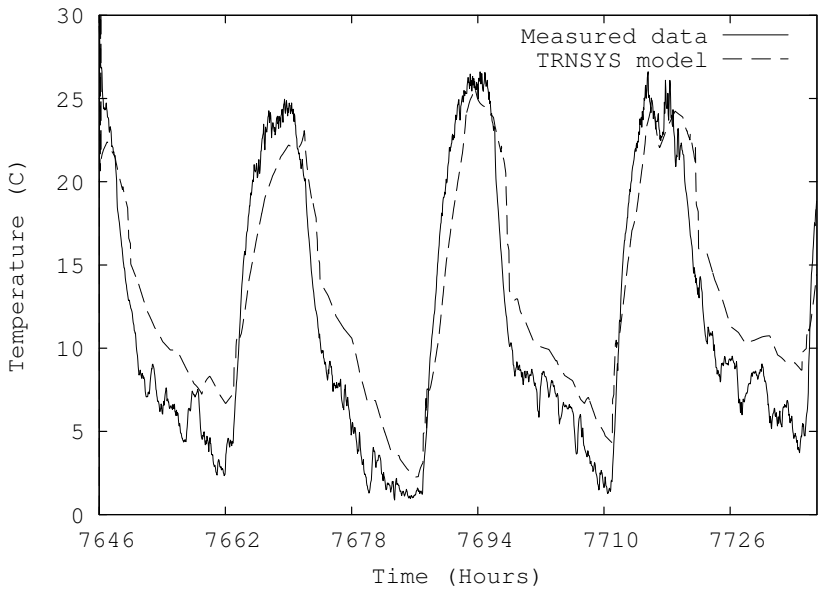

Figure 4.9: Temperature comparison for chrome double glazed collector outer glazing

During the course of this work, a new version of TRNSYS, with a more complete component library, became available. For example, TRNSYS 17 includes a model of a tank with wrap-around heat exchanger. This is the TYPE 1237 tank. The parameters of the TYPE 1237 are listed in Table 4.5. The predicted and measured temperatures of tank node 1, 6 and 8, located at the top of the tank, at the heat exchanger, and near the bottom of the tank respectively, are shown in Figure 4.11. When comparing measured water tank temperatures with those predicted by TRNSYS, As can be seen, the measured water tank temperatures are lower than the temperatures in the simulation model at night. This is probably due to incorrect parameter settings for the water tank, such as edge loss coefficient, bottom loss coefficient, etc. Overall the error is small and acceptable.

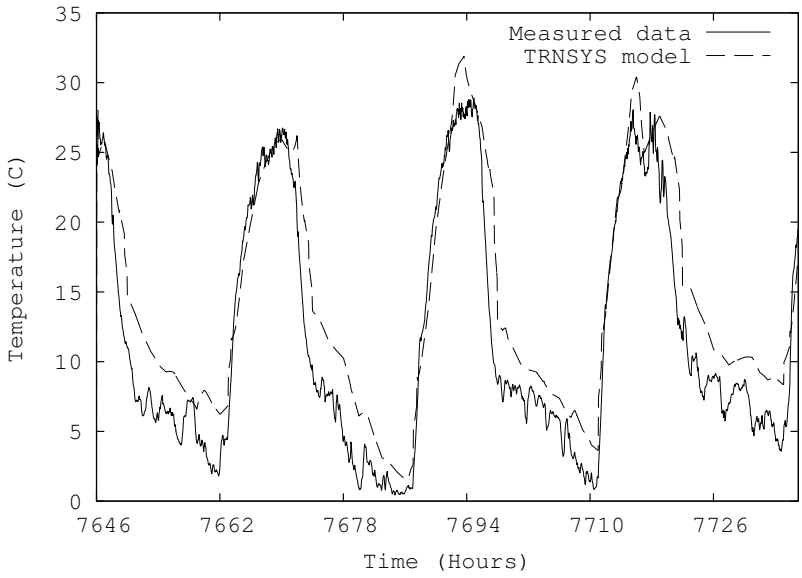

Figure 4.10: Temperature comparison for chrome single glazed collector glazing

Table 4.5: Water tank TYPE 1237 parameters

Parameter	**Value**	**Units**
Number of tank nodes	8	
Number of ports	1	
Number of miscellaneous heat flows	0	
Tank volume	0.297	m^3
Tank height	1.4	m
Fluid specific heat	4.19	$kJ/kg \cdot K$
Fluid density	1000	kg/m^3
Fluid thermal conductivity	2.14	$kJ/hr \cdot m \cdot K$
Fluid viscosity	3.21	$kg/m \cdot hr$
Fluid thermal expansion coefficient	0.00026	$1/K$
Top loss coefficient	2	$kJ/hr \cdot m^2 \cdot K$
Bottom loss coefficient	2	$kJ/hr \cdot m^2 \cdot K$
HX loss coefficient	2	$kJ/hr \cdot m^2 \cdot K$

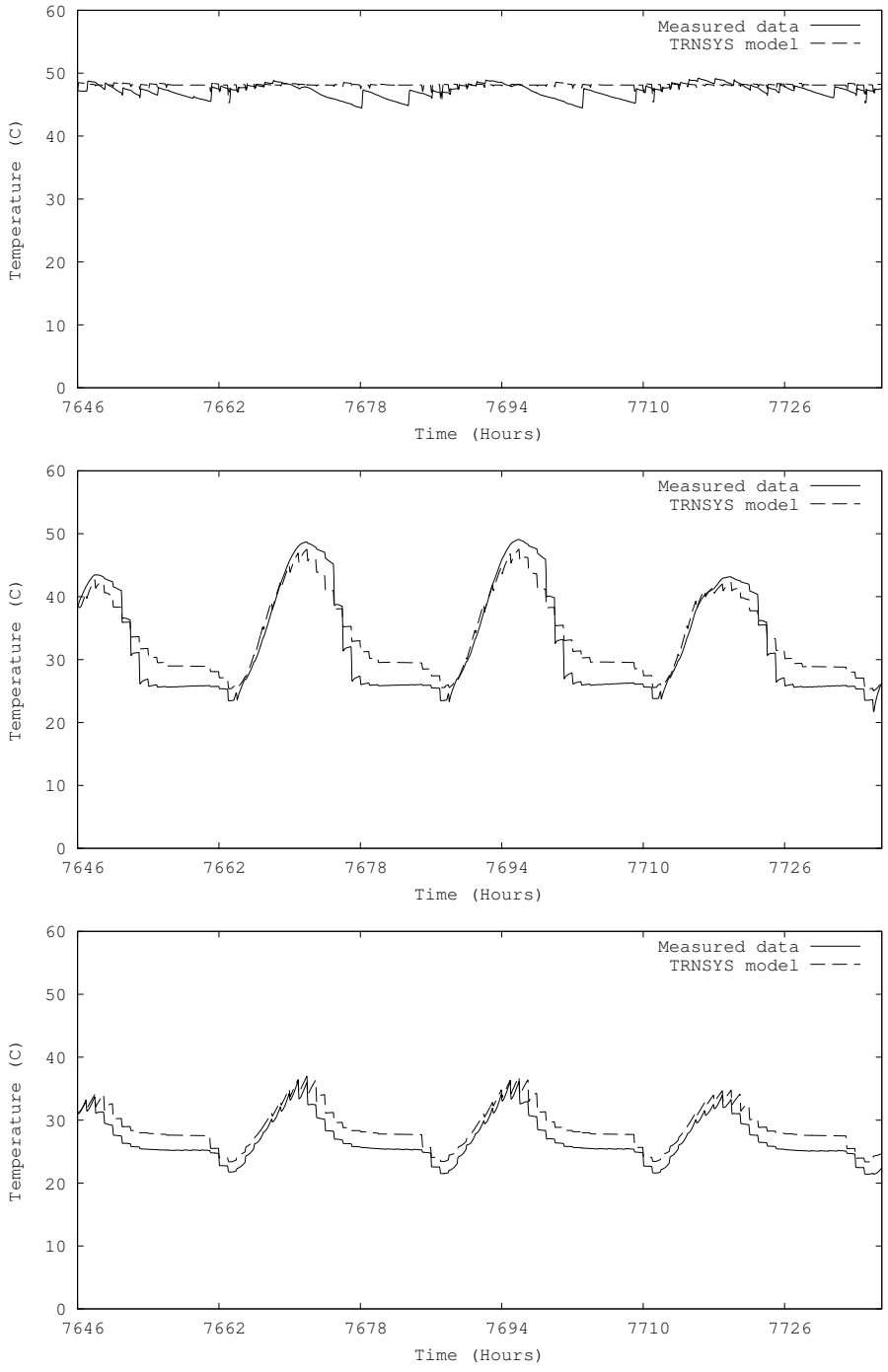

Figure 4.11: Temperatures of the tank at node 1, at the top of the tank, node6, near the heat exchanger, and at node 8, near the tank bottom.

Figure 4.12 presents a comparison of the predicted and measured useful energy gain by the single and double glazed collector. As can be seen, the simulation value is about 20% higher than the measured value. This is probably due to incorrect parameter settings for the collectors, such as incorrect absorptance of the plate, emissivity of the plate or the glass, etc.

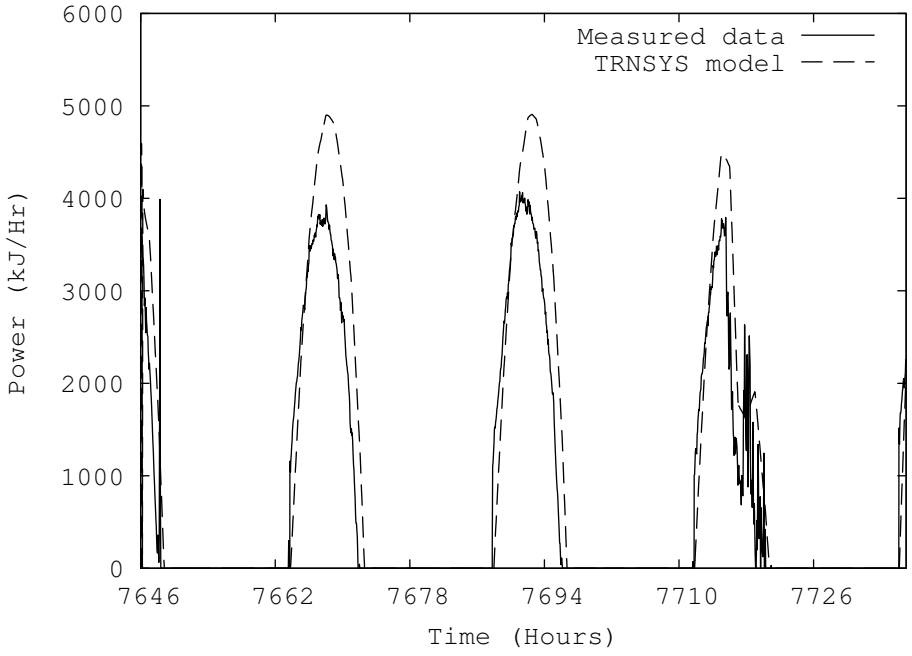

Figure 4.12: Useful energy gain by the collectors

4.4 Sensitivity analysis of the TRNSYS model

An analysis was conducted to determine the sensitivity of the TRNSYS model predictions to various collector parameters. This information is useful to determine which parameters are most important to the accuracy of the overall prediction of performance of the system which in turn is essential for accurate training of the fault detection system. The analysis focused on a painted-fin single glazed collector model.

Five parameters were considered for this analysis: plate emissivity, plate absorptance, glass emissivity, back heat loss coefficient or edge heat loss coefficient, and index of refraction of cover material. The plate emissivity, plate absorptance, glass emissivity, and index of refraction of cover material influence the solar irradiance transmission and absorptance by the plate and cover. The overall heat loss coefficient is the sum of the top, bottom and edge loss coefficient. The back and edge heat loss coefficient influence the overall heat loss by the collector to the environment. The influence of these five parameters on the collector plate temperature, collector glazing temperature and collector useful energy gain is shown in Figure 4.13, Figure 4.14, and Figure 4.15 respectively. As shown in these three figures, there are some dips caused by the hourly water draws.

Sensitivity studies were conducted for predictions of the plate temperature, which is a major driver of the glazing temperature. As shown in Figure 4.13, the plate temperature is most sensitive to plate absorptance. A change in the plate absorptance from 0.95 to 0.75 produces 7°C decreases in the plate peak temperatures. 0.75 and 0.95 are in normal range of the plate absorptance. A change in the back heat loss coefficients from $15kJ/hr{\cdot}m^2{\cdot}K$ to 5 $kJ/hr{\cdot}m^2{\cdot}K$ produces 4°C increases in the plate peak temperatures. The original edge heat loss coefficient is 18 $kJ/hr \cdot m^2 \cdot K$ and the top heat loss coefficient is calculated by empirical equations which are included in the Appendix C. The plate temperature is less affected by changes in the plate or glass emissivity, or the index of refraction of cover material.

As shown in Figure 4.14, the glazing temperature is most sensitive to the index of refraction of cover material and back or edge heat loss coefficient. A change in the plate absorptance from 0.95 to 0.75 produces 5°C decreases in the glazing peak temperatures. The glazing temperatures are only slightly affected by plate emissivity, glass emissivity, heat loss coefficients, or the index of refraction of cover material.

Figure 4.15 shows the results of the tests for the collector's useful energy gain

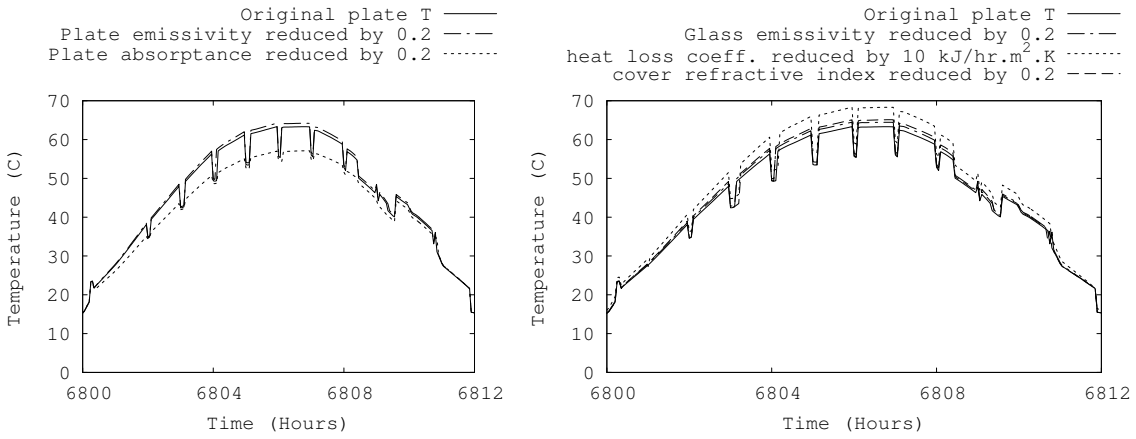

Figure 4.13: Collector plate temperature for various values. Dips are due to the hourly water draws.

and it shows that it is most sensitive to the plate absorptance and back heat loss coefficient. A change in the plate absorptance from 0.95 to 0.75 produces decreases in the predicted total energy gain by about 16%. A change in the back heat loss coefficients from $15 kJ/hr \cdot m^2 \cdot K$ to 5 $kJ/hr \cdot m^2 \cdot K$ produces about 12% increases in the total energy gain. The total energy gain are only slightly affected by plate or glass emissivity, or the index of refraction of cover material.

Figure 4.14: Collector glazing temperature for various values. Dips are due to the hourly water draws.

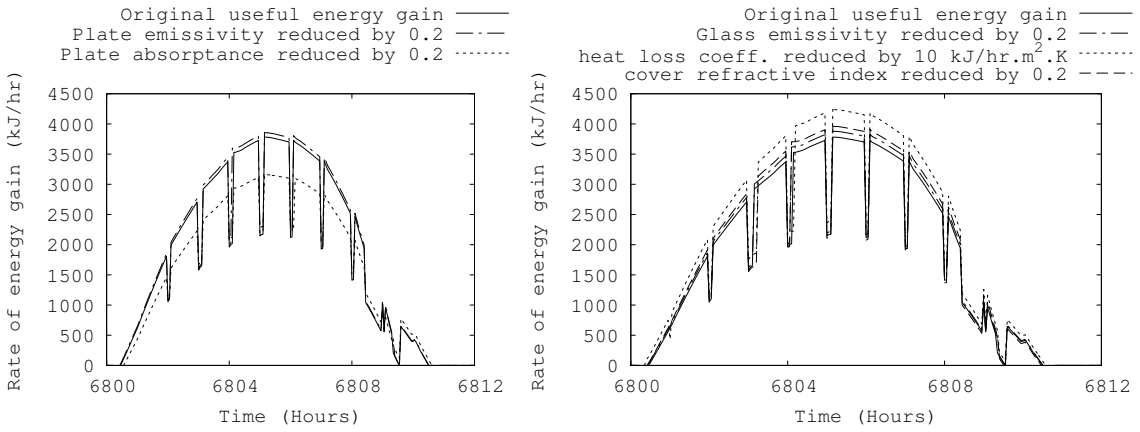

Figure 4.15: Collector useful energy gain for various values. Dips are due to the decrease of flow rate and the water tank outlet temperature during the hourly water draws.

Chapter 5

Application of Hierarchical ART

As noted earlier, while in principle the ART ANN could be trained in the field, this is generally impractical. In addition, faults should be detected from the first day of operation of a new system. In fact it is common for certain types of faults to occur early in the life of a system - a process known as infant mortality, which is still common in SHW systems as a result of the fact that each installation is in many ways a unique system. These early failures should be captured quickly and corrected, as they are critical to ensure that the system has a good chance of producing the benefits expected from its installation.

There are two time scales which dominate the operating conditions of a SHW system - daily variation and seasonal variation. Although in principle the ART network could learn to distinguish the two scales following the regular learning process, in this work the seasonal time scale was separated from the daily time scale by performing training on a monthly basis. The underlying assumption is that seasonal variation is small over the course of a month, and that daily patterns for a certain month should be classifiable, over the long term, as belonging to that month. The data used in the training are hourly historical weather data for Albuquerque, NM,

from which temperatures and irradiance measurements are extracted.

To train the ART ANN, the TRNSYS model is operated with the same control algorithm as the physical experiment, which in turn is representative of fielded systems. The time-step is three minutes, sufficiently small to capture parameter (collector plate temperature and water tank outlet temperature) variations resulting from system operation. For this particular case, the model is operated using a data set corresponding to the month of January for each of the years from 2000 to 2004.

The purpose of model-based training is to allow the long-term memory to form by exposing the network to a set of inputs large enough that exposure to additional inputs does not result in the creation of additional classes in the recognition layer. The size of the set of input patterns required to do this is a function of the vigilance parameter chosen. For the set of inputs composed of tank outlet temperature, plate temperature, derivative of plate temperature and time of day, the learning curve is shown in Figure 5.1. The rate of new class creation is high at early times, and asymptotes to zero for large enough times. The number of classes created at long times increases exponentially with vigilance parameter. The time for complete learning also increases with vigilance parameter, requiring just one January for $\rho = 0.5$, and five instances of January for $\rho = 0.8$.

The parameters used for training of the ART network are those commonly found in SHW systems, for example collector plate temperature and heat exchanger outlet temperature. An additional parameter generally available to a system controller is date and time. From these inputs, it is also possible to extract secondary parameters, for example the derivative of temperature. Finally, it is possible to provide as inputs a time history of temperatures, instead of only the current temperature. In principle, if the time derivative of temperature is important, the ART network should learn to compute this - given lengthier training.

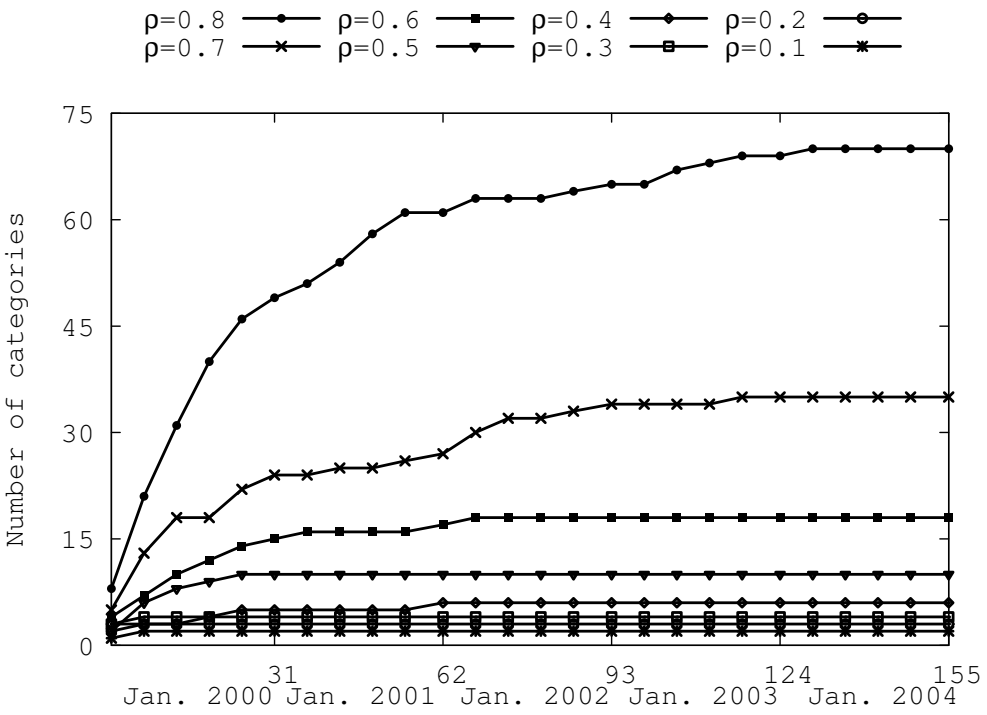

Figure 5.1: Generation of new classes as a function of time, for various vigilance parameter (ρ) values. The time required for full training increases with increasing vigilance parameter.

In this particular case, the inputs used are tank outlet temperature, plate temperature, derivative of plate temperature and time of day. When the number of inputs is greater then three, classes in the recognition layer are described by hyperboxes which are difficult to represent visually. To illustrate the creation of classes, the projection in the three dimensional space (formed by the inputs of plate temperature, time derivative of plate temperature, and time) of the hyperboxes is shown in Figure 5.2. Clustering of data belonging to individual classes (e.g. cloudy day vs. sunny day) is clearly visible.

Knowledge of the necessary time resolution of the training data is crucial for understanding the applicability of the ART ANN to real-time fault detection and

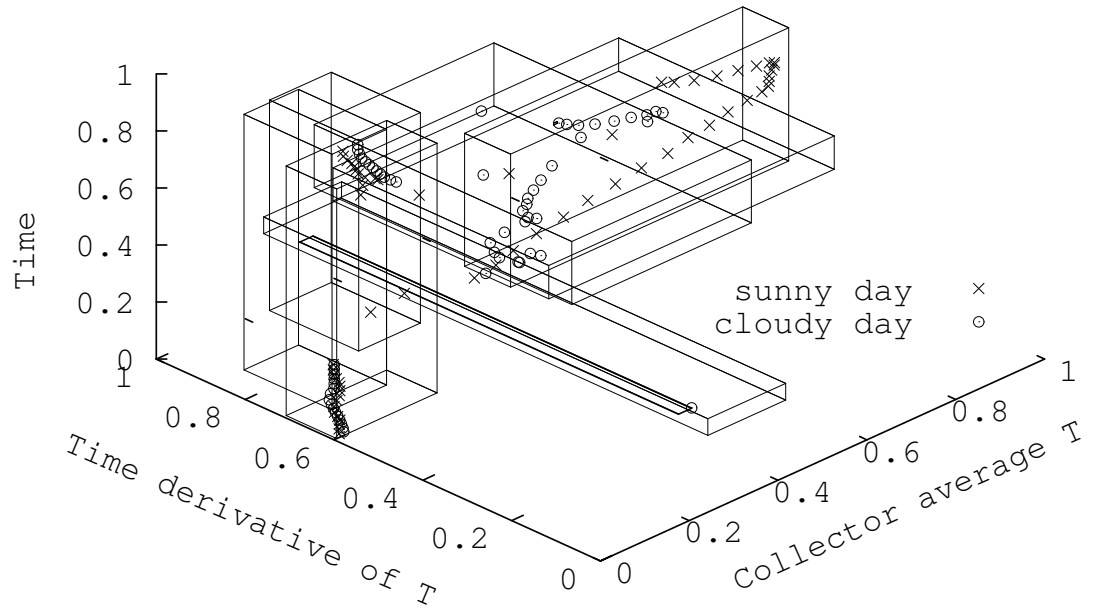

Figure 5.2: Three-dimensional projection of the class hyperboxes generated by training. New data points falling outside the boxes are categorized as a novelty and flagged.

monitoring. One important concern is that historical weather and solar data are generally available at relatively low resolution, typically one hour, while a fielded system may have access to data at one-second resolution. The obvious question that arises is whether training based on one-hour data is comparable to training based on one-second resolution, and if so, whether an ANN trained with one-hour data is capable of responding in real time to a fault, especially a high-severity one that may require fast response. To answer the first question, an ANN was trained under identical conditions, save for the training data. One set of training data was the standard one-hour weather data, the other was the same data with a superimposed noise signal. A typical day in the set is shown in Figure 5.3.

The learning curves which result from the smooth and noisy data sets are plotted in Figure 5.4, for a vigilance parameter of 0.8. The categorization process appears

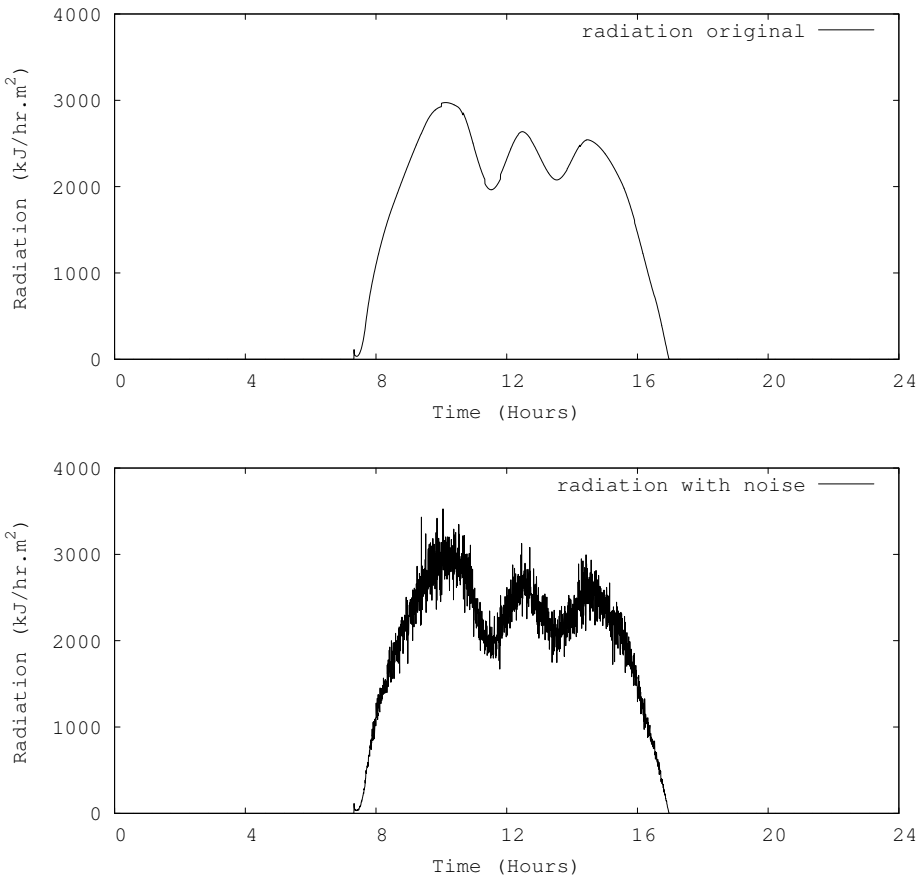

Figure 5.3: Typical hourly irradiance from historical weather data, and corresponding high-frequency data, simulating the passage of thin clouds between the sun and the solar collector. Note that noise was only added to data for cloudy days.

to be similar, with learning from the noisy data occurring slightly more slowly, but ultimately generating two more categories. However, the results seem to indicate that learning from smooth (hourly) data can be used to monitor events on the sub-hour scale.

To demonstrate that the input patterns can be classified into class hierarchies by the Hierarchical ART neural network, the following test is conducted using TRNSYS simulation data to train and test the Hierarchical ART neural network. The data set contains three kinds of failures of SHW systems: pump failure, degrada-

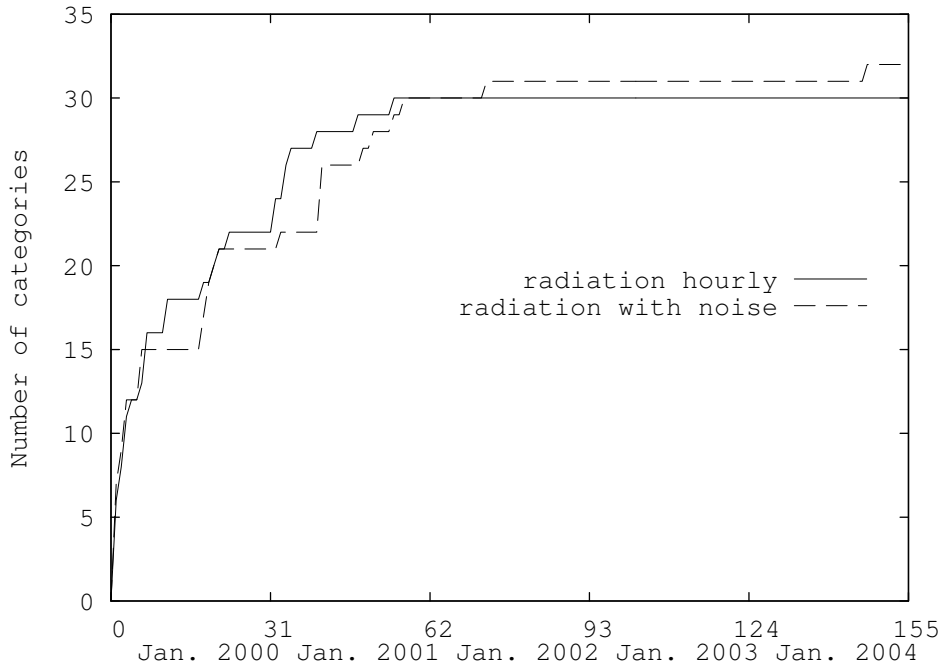

Figure 5.4: Generation of new categories as a function of time, for the smooth and noisy data sets. The vigilance parameter is ρ =0.8.

tion and thermosiphon. The input patterns of the neural network are collector plate mean temperature over a window of 12 minutes, variation of the collector plate temperature, time of day, and temperature difference between collector plate temperature and water tank outlet temperature. The input vectors are normalized and use complement coding which can represent both the presence and absence of features (Carpenter et al., 1991a). The neural network is trained off-line on a training set from a TRNSYS fault-free model. The test result shows in Figure 5.5. Major faults are detected near the root of the tree, while minor faults are nearer to the branch terminals.

A graphical user interface (GUI) is created for the hierarchical ART. The GUI provides an interface for the training module and testing module. In the training module (as shown in Figure 5.6), the user sets the choice parameter (which is defined

by equation 2.19), the number of layers, the vigilance parameters and the number of inputs, then chooses the training data file and runs the training process. When the training process is done, in the testing module (as shown in Figure 5.7), the user chooses the testing data file (i.e. the data from an experiment or from other TRNSYS simulation) and runs the test. In Figure 5.7, the x-axis is a temporal reference number, and the y-axis is the fault detection layer. The stars indicate faults detected by the ART system. Appendix B contains a listing of the GUI code.

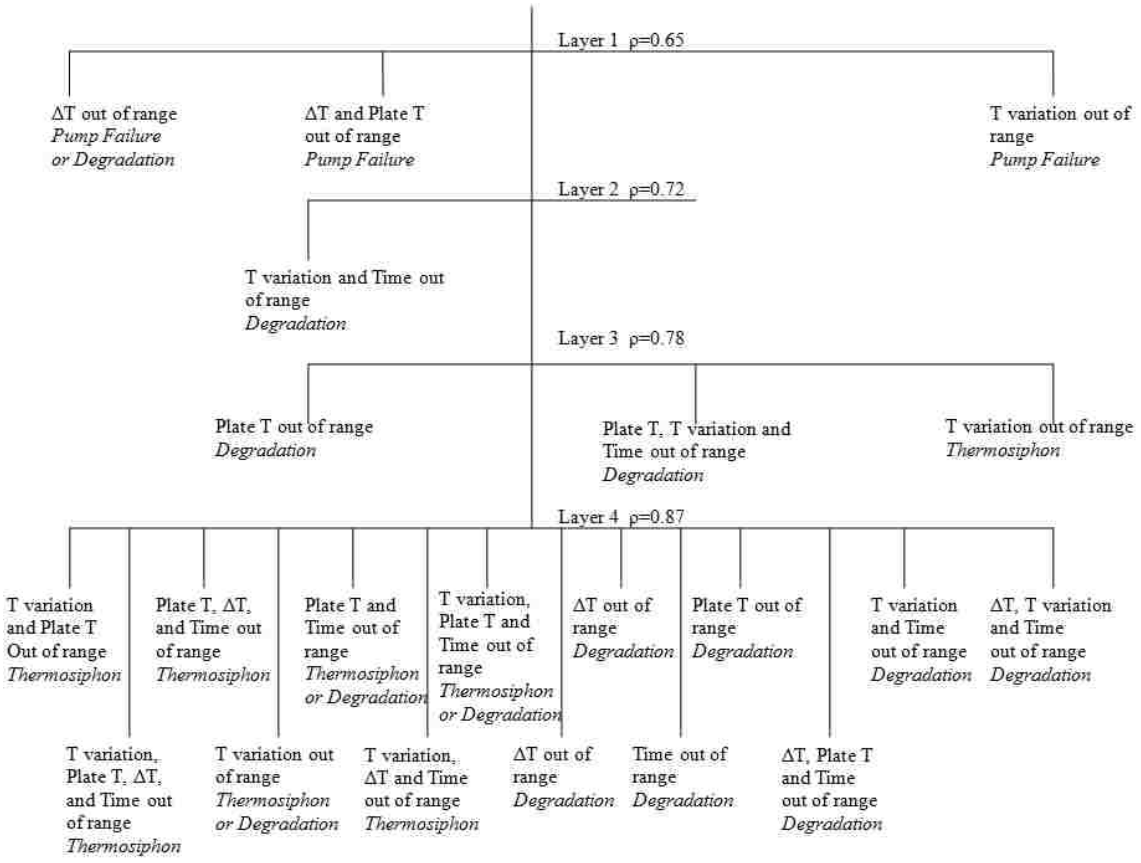

Figure 5.5: The hierarchy of SHW data sets generalized by a four layer Hierarchical ART neural network. The vigilance levels are $\rho_1 = 0.65, \rho_2 = 0.72, \rho_3 = 0.78, \rho_4 = 0.87$. In this example, 3, 1, 3 and 14 categories are created in layer 1, 2, 3 and 4 respectively.

Figure 5.6: The training module of the GUI of the hierarchical ART

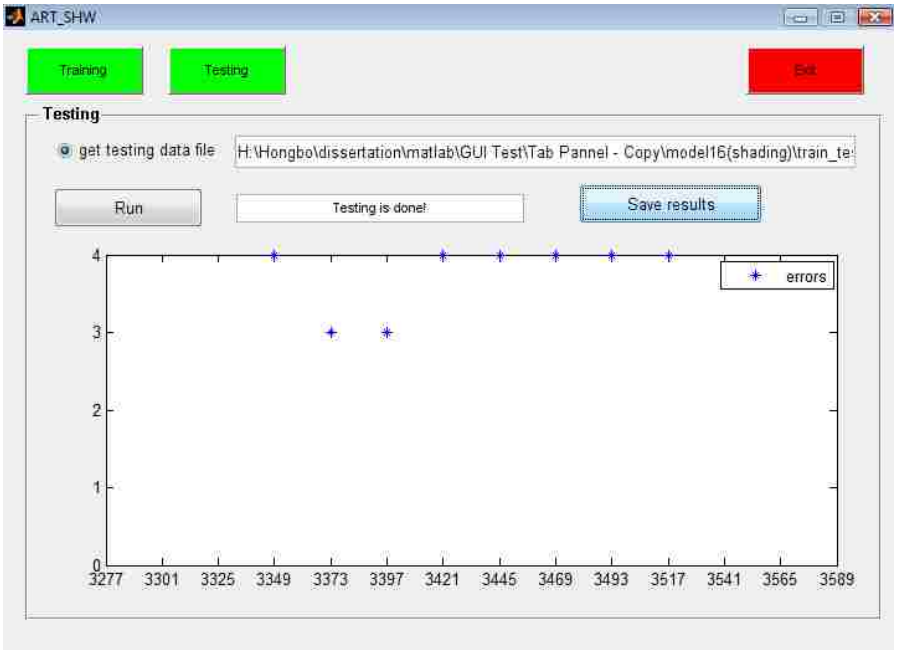

Figure 5.7: The testing module of the GUI of the hierarchical ART. Stars in the bottom plot represent the faults detected by the ART system.

5.1 Test A: Failed Solar Loop Pump

The application of Hierarchical ART technology began with simulated tests using a verified TRNSYS model. In the first test, called Simulated Test A, the model is used to simulate a failure. The model was run using a 3 minute time step over a simulation period of five days. These consecutive days were selected at random for a period in January, 2005. January was chosen because that was the month in which the ART had been trained and was also the time in which an actual test would be conducted. During the simulation run, the solar pump was shut down from 10am to 2pm on Friday, Saturday and Sunday. Thus 14,880 data points were collected, in which there are 1,053 time steps that reflected a failed pump, which stops pumping and completely fails. A severe fault, such as this one, should be detected quickly by the ART network. Evidence of this recognition should appear in the first or second level of the ART hierarchy.

With vigilance levels $\rho_1 = 0.58, \rho_2 = 0.68, \rho_3 = 0.74, \rho_4 = 0.8$, most of the 1,053 times that involved a broken pump caused ART to attempt to create new classes at the first layer, while only 26 of 13,827 normal data caused attempts to create new classes at any level. This corresponds to a detection rate of 99.8% for this simulated test and a false alarm rate of 0.2%. Figure 5.8 shows the test results on a Saturday in January 2005, in which all 81 data points corresponding to a broken pump, between the time 130 hours and 134 hours, resulted in an attempt to create new classes in the first layer, while 2 normal data points at 134.05 and 134.1 hours also attempt to create new classes in the first layer.

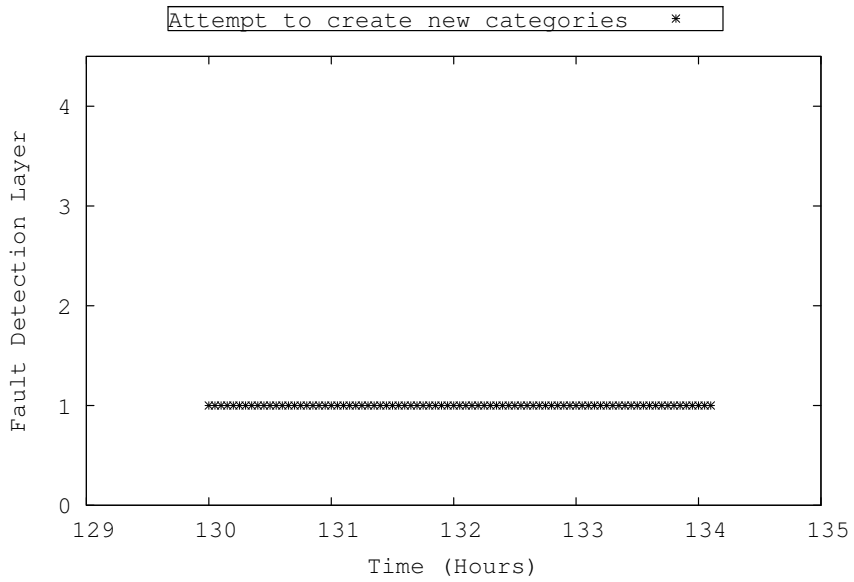

Figure 5.8: Pump failure happens between 130 and 134 hours

5.2 Test B: Impeller Degradation

Another test was designed to assess whether a slowly diminishing solar loop flow rate can be detected by the ART system. The flow rate might be reduced if the solar pump impeller is being peened by debris (such as a ball of solder) in the piping. A slowly diminishing flow rate might be a prelude to pump failure in which the pump degrades to a point that it can no longer overcome the pressure head in the loop and flow stops.

A test of this condition was simulated with the TRNSYS model in which the flow rate is progressively reduced from 1 to 0.8 in five weeks (1 the first week, 0.95 the second week and so on). The simulation was conducted from a three week period starting in January. Vigilance levels were the same as those in Test A. Starting from the third week when the flow rate reduced to 90%, new classes are created every day. As the flow rate continues to decrease, additional new classes are created in the

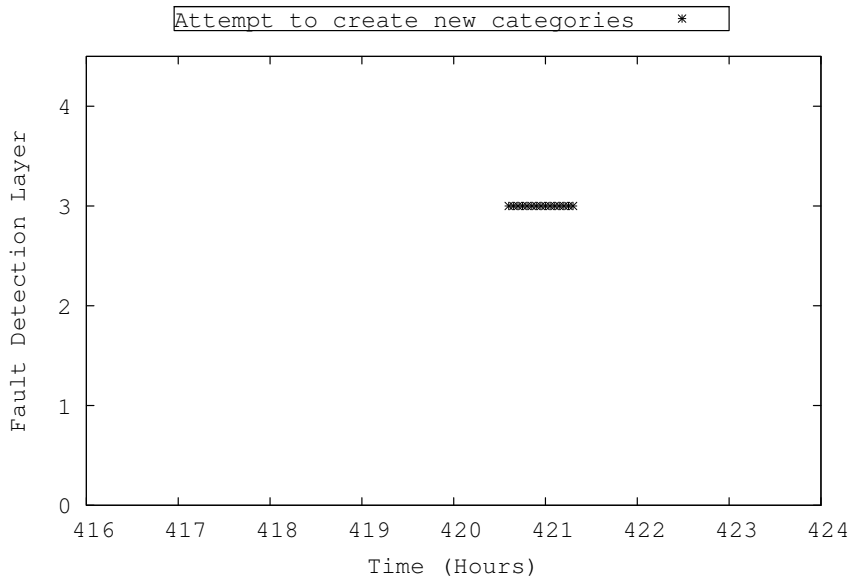

Figure 5.9: The impeller degradation can be detected when flow rate reduces to 90% of normal

daytime and more categories are shown in the first layer or the second layer, which has lower vigilance levels. The results for one day in the third week are shown in Figure 5.9. In that Figure, 15 of 480 data sets attempted to create new classes, most of which are in the second and third layer. These results mean that the impeller degradation can be detected as soon as the flow rate decreases to 90% of normal.

5.3 Test C: Thermosyphon

A third simulated test is for a mode of failure in which a faulty check-valve in the collector loop allows for nighttime reverse thermosyphoning. When the water temperature in the water tank is higher than the collector's temperature (usually at night), the water in the tank's heat exchanger rises to the top of the collector and

the cold water in the collector moves down to the heat exchanger through the non-operating pump. In this simulated test, vigilance levels are the same as that in Test A and B are applied. The simulated test starts on January 1^{st}, 2005. Figure 5.10 shows the results of this test for one day in January 2005. Starting from about 5pm, 62 data points create new classes in the third layer. Because the temperature difference between the collector and the water tank is still small, the flow rate is low. As a result, thermosyphoning has little influence on the input parameters that are used for the neural networks. However, from about 8pm to midnight, with the temperature difference increasing, 66 data points resulted in attempts to create new classes in the second layer. In addition, in the early morning 58 data points create new classes in the third layer and 14 data points create new classes in the fourth layer. Thus the ART system has the capability to detect this fault condition.

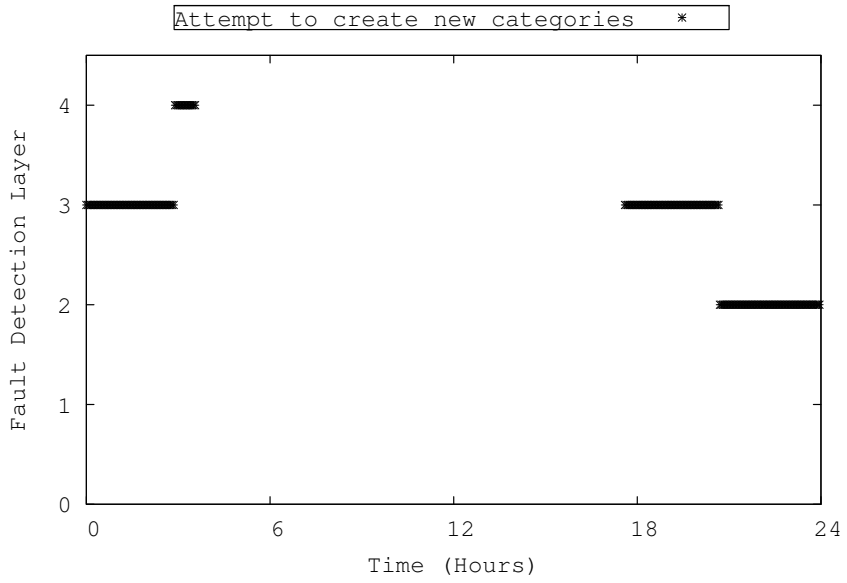

Figure 5.10: Thermosyphon happens late in the day and can be detected mostly in the first layer.

Chapter 6

Experimental results

After training the hierarchical ART neural network using a fault-free, verified TRNSYS model with 5 years of weather data (2000-2004), as described previously (He et al., 2011), a number of tests were performed to assess the ability of the network to detect various kinds of fault or degradation. The tests performed were: (1) pump failure; (2) pump degradation; and (3) shading. All the tests were conducted on the SHWRT.

6.1 Test A: pump fault

The pump failure test was conducted from January 7th, 2011 (5pm) through January 12th (10am). For the first 3 days, the system was operated normally. On the morning of the 4th day (January 11th) at about 8am, a solar loop pump failure which was similar to a real pump failure was simulated. The pump starts up normally but soon thereafter begins to pump erratically, then fails completely, replicating the condition where debris in the solar loop blocks the impeller. The temperature and flow rate conditions measured during the experiment are shown in Figure 6.1. The collector

plate temperature becomes unusually high due to lack of coolant flow, while the tank heat exchanger outlet temperature is lower than normal due to lack of solar heat. The flow rate is zero after pump failure.

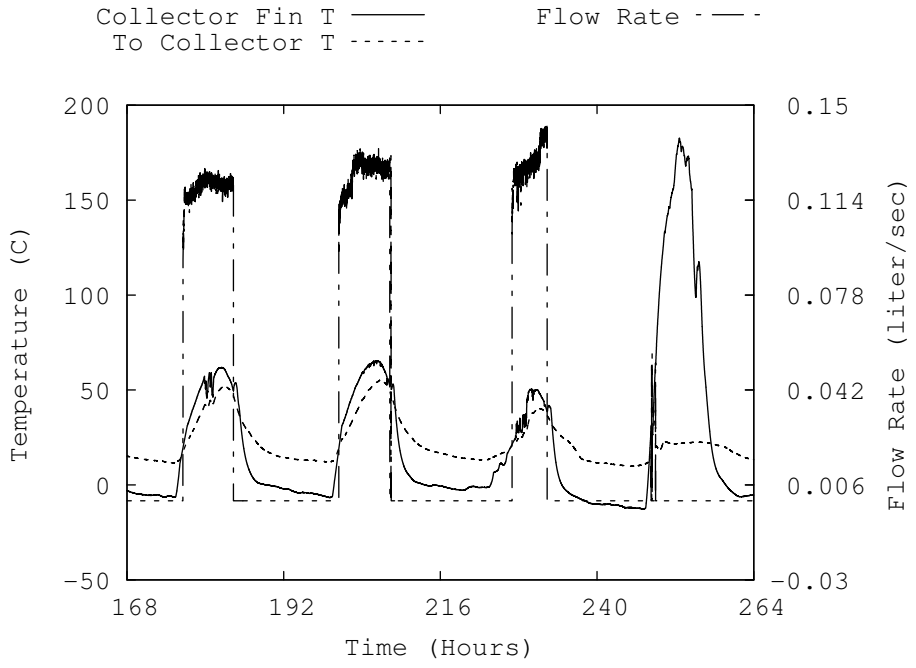

Figure 6.1: SHW system conditions prior to and after pump fault

As discussed previously, the input patterns to the neural network are collector plate mean temperature (averaged over the previous 12 minutes), time derivative of the collector plate temperature, time of day, and temperature difference between collector plate and water tank heat exchanger outlet temperature. As shown in Figure 6.2, the hierarchical ART detects the simulated pump failure very rapidly, initially at low-severity but quickly settling on a string of high-severity faults. It is also interesting to note that the detection of the pump failure simulated in the TRNSYS model followed exactly the same pattern as with the experimental results.

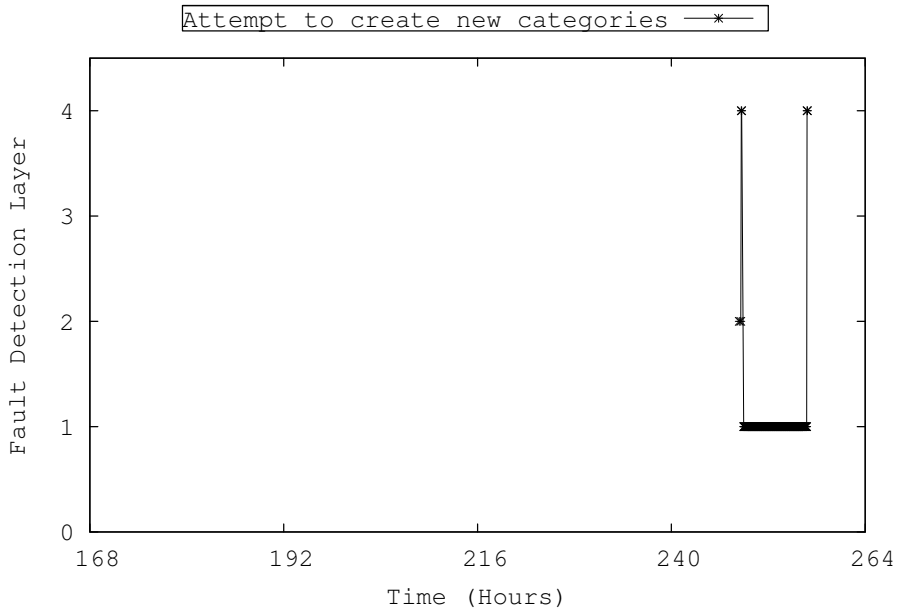

Figure 6.2: Attempts by the hierarchical ART network to create new classes, signifying novelty, at hour 247, when the pump fails. The severity is low initially, but quickly becomes high, signifying major deviation from normal conditions.

6.2 Test B: Impeller degradation

The pump degradation test was conducted by introducing an additional pressure drop in the flow loop, resulting in a reduction in flow rate. This test was conducted from February 24^{th}, through March 7^{th}, 2011. It simulates a condition in which debris in a solar loop, such as drops of solder introduced during construction, are slowly peening the edges of the impeller blades, compromising their ability to move water and resulting in a reduction in the flow rate over time. This process normally occurs over months of normal operation and continues until the impeller is incapable of overcoming the head in the supply piping upon startup (in a drainback system). In this case the process was speeded up to a four-day period due to constraints in the testing schedule.

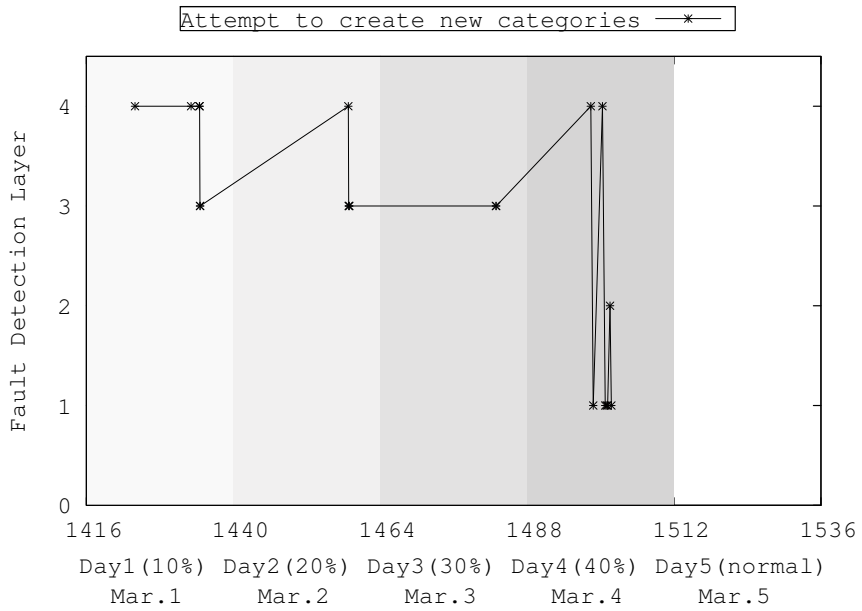

Figure 6.3: Attempts by the hierarchical ART network to create new classes, signifying novelty, during a test in which flow rate is gradually reduced over the course of four days. When flow rate decreases past a threshold between 30% and 40%, increasingly severe novelty is signaled, however even small changes in flow rate are detected early on, at appropriately low severity levels.

The SHWRT's SHW system operated normally for four days using the Solar Rating Certification Corporation's standard hot water draw profile (shown in Figure 3.4). On the fifth day, and for four consecutive days following, the solar loop flow was gradually reduced by around 10% per day. As described in the previous chapter, the solar loop flow was reduced by 5% per week in the degradation test with the TRNSYS simulation data. Regardless of the reduction rate, both the simulation and experimental results indicate that the ART system started to detect the faults when the flow rate reduced to 90%. The faults were detected at three levels (4, 3 and 2) in the simulated degradation test (shown in Figure 5.9). In contrast, the faults were detected only at at level 4 in the experiment. This difference is probably because the simulation test and the experiment test were run in different seasons and two

different sets of training data were used for the fault detection system.

Figure 6.3 shows the ART system's error reporting during the period in which the flow was being reduced. On the x-axis is a temporal reference number. The grey shaded boxes indicate each consecutive day. The black dots indicate the detection of an unusual condition. Note that the severity level of the error on y-axis is in reverse order, with a "1" representing the most severe unusual condition.

As can be seen, the ART system begins to detect unusual conditions as soon as the flow was reduced. By the fourth day, the system has escalated the level of severity significantly. When the flow was reset to its normal range, the number of detected unusual conditions dropped dramatically.

6.3 Test C: Shading

The primary cause of shading of residential solar collectors is trees. Because growth is gradual, it is difficult for the system owner to notice the problem, which could therefore extend over years of operation. Correspondingly, it is also difficult for a fault detection system to notice a problem -shading caused by a tree could easily fall in the same class as shading caused by clouds. To assess the capacity of the system to detect shading, the collectors were partially covered with movable panels, which were shifted through the day to mimic shading caused by a tree, as shown in Figure 6.4.

Two sets of detection results for the shading test are shown in Figure 6.5. One uses the same inputs as the previous tests, namely plate temperature, time derivative of plate temperature, and tank heat exchanger outlet temperature. The other uses only two inputs, the standard deviation of the plate temperature over the last five days at 1 PM, and the standard deviation of the same. The vigilance levels for the

two tests are, respectively, 0.58, 0.68, 0.74, 0.8 and 0.4, 0.45, 0.5, 0.55.

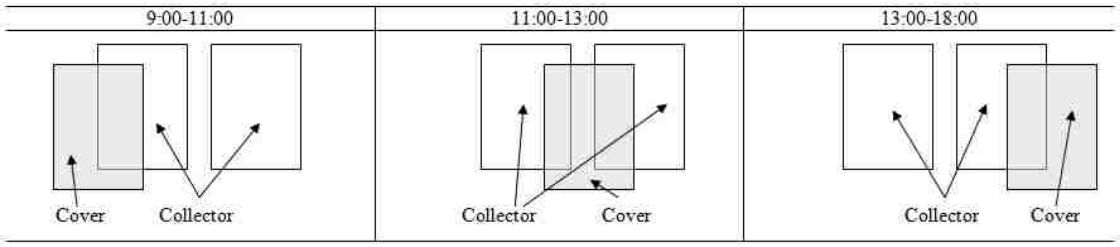

Figure 6.4: Simulated shading of collectors, obtained by moving a cover repeatedly through the day.

For both input combinations, the ART network detects shading. However, detection of shading with the 'standard' inputs required a recalibration of the vigilance parameters, to an extent where cloudy weather is also mistakenly signaled as a potential fault. On the other hand, the five-day mean and standard deviation inputs successfully detect shading, but do not confuse it with cloudy weather. The reason for the better performance of the second set of inputs is simple. The first set of inputs allows recognition of patterns which are collected for short periods (12 minutes). Over this length of time, shading could look exactly the same as a cloud, which could also reduce total heat collection for a similar amount of time. Unlike other faults, shading produces patterns that could match existing categories - the difference being that the faults are replicated at the same time every day. Using inputs that average sensor information over the course of several days allows the detection of regular faults which have a different signature from semi-random (weather) events over the long term, such as shading, very dirty or broken glazing, loss of coolant.

Therefore, a novelty detection system should be built with a dual rule - one which detects faults visible over short times, and one which looks at long-term variability. The two systems could run in parallel.

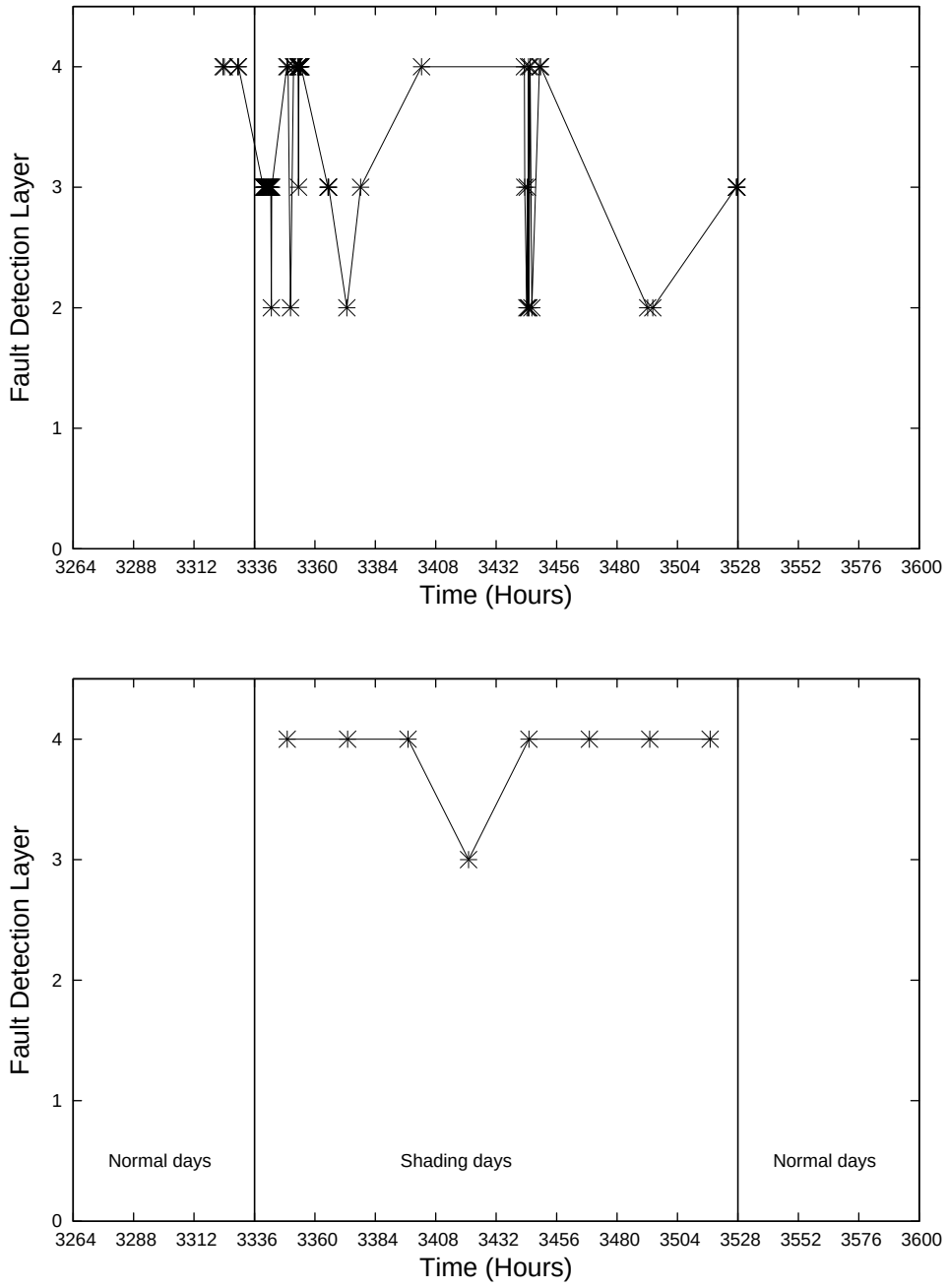

Figure 6.5: Shading fault detection performance for regular inputs, and average and standard deviation of plate temperature.

Chapter 7

Conclusions and future work

A fault detection system for SHW systems is presented in this work. This fault detection system has several advantages. First, the input patterns of the fault detection system are generated from two sensors: collector plate temperature and water tank heat exchanger outlet temperature, which are normally installed in residential SHW systems installed by commercial operators. Second, the fault detection system can be developed based only on simulation fault-free data or historical fault-free measurement data, and no faulty training data are required. Third, various types of fault and performance degradation have been detected successfully and faults are detected at various levels depending on their severity. Fourth, the fault detection system has high computational efficiency and can be implemented in on-line system monitoring.

The ART-based fault detection system is developed and tested in Albuquerque, NM, future research should focus on enhancing the capabilities of fault diagnostics. Conclusions and some possible future for this work are presented in this chapter.

Through various examples, the viability of using an autonomous system which is able to detect degradation and failure of a solar hot water heating system is demon-

strated. The viability of such a system has many contributing factors, including performance, the potential for low-cost implementation, the capacity to adapt to various configurations, and robustness. From the point of view of performance, it was shown, both in numerical simulation and in experiment, that faults of all types, from fatal ones which need immediate attention to ones that are in early stages of development and are not immediately visible even to an expert human operator are detected successfully.

By using an adaptive system, there is no need to anticipate all the possible failures, or combinations thereof, which could occur. It is enough to train the ART network from model results, without knowing much detail about the physics of the process. While this is an attractive feature, there are two potential pitfalls. One is that building an accurate model of a solar system in TRNSYS is not trivial, and requires a considerable amount of expertise. However, it is theoretically possible to automate this process, once sources of data for the system components is identified. The other is that the required fidelity of the model is at this point unknown. One positive indication is that training outcomes with one-hour data files differed very little from training done using simulated one-second data files. However, it will be necessary to identify which model parameters must be specified with high precision, and which ones have little effect.

One of the strengths of the system is that only few data streams are needed, meaning that it will not be necessary to instrument solar hot water systems with additional sensors, something which would not be acceptable in an aggressively competitive industry where reducing costs is paramount. On the other hand, it will be necessary to conduct further research in identifying the optimal inputs to provide to the ART network. For example, it was seen in this research that inputs which work very well for identifying a certain class of faults, having to do with the circulation of the heat medium, did not allow the ART network to successfully identify faults

connected with the ability of the system to absorb radiation (i.e. shading). One way to overcome this difficulty would be to train parallel ARTs, each used to detect different kinds of faults. Alternatively, a larger set of inputs could be chosen to train a single ART, which would then produce a set of higher-dimensional classes of normal operating conditions.

Finally, it was shown how a hierarchical ART architecture would provide a number of useful features, including a higher classification efficiency, as well as an indication of the level of severity of the problem. Research is ongoing towards resolving some of the issues noted here, but I am confident that ART-based anomaly detection will find application in SHW system monitoring services, as well as, eventually, in larger, more complex systems such as commercial building HVAC systems or subsystems.

In the future I plan to make advances in the steps to commercialization of the ART technology, based on the needs of potential customers who may be interested in this technology. In particular, I will investigate streamlining of the modeling and training process to make it possible for non-experts to conduct the task. I will also investigate the availability of information to allow this streamlining. Finally, I will examine the computational complexity of possible implementations of the technology, either in a distributed fashion on locally processing devices, or in a centralized fashion. In particular, I have the following plans:

1. **False alarm rate analysis.** As discussed in Chapter 5, Figure 5.1 shows the relationship between the number of classes and the training samples for different vigilance parameters. From this figure, we can see that the vigilance parameter levels out sooner with low vigilance values and the number of training samples required for the training process can be predicted from these curves. The number of training samples can also be determined by the analysis of false alarm rate for different vigilance parameters. If a novelty is due to normal

variation, it is a false alarm. The false alarm rate will level out with increasing training samples. This false alarm rate may level out sooner with low vigilance values. High vigilance parameter may get a higher reasoning false alarm rate value, which means the false alarm rate will convergent to a high value with increasing training data and more novelties are due to normal variation. Low false alarm rate is important for a robust fault detection method. The false alarm rate can be used as a sensible criterion for choosing number of training samples and vigilance values.

2. **Optimization of vigilance parameters.** The prediction of alarms is dependent on the vigilance parameters. Alarms can be detected at varying levels of severity by a suitable set of vigilance parameters. As discussed in Chapter 6, for the shading test some false alarms are created by the ART system with the same vigilance parameters. It is difficult to choose a set of vigilance parameters which works for all the tests. Users of the ART fault detection are not expected to have the technical knowledge about the ART network. I will develop an advanced ART network which can choose the vigilance parameters automatically. Genetic algorithms (GAs) have the ability of global searching in a parallel manner based on the mechanics of natural selection and natural genetics, and many researchers combine the ART networks with GAs to improve the performance of ART networks. For example, when GAs are used, the dimension of the ART network input space can be reduced, which in turn reduce the training time was reduced (Punitha et al., 2007). Kaylani et al. (2007) used GAs to solve the category proliferation problem in ART. One application of GAs in the current project will be using GAs to select a set of appropriate vigilance parameters for the ART-based fault detection. The normal and faulty data are generated from simulation models or historical measured data. Then the vigilance parameters will be chosen by training the GAs with these normal and faulty data. The generalization of ART will be improved and the ART-based fault detection will

be able to apply to more complex, larger distributed energy systems.

3. **Sensitivity analysis for model detail.** I have, to this point, made a model that, to the best of my capacity, is an accurate reflection of the real system. In a commercial situation, a vendor will have a good idea of the equipment installed, but will not have a detailed knowledge of how to model the systems. The model will then need to be generated automatically. The question is, how detailed does the model need to be to provide adequate training? I will change the level of refinement of the TRNSYS models that are used for training, and then verify the ability of the trained ART network to detect novelty, including detection at the appropriate severity level.

4. **Sensitivity to inputs.** So far, I have used inputs typical of domestic SHW systems, including a sensor mounted on the collector and one mounted at the storage water tank. Operators of monitoring systems, who are our most likely customers, may want to monitor parameters that are different from the ones in my study, for example ones mounted exclusively near the storage tank, to avoid costly rooftop installation. I need to understand if the ART networks can work in a satisfactory manner with a variety of inputs, other than the ones I have successfully used so far. I will test use of a variety of inputs for the ART network, and select the ones which work best while satisfying constraints that may be imposed by service providers.

5. **Availability of information.** After investigating the appropriate level of detail that should be used in the model for training purposes, I need to determine how much of this information is available from existing sources, for example the SRCC (Solar Rating and Certification Corporation). The idea is that a vendor, a homeowner or a service provider will go to a web site to build their system, with the ability to choose individual components from a predefined menu. Components in the menus will need to have the necessary

information to build the TRNSYS model, including parameters such as collector net absorber area, surface radiant characteristics, heat exchanger type and dimensions, etc. Other parameters will need to be measured directly, for example the tilt and orientation of the collector, the length of pipes connecting components, etc. The system will also need to be able to assign reasonable default values in case the information is not readily available.

6. **Computational complexity and space complexity of the ART-based neural networks analysis.** I will analyze the numbers of allowable categories which are created during the training process and the runtime for a single epoch with the training data set. Because of the high computational efficiency of the ART-based neural networks, the fault detection system can run as a real-time monitor system.

The above recommendations will improve the accuracy and robustness of the fault detection system. This dissertation research is an important early step for commercialization the ART-based fault detection system.

Appendices

Appendix A

Pump controller Matlab code

```
% SolarCollector.m
% ------------------------------------------------------------------------------------------------------------------------
%
% Simple first-order solar collector model (M-file called by TRNSYS type 155)
%
% Data passed from / to TRNSYS
% ----------------------------
%
% trnTime (1x1)        : simulation time
% trnInfo (15x1)       : TRNSYS info array
% trnInputs (nIx1)     : TRNSYS inputs
% trnStartTime (1x1)   : TRNSYS Simulation Start time
% trnStopTime (1x1)    : TRNSYS Simulation Stop time
% trnTimeStep (1x1)    : TRNSYS Simulation time step
% mFileErrorCode (1x1) : Error code for this m-file. It is set to 1 by TRNSYS and the m-file should set it to 0 at the
%                        end to indicate that the call was successful. Any non-zero value will stop the simulation
% trnOutputs (nOx1)    : TRNSYS outputs
%
%
% Notes:
% ------
%
% You can use the values of trnInfo(7), trnInfo(8) and trnInfo(13) to identify the call (e.g. first iteration, etc.)
% Real-time controllers (callingMode = 10) will only be called once per time step with trnInfo(13) = 1 (after convergence)
%
% The number of inputs is given by the size of trnInputs and by trnInfo(3)
% The number of expected outputs is given by trnInfo(6)
% ------------------------------------------------------------------------------------------------------------------------
% This example implements a very simple solar collector model. The component is iterative (should be called at each
% TRNSYS call)
%
% trnInputs
% ---------
%
```

```
% trnInputs(1) : T, collector outlet temperature
% trnInputs(2) : T, water tank top temperature
% trnInputs(3) : T, water tank bottom temperature
% trnInputs(4) : forcing function
% trnInputs(5) : T, outside ambient temperature
%
% trnOutputs
%
% trnOutputs(1) : pump on/off signal
% trnOutputs(2) : sky temperature
%
% MKu, October 2004
% ------------------------------------------------------------------------------------------------------------------------


% TRNSYS sets mFileErrorCode = 1 at the beginning of the M-File for error detection
% This file increments mFileErrorCode at different places. If an error occurs in the m-file the last succesful step will
% be indicated by mFileErrorCode, which is displayed in the TRNSYS error message
% At the very end, the m-file sets mFileErrorCode to 0 to indicate that everything was OK

mFileErrorCode = 100    % Beginning of the m-file

% --- Solar collector parameters----------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------------------------------

mFileErrorCode = 110    % After setting parameters


% --- Process Inputs ---------------------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------------------------------

T_coll_out = trnInputs(1);
Tank_top_T = trnInputs(2);
Tank_bot_T = trnInputs(3);
Forcing_func = trnInputs(4);
Ambient_T = trnInputs(5);
T_dp=5;

mFileErrorCode = 120    % After processing inputs


% --- First call of the simulation: initial time step (no iterations) --------------------------------------------------
% ------------------------------------------------------------------------------------------------------------------------
% (note that Matlab is initialized before this at the info(7) = -1 call, but the m-file is not called)

if ( (trnInfo(7) == 0) & (trnTime-trnStartTime < 1e-6) )

    % This is the first call (Counter will be incremented later for this very first call)
    nCall = 0;

    % This is the first time step
    nStep = 1;

    % Initialize history of the variables for plotting at the end of the simulation
    nTimeSteps = (trnStopTime-trnStartTime)/trnTimeStep + 1;
    history.onoff = zeros(nTimeSteps,1);
    history.en = zeros(nTimeSteps,1);
```

```
    % No return, we will calculate the solar collector performance during this call
    mFileErrorCode = 130    % After initialization

end


% --- Very last call of the simulation (after the user clicks "OK"): Do nothing ---------------------------------------
% ---------------------------------------------------------------------------------------------------------------------

if ( trnInfo(8) == -1 )

    mFileErrorCode = 1000;

    mFileErrorCode = 0; % Tell TRNSYS that we reached the end of the m-file without errors
    return

end


% --- Post convergence calls: store values ------------------------------------------------------------------------
% ---------------------------------------------------------------------------------------------------------------------

if (trnInfo(13) == 1)

    mFileErrorCode = 140;   % Beginning of a post-convergence call

    history.onoff(nStep) = on_pump;
    history.en(nStep) = en_pump;
%     history.func(nStep) = Forcing_func;

    mFileErrorCode = 0; % Tell TRNSYS that we reached the end of the m-file without errors
    return  % Do not update outputs at this call

end


% --- All iterative calls -----------------------------------------------------------------------------------------
% ---------------------------------------------------------------------------------------------------------------------

% --- If this is a first call in the time step, increment counter ---

if ( trnInfo(7) == 0 )
    nStep = nStep+1;
end

% --- Get TRNSYS Inputs ---

nI = trnInfo(3);     % For bookkeeping
nO = trnInfo(6);   % For bookkeeping

T_coll_out = trnInputs(1);
Tank_top_T = trnInputs(2);
Tank_bot_T = trnInputs(3);
Forcing_func = trnInputs(4);
Ambient_T = trnInputs(5);

mFileErrorCode = 150;   % After reading inputs
```

```
% --- Calculate solar collector performance ---
dis_pump = 0;
en_pump = 1;
on_pump = 1;

if Tank_bot_T > 60
    dis_pump == 1;
else
    dis_pump == 0;
end

if T_coll_out > 97
    dis_pump == 1;
else
    dis_pump == 0;
end

if nStep == 1
    if (Tank_bot_T < 55) && (T_coll_out < 97)
        en_pump = 1;
    else
        en_pump =0;
    end
end

if en_pump == 1
    if (T_coll_out - Tank_bot_T) >7
        on_pump = 1*Forcing_func;
    else
        on_pump = 0;
    end
else
    on_pump = 0;
end

if history.onoff(nStep-1) > 0
    if (T_coll_out-Tank_bot_T) > 0
        on_pump = 1*Forcing_func;
    else
        on_pump = 0;
    end
end

sky_T = 0.0552*(Ambient_T+273.15)^1.5-273.15;

% --- Set outputs ---
trnOutputs(1) = on_pump;
trnOutputs(2) = sky_T;

mFileErrorCode = 0; % Tell TRNSYS that we reached the end of the m-file without errors
return
```

Appendix B

Hierarchical ART GUI Code

B.1 Graphical design part

```
function varargout = ART_SHW(varargin)
% ART_SHW M-file for ART_SHW.fig
%      ART_SHW, by itself, creates text1 new ART_SHW or raises the existing
%      singleton*.
%
%      H = ART_SHW returns the handle to text1 new ART_SHW or the handle to
%      the existing singleton*.
%
%      ART_SHW('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ART_SHW.M with the given input arguments.
%
%      ART_SHW('Property','Value',...) creates text1 new ART_SHW or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before scoutdetails_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to ART_SHW_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help ART_SHW

% Last Modified by GUIDE v2.5 02-May-2011 15:10:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ART_SHW_OpeningFcn, ...
                   'gui_OutputFcn',  @ART_SHW_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ART_SHW is made visible.
function ART_SHW_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ART_SHW (see VARARGIN)


% Choose default command line output for ART_SHW
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ART_SHW wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Plot helicopter pictures on startup
% scout = imread('scout.bmp');
% axes(handles.axes1);
% image(scout);
% axis off;
%
% yaw_pedals = imread('Yaw Pedals.jpg');
% axes(handles.axes3);
% image(yaw_pedals);
% axis off;
%
% collective_lever = imread('Collective Stick.jpg');
% axes(handles.axes4);
% image(collective_lever);
% axis off;
%
% cyclic_stick = imread('Cockpit Instruments.jpg');
% axes(handles.axes5);
% image(cyclic_stick);
% axis off;

% --- Outputs from this function are returned to the command line.
```

```
function varargout = ART_SHW_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%Scout Details
function tab1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Control Details
function tab2_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Control Details
function tab3_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in exit.
function exit_Callback(hObject, eventdata, handles)
% hObject    handle to exit (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close ART_SHW;

% --- Executes on button press in train.
function train_Callback(hObject, eventdata, handles)
% hObject    handle to train (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles = guidata(ART_SHW);

set(handles.tab1,'Visible','on');
set(handles.tab2,'Visible','off');
% set(handles.tab3,'Visible','off');



% --- Executes on button press in test.
function test_Callback(hObject, eventdata, handles)
% hObject    handle to test (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles = guidata(ART_SHW);

set(handles.tab1,'Visible','off');
```

```
set(handles.tab2,'Visible','on');
% set(handles.tab3,'Visible','off');

% --- Executes on button press in result.
% function result_Callback(hObject, eventdata, handles)
% % hObject    handle to result (see GCBO)
% % eventdata  reserved - to be defined in text1 future version of MATLAB
% % handles    structure with handles and user data (see GUIDATA)
%
% set(handles.tab1,'Visible','off');
% set(handles.tab2,'Visible','off');
% set(handles.tab3,'Visible','on');



% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)




function text1_Callback(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text1 as text
%        str2double(get(hObject,'String')) returns contents of text1 as text1 double


% --- Executes during object creation, after setting all properties.
function text1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have text1 white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function text2_Callback(hObject, eventdata, handles)
% hObject    handle to text2 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text2 as text
%        str2double(get(hObject,'String')) returns contents of text2 as text1 double


% --- Executes during object creation, after setting all properties.
```

```
function text2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text2 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have text1 white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function text3_Callback(hObject, eventdata, handles)
% hObject    handle to text3 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text3 as text
%        str2double(get(hObject,'String')) returns contents of text3 as text1 double


% --- Executes during object creation, after setting all properties.
function text3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text3 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have text1 white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in runtrain.
function runtrain_Callback(hObject, eventdata, handles)
% hObject    handle to runtrain (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

clear out_results wij01 wij12 wij23 wij34

global max0 min0 CAT01 CAT12 CAT23 CAT34 wij01 wij12 wij23 wij34 F1 alpha rho Nolayer rho

str1=get(handles.text1,'string');
str2=get(handles.text2,'string');
str3=get(handles.text3,'string');
str4=get(handles.text4,'string');
str5=get(handles.text5,'string');
str6=get(handles.text6,'string');
str7=get(handles.text7,'string');
str8=get(handles.text8,'string');

% vala=str2num(stra);
% valb=str2num(strb);
% valc=vala+valb;
% set(handles.text3,'string',num2str(valc))
```

```
alpha=str2num(str1);
Nolayer=str2num(str2);
% alpha=1e-5;            %choice parameter
rho(1)=str2num(str3);
rho(2)=str2num(str4);
rho(3)=str2num(str5);
rho(4)=str2num(str6);
F1=2*str2num(str7);
% rho=[0.65 0.72 0.78 0.87];

% daystep=480;
% daynum=0;
% daymonth=31;

% [node(:,1),node(:,2),node(:,3)]= textread('train1week.txt','%f %f %f');

% [node(:,1),node(:,2),node(:,3)]=textread(str8,'%f %f %f');

if F1/2==1
    [node(:,1),node(:,2)]=textread(str8,'%f %f');
elseif F1/2==2
    [node(:,1),node(:,2),node(:,3)]=textread(str8,'%f %f %f');
elseif F1/2==3
    [node(:,1),node(:,2),node(:,3),node(:,4)]=textread(str8,'%f %f %f %f');
elseif F1/2==4
    [node(:,1),node(:,2),node(:,3),node(:,4),node(:,5)]=textread(str8,'%f %f %f %f %f');
elseif F1/2==5
    [node(:,1),node(:,2),node(:,3),node(:,4),node(:,5),node(:,6)]=textread(str8,'%f %f %f %f %f %f');
elseif F1/2==6
    [node(:,1),node(:,2),node(:,3),node(:,4),node(:,5),node(:,6),node(:,7)]=textread(str8,'%f %f %f %f %f %f %f');
elseif F1/2==7
    [node(:,1),node(:,2),node(:,3),node(:,4),node(:,5),node(:,6),node(:,7),node(:,8)]=textread(str8,'%f %f %f %f %f %f %f %f');
elseif F1/2==8
   [node(:,1),node(:,2),node(:,3),node(:,4),node(:,5),node(:,6),node(:,7),node(:,8),node(:,9)]=
            textread(str8,'%f %f %f %f %f %f %f %f %f');
elseif F1/2==9
   [node(:,1),node(:,2),node(:,3),node(:,4),node(:,5),node(:,6),node(:,7),node(:,8),node(:,9),node(:,10)]=
            textread(str8,'%f %f %f %f %f %f %f %f %f %f');
elseif F1/2==10
   [node(:,1),node(:,2),node(:,3),node(:,4),node(:,5),node(:,6),node(:,7),node(:,8),node(:,9),node(:,10),node(:,11)]=
            textread(str8,'%f %f %f %f %f %f %f %f %f %f %f');
end

wij01(1,:)=ones(1,F1);

CAT01=0;

out_results=0;
a1_test(1,1:F1)=0;
a2_test(1,1:F1)=0;

snode=size(node);

for i=1:snode(1,2)-1
    min0(i)=min(node(:,i+1));
    max0(i)=max(node(:,i+1));
end
```

```
min0(1)=min0(1)-30;

for i=1:snode(1,1)
    for j=2:snode(1,2)
        node(i,j)=(node(i,j)-min0(j-1))/(max0(j-1)-min0(j-1));
    end
end

clear i j

np=snode(1,1);

for point=1:1:np
    out_results(point,1)=node(point,1);
    for i=1:(F1/2)
        out_results(point,i+1)=node(point,i+1);
        a0(point,2*i-1)=node(point,i+1);
        a0(point,2*i)=1-a0(point,2*i-1);
    end
end

clear node point

for k=1:Nolayer
    if k==1
        CAT=CAT01;
        wijtest=wij01;
        [CAT_temp,wijold_temp,map_temp]=fuzzyart(a0,wijtest,CAT,rho(k));

        CAT01=CAT_temp;
        wij01=wijold_temp;
        count1(1:CAT01)=0;
        for point=1:1:np
            count1(map_temp(point))=count1(map_temp(point))+1;
            a1_temp{1,map_temp(point)}(1,count1(map_temp(point)))=point;
            for h=1:F1
                a1{1,map_temp(point)}(count1(map_temp(point)),h)=a0(point,h);
            end
            out_results(point,snode(1,2)+k)=map_temp(point);
        end

        clear CAT_temp wijold_temp map_temp point h
    end

    if k==2
        CAT12(1:CAT01)=0;
        for h=1:CAT01
            if count1(h)>0
                for r=1:count1(h)
                    for t=1:F1
                        a1_test(r,t)=a1{1,h}(r,t);
                    end
                end
                clear r t

                wij12{1,h}(1,:)=ones(1,F1);
                wijtest(1,1:F1)=wij12{1,h}(1,1:F1);
```

```
            CAT=CAT12(h);
            [CAT_temp,wijold_temp,map_temp]=fuzzyart(a1_test,wijtest,CAT,rho(k));
            CAT12(h)=CAT_temp;

            [line12,col12]=size(wijold_temp);
            for r=1:line12
                for t=1:col12
                    wij12{1,h}(r,t)=wijold_temp(r,t);
                end
            end
            clear r t

            for r=1:count1(h)
                out_results(a1_temp{1,h}(1,r),snode(1,2)+k)=map_temp(r);
            end
            clear wijtest map_temp wijold_temp CAT CAT_temp line12 r a1_test
        end
    end

    clear count1 a1_temp a1 h r t

    for h=1:CAT01
        count2{h}(1:CAT12(h))=0;
    end
    clear h

    for point=1:1:np
        count2{1,out_results(point,snode(1,2)+k-1)}(1,out_results(point,snode(1,2)+k))=count2{1,out_results(point,snode(1,2)+k-1)}
                    (1,out_results(point,snode(1,2)+k))+1;
        a2_temp{1,out_results(point,snode(1,2)+k-1)}{1,out_results(point,snode(1,2)+k)}
                    (1,count2{1,out_results(point,snode(1,2)+k-1)}(1,out_results(point,snode(1,2)+k)))=point;
        for h=1:F1
            a2{1,out_results(point,snode(1,2)+k-1)}{1,out_results(point,snode(1,2)+k)}(count2{1,out_results(point,snode(1,2)+k-1)}
                  (1,out_results(point,snode(1,2)+k)),h)=a0(point,h);
        end
    end
    clear point h
end

if k==3
    for h=1:CAT01
        CAT23(h,1:CAT12(h))=0;
    end
    clear h
    for h=1:CAT01
        for r=1:CAT12(h)
            if count2{1,h}(1,r)>0
                for t=1:count2{1,h}(1,r)
                    for u=1:F1
                        a2_test(t,u)=a2{1,h}{1,r}(t,u);
                    end
                end
                clear t u

                wij23{1,h}{1,r}(1,:)=ones(1,F1);
                wijtest(1,1:F1)=wij23{1,h}{1,r}(1,1:F1);
                CAT=CAT23(h,r);
                [CAT_temp,wijold_temp,map_temp]=fuzzyart(a2_test,wijtest,CAT,rho(k));
```

```
                CAT23(h,r)=CAT_temp;

                [line23,col23]=size(wijold_temp);
                for t=1:line23
                    for u=1:col23
                        wij23{1,h}{1,r}(t,u)=wijold_temp(t,u);
                    end
                end
                clear t u

                for t=1:count2{1,h}(1,r)
                    out_results(a2_temp{1,h}{1,r}(1,t),snode(1,2)+k)=map_temp(t);
                end
                clear a2_test wijtest map_temp wijold_temp CAT_temp t
            end
        end
    end

    clear count2 a2_temp a2 h r t u

    for h=1:CAT01
        for r=1:CAT12(h)
            count3{1,h}{1,r}(1,1:CAT23(h,r))=0;
        end
    end
    clear h r

    for point=1:1:np
        count3{1,out_results(point,snode(1,2)+k-2)}{1,out_results(point,snode(1,2)+k-1)}(1,out_results(point,snode(1,2)+k))
                    =count3{1,out_results(point,snode(1,2)+k-2)}{1,out_results(point,snode(1,2)+k-1)}
                                (1,out_results(point,snode(1,2)+k))+1;
        a3_temp{1,out_results(point,snode(1,2)+k-2)}{1,out_results(point,snode(1,2)+k-1)}{1,out_results(point,snode(1,2)+k)}
                                (1,count3{1,out_results(point,snode(1,2)+k-2)}{1,out_results(point,snode(1,2)+k-1)}
                                (1,out_results(point,snode(1,2)+k)))=point;
        for h=1:F1
            a3{1,out_results(point,snode(1,2)+k-2)}{1,out_results(point,snode(1,2)+k-1)}{1,out_results(point,snode(1,2)+k)}
                                (count3{1,out_results(point,snode(1,2)+k-2)}{1,out_results(point,snode(1,2)+k-1)}
                                (1,out_results(point,snode(1,2)+k)),h)=a0(point,h);
        end
    end
    clear point h

end

if k==4
    for h=1:CAT01
        for r=1:CAT12(h)
            CAT34{1,h}(r,1:CAT23(h,r))=0;
        end
    end
    clear h r

    for h=1:CAT01
        for r=1:CAT12(h)
            for t=1:CAT23(h,r)
                if count3{1,h}{1,r}(1,t)>0
                    for u=1:count3{1,h}{1,r}(1,t)
                        for v=1:F1
```

```
                            a3_test(u,v)=a3{1,h}{1,r}{1,t}(u,v);
                        end
                    end
                    clear u v

                    wij34{1,h}{1,r}{1,t}(1,:)=ones(1,F1);
                    wijtest(1,1:F1)=wij34{1,h}{1,r}{1,t}(1,1:F1);
                    CAT=CAT34{1,h}(r,t);
                    [CAT_temp,wijold_temp,map_temp]=fuzzyart(a3_test,wijtest,CAT,rho(k));
                    CAT34{1,h}(r,t)=CAT_temp;

                    [line34,col34]=size(wijold_temp);
                    for u=1:line34
                        for v=1:col34
                            wij34{1,h}{1,r}{1,t}(u,v)=wijold_temp(u,v);
                        end
                    end
                    clear u v

                    for u=1:count3{1,h}{1,r}(1,t)
                        out_results(a3_temp{1,h}{1,r}{1,t}(1,u),snode(1,2)+k)=map_temp(u);
                    end
                    clear a3_test wijtest map_temp wijold_temp CAT_temp u
                end
            end
        end
    end

    clear count3 a3_temp a3 h r t u v
  end
end

clear a0 node3

nr=size(out_results);

fid = fopen('results_train.txt','wt');
for k=1:np
    for h=1:(F1/2+1)
        fprintf(fid,'%f ',out_results(k,h));
    end
    for h=(nr(1,2)-Nolayer+1):nr(1,2)
        fprintf(fid,'%2d ',out_results(k,h));
    end
    fprintf(fid,'\n');
end
clear k
fclose(fid);
set(handles.text9,'string','Training is done!')

function text9_Callback(hObject, eventdata, handles)
% hObject    handle to text9 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text9 as text
%        str2double(get(hObject,'String')) returns contents of text9 as text1 double
```

```
% --- Executes during object creation, after setting all properties.
function text9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text9 (see GCBO)
% eventdata  reserved - to be defined in text1 future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have text1 white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function text11_Callback(hObject, eventdata, handles)
% hObject    handle to text11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text11 as text
%        str2double(get(hObject,'String')) returns contents of text11 as a double


% --- Executes during object creation, after setting all properties.
function text11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in runtest.
function runtest_Callback(hObject, eventdata, handles)
% hObject    handle to runtest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

clear ntest numtest ntest2 ntest3 ttest out_test err_test

global max0 min0 CAT01 CAT12 CAT23 CAT34 wij01 wij12 wij23 wij34 F1 alpha rho Nolayer rho

str9=get(handles.text10,'string');

% [ntest(:,1),ntest(:,2),ntest(:,3)]= textread('NN05117_3min.txt','%f %f %f');

if F1/2==1
    [ntest(:,1),ntest(:,2)]=textread(str9,'%f %f');
elseif F1/2==2
    [ntest(:,1),ntest(:,2),ntest(:,3)]=textread(str9,'%f %f %f');
elseif F1/2==3
    [ntest(:,1),ntest(:,2),ntest(:,3),ntest(:,4)]=textread(str9,'%f %f %f %f');
elseif F1/2==4
    [ntest(:,1),ntest(:,2),ntest(:,3),ntest(:,4),ntest(:,5)]=textread(str9,'%f %f %f %f %f');
elseif F1/2==5
```

```
    [ntest(:,1),ntest(:,2),ntest(:,3),ntest(:,4),ntest(:,5),ntest(:,6)]=textread(str9,'%f %f %f %f %f %f');
elseif F1/2==6
    [ntest(:,1),ntest(:,2),ntest(:,3),ntest(:,4),ntest(:,5),ntest(:,6),ntest(:,7)]=textread(str9,'%f %f %f %f %f %f %f');
elseif F1/2==7
    [ntest(:,1),ntest(:,2),ntest(:,3),ntest(:,4),ntest(:,5),ntest(:,6),ntest(:,7),ntest(:,8)]=
            textread(str9,'%f %f %f %f %f %f %f %f');
elseif F1/2==8
    [ntest(:,1),ntest(:,2),ntest(:,3),ntest(:,4),ntest(:,5),ntest(:,6),ntest(:,7),ntest(:,8),ntest(:,9)]=
            textread(str9,'%f %f %f %f %f %f %f %f %f');
elseif F1/2==9
    [ntest(:,1),ntest(:,2),ntest(:,3),ntest(:,4),ntest(:,5),ntest(:,6),ntest(:,7),ntest(:,8),ntest(:,9),ntest(:,10)]=
            textread(str9,'%f %f %f %f %f %f %f %f %f %f');
elseif F1/2==10
    [ntest(:,1),ntest(:,2),ntest(:,3),ntest(:,4),ntest(:,5),ntest(:,6),ntest(:,7),ntest(:,8),ntest(:,9),ntest(:,10),ntest(:,11)]=
          textread(str9,'%f %f %f %f %f %f %f %f %f %f %f');
end

numtest=size(ntest);

for i=1:numtest(1,1)
    for j=2:numtest(1,2)
        if ntest(i,j)>max0(j-1)
            ntest(i,j)=max0(j-1);
        elseif ntest(i,j)<min0(j-1)
            ntest(i,j)=min0(j-1);
        end
        ntest(i,j)=(ntest(i,j)-min0(j-1))/(max0(j-1)-min0(j-1));
    end
end

err_num=0;
for m=1:numtest(1,1)

    out_test(m,1)=ntest(m,1);
    for i=1:(F1/2)
        out_test(m,i+1)=ntest(m,i+1);
        ttest(2*i-1)=ntest(m,i+1);
        ttest(2*i)=1-ttest(2*i-1);
    end

    for i=1:Nolayer
        if i==1
            CATa=CAT01;
            wijtest=wij01;
        end

        if i==2
            CATa=CAT12(out_test(m,numtest(1,2)+i-1));
            [line12,col12]=size(wij12{1,out_test(m,numtest(1,2)+i-1)});
            for h=1:line12
                for r=1:col12
                    wijtest(h,r)=wij12{1,out_test(m,numtest(1,2)+i-1)}(h,r);
                end
            end
            clear h r
        end

        if i==3
```

```
        CATa=CAT23(out_test(m,numtest(1,2)+i-2),out_test(m,numtest(1,2)+i-1));

        [line23,col23]=size(wij23{1,out_test(m,numtest(1,2)+i-2)}{1,out_test(m,numtest(1,2)+i-1)});

        for h=1:line23
            for r=1:col23
                wijtest(h,r)=wij23{1,out_test(m,numtest(1,2)+i-2)}{1,out_test(m,numtest(1,2)+i-1)}(h,r);
            end
        end
        clear h r
    end

    if i==4
        CATa=CAT34{1,out_test(m,numtest(1,2)+i-3)}(out_test(m,numtest(1,2)+i-2),out_test(m,numtest(1,2)+i-1));
        [line34,col34]=size(wij34{1,out_test(m,numtest(1,2)+i-3)}{1,out_test(m,numtest(1,2)+i-2)}
                                {1,out_test(m,numtest(1,2)+i-1)});
        for h=1:line34
            for r=1:col34
                wijtest(h,r)=wij34{1,out_test(m,numtest(1,2)+i-3)}{1,out_test(m,numtest(1,2)+i-2)}
                                {1,out_test(m,numtest(1,2)+i-1)}(h,r);
            end
        end
        clear h r
    end

    wijtest(CATa+1,1:F1)=1;
    for r=1:CATa+1
        Ti(r)=norm(min(ttest,wijtest(r,:)),1)/(alpha+norm(wijtest(r,:),1));
    end
    clear r

while 2>0
    [Tmax_test,Jmax_test]=max(Ti);
    if norm(min(ttest,wijtest(Jmax_test,:)),1)>=rho(i)*norm(ttest,1)
        break;
    end
    if Tmax_test==0
        break;
    end
    Ti(Jmax_test)=0;
end

if (Jmax_test==CATa+1) || (Tmax_test==0)
    out_test(m,numtest(1,2)+i)=9999;
    err_num=err_num+1;
    for j=1:numtest(1,2)
        err_test(err_num,j)=out_test(m,j);
    end

    for u=1:i
        err_test(err_num,numtest(1,2)+u)=out_test(m,numtest(1,2)+u);
    end
    err_test(err_num,numtest(1,2)+Nolayer+1)=i;
    clear u
    if i<Nolayer
        for u=(i+1):Nolayer
            err_test(err_num,numtest(1,2)+u)=0;
            out_test(m,numtest(1,2)+u)=0;
```

```
            end
        end
%         err_num=err_num+1;
        break;
    else
        out_test(m,numtest(1,2)+i)=Jmax_test;
    end
    clear Ti wijtest CATa
    end
end

clear ntest3

nt=size(out_test);

fid = fopen('results_test.txt','wt');
for k=1:numtest(1,1)

    for h=1:(F1/2+1)
        fprintf(fid,'%f ',out_test(k,h));
    end
    for h=(nt(1,2)-Nolayer+1):nt(1,2)
        fprintf(fid,'%2d ',out_test(k,h));
    end
    fprintf(fid,'\n');

end
clear k
fclose(fid);

if err_num>0
    nc=size(err_test);
    fid = fopen('errors.txt','wt');
    for k=1:err_num
        for h=1:(F1/2+1)
            fprintf(fid,'%f ',err_test(k,h));
        end
        for h=(nc(1,2)-Nolayer):nc(1,2)
            fprintf(fid,'%2d ',err_test(k,h));
        end
        fprintf(fid,'\n');
    end
    clear k
    fclose(fid);
end

clear out_results wij01 wij12 wij23 wij34

set(handles.text11,'string','Testing is done!')

if err_num>0

axes(handles.axes1)

plot(err_test(:,1),err_test(:,nc(1,2)),'*')

set(handles.axes1,'Xlim',[out_test(1,1) out_test(nt(1,1),1)])
set(handles.axes1,'Ylim',[0 Nolayer])
```

```
set(handles.axes1,'XTick',[out_test(1,1):24:out_test(nt(1,1),1)])
set(handles.axes1,'YTick',[0:1:Nolayer])

handles.legend_plot1 = legend('errors');
end
guidata(hObject, handles);



function text4_Callback(hObject, eventdata, handles)
% hObject    handle to text4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text4 as text
%        str2double(get(hObject,'String')) returns contents of text4 as a double


% --- Executes during object creation, after setting all properties.
function text4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function text5_Callback(hObject, eventdata, handles)
% hObject    handle to text5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text5 as text
%        str2double(get(hObject,'String')) returns contents of text5 as a double


% --- Executes during object creation, after setting all properties.
function text5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function text6_Callback(hObject, eventdata, handles)
% hObject    handle to text6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text6 as text
%        str2double(get(hObject,'String')) returns contents of text6 as a double


% --- Executes during object creation, after setting all properties.
function text6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function text8_Callback(hObject, eventdata, handles)
% hObject    handle to text8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text8 as text
%        str2double(get(hObject,'String')) returns contents of text8 as a double


% --- Executes during object creation, after setting all properties.
function text8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1

if get(hObject,'value')
    [filename,pathname,index]=uigetfile({'*.txt';'*.dat'},'training data file');
    if index
        set(handles.text8,'string',[pathname filename])
    end
end
```

```
function text7_Callback(hObject, eventdata, handles)
% hObject    handle to text7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text7 as text
%        str2double(get(hObject,'String')) returns contents of text7 as a double


% --- Executes during object creation, after setting all properties.
function text7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function text10_Callback(hObject, eventdata, handles)
% hObject    handle to text10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of text10 as text
%        str2double(get(hObject,'String')) returns contents of text10 as a double


% --- Executes during object creation, after setting all properties.
function text10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of radiobutton2

if get(hObject,'value')
    [filename,pathname,index]=uigetfile({'*.txt';'*.dat'},'training data file');
    if index
        set(handles.text10,'string',[pathname filename])
    end
end

% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
savePlotWithinGUI(handles.axes1,handles.legend_plot1);
```

B.2 Fuzzy ART network part

```
function [CAT_out,wijold_out,map_out]=fuzzyart(node_in,wijold_in,CAT_in,rho_in)

beta=1;
alpha_in=1e-5;            %choice parameter

wijold_mod=1;

ns=size(node_in);
niter=1;
%  while 1>0,
for niter=1:2

    for npoint=1:ns(1)
        for h=1:ns(2)
            a_in(h)=node_in(npoint,h);
        end

        for j=1:CAT_in+1
            T(j) = norm(min(a_in,wijold_in(j,:)),1)/(alpha_in+norm(wijold_in(j,:),1));
        end
        while 1>0,
            [Tmax_out,Jmax_out]=max(T);
            if norm(min(a_in,wijold_in(Jmax_out,:)),1) >= rho_in*norm(a_in,1)
                map_out(npoint)=Jmax_out;
                break;
            end
            T(Jmax_out)=0;

            if Tmax_out==0
                Jmax_out = CAT_in+1;
                break;
            end
        end
        wijold_in(Jmax_out,:)=beta*min(a_in,wijold_in(Jmax_out,:))+(1-beta)*wijold_in(Jmax_out,:);
```

```
        if Jmax_out==CAT_in+1
            CAT_in=CAT_in+1;
            wijold_in = aug(wijold_in,1);
        end
    end

if size(wijold_in)==size(wijold_mod)
    if norm(wijold_in-wijold_mod)<0.0002
        break;
    end
end
wijold_mod=wijold_in;

end
CAT_out=CAT_in;
wijold_out=wijold_in;
```

B.3 Other functions

```
function aa=aug(a,n)

[lines,cols]=size(a);
u=ones(1,cols);
for i=1:n
  a = [a;u];
end
aa = a;

function rectang(x1,y1,x2,y2,cc)

sym=sprintf('-:-:-:');
col=sprintf('ymcrgbk');

str1=col(mod(cc,4)+1);
str2=sym(mod(cc,4)+1);
col3=sprintf('%s',str1);
lsty3=sprintf('%s',str2);



h=line([x1,x2],[y1,y1]);
set(h,'Color',col3,'LineStyle',lsty3);
% set(h,'Color',col3);
% set(h,'LineStyle',lsty3);
hold on
h=line([x2,x2],[y1,y2]);
 set(h,'Color',col3,'LineStyle',lsty3);
hold on
h=line([x2,x1],[y2,y2]);
 set(h,'Color',col3,'LineStyle',lsty3);
hold on
h=line([x1,x1],[y2,y1]);
```

```
 set(h,'Color',col3,'LineStyle',lsty3);
hold on
```

Appendix C

Double glazed collector TYPE 242 Fortran Code

```
SUBROUTINE TYPE242 (TIME,XIN,OUT,T,DTDT,PAR,INFO,ICNTRL,*)
C************************************************************************
C Object: a
C Simulation Studio Model: Type242
C
C Author: b
C Editor: d
C Date:  January 2010 last modified: January 2010
C
C
C ***
C *** Model Parameters
C ***
C Collector length m [0.;+Inf]
C Collector width m [0.;+Inf]
C Absorber plate thickness m [0.;+Inf]
C Conductivity of absorber material kJ/hr.m.K [0.;+Inf]
C Number of tubes - [1;+Inf]
C Inner tube diameter m [0.;+Inf]
C Outer tube diameter m [0.;+Inf]
C Bond resistance h.m2.K/kJ [0.;+Inf]
C Fluid specific heat kJ/kg.K [0.0;+Inf]
C Absorptance of the absorber plate Fraction [0.;1.]
C Emissivity of the absorber plate Fraction [0.;1.]
C Top loss mode - [1;1]
C Number of identical covers - [0;+Inf]
C Index of refraction of cover material - [0.;+Inf]
C Extinction coefficient, thickness product - [0.;1.]
C Emissivity of the glass Fraction [0.;1.]
C Plate spacing m [0.;+Inf]
C Glass spacing m [0.;+Inf]
```

```
C ***
C *** Model Inputs
C ***
C Inlet temperature C [-Inf;+Inf]
C Inlet flow rate kg/hr [0.0;+Inf]
C Ambient temperature C [-Inf;+Inf]
C Sky temperature C [-Inf;+Inf]
C Wind velocity m/s [0.;+Inf]
C Incident solar radiation kJ/hr.m^2 [0.0;+Inf]
C Total horizontal radiation kJ/hr.m^2 [0.0;+Inf]
C Horizontal diffuse radiation kJ/hr.m^2 [0.0;+Inf]
C Ground reflectance Fraction [0.0;1.0]
C Incidence angle degrees [-360;+360]
C Collector slope degrees [-360;+360]
C Back heat loss coefficient kJ/hr.m^2.K [0.;+Inf]
C Edge heat loss coefficient kJ/hr.m^2.K [0.;+Inf]
C Fluid heat transfer coefficient - [0.;+Inf]
C Atmospheric pressure atm [0.;+Inf]

C ***
C *** Model Outputs
C ***
C Temperature at outlet C [-Inf;+Inf]
C Flow rate at outlet kg/hr [0.0;+Inf]
C Useful energy gain kJ/hr [0.0;+Inf]
C Collector F'- [-Inf;+Inf]
C Collector FR - [-Inf;+Inf]
C Collector top losses kJ/hr [-Inf;+Inf]
C Collector back losses kJ/hr [-Inf;+Inf]
C Collector edge losses kJ/hr [-Inf;+Inf]
C Mean fluid temperature C [-Inf;+Inf]
C Plate temperature C [-Inf;+Inf]
C Incidence angle modifier - [-Inf;+Inf]
C Overall heat loss coefficient kJ/hr.m^2.K [-Inf;+Inf]
C RHO DIFFUSE OUT - [-Inf;+Inf]
C TAU ALPHA OUT - [-Inf;+Inf]
C First glass temperature C [-Inf;+Inf]
C Second glass temperature C [-Inf;+Inf]

C ***
C *** Model Derivatives
C ***

C (Comments and routine interface generated by TRNSYS Studio)
C*************************************************************************

C    TRNSYS acess functions (allow to acess TIME etc.)
      USE TrnsysConstants
      USE TrnsysFunctions

C-------------------------------------------------------------------------------------------------------------------------
C    REQUIRED BY THE MULTI-DLL VERSION OF TRNSYS
      !DEC$ATTRIBUTES DLLEXPORT :: TYPE242 !SET THE CORRECT TYPE NUMBER HERE
C-------------------------------------------------------------------------------------------------------------------------
C-------------------------------------------------------------------------------------------------------------------------
C    TRNSYS DECLARATIONS
      IMPLICIT NONE !REQUIRES THE USER TO DEFINE ALL VARIABLES BEFORE USING THEM
```

```
DOUBLE PRECISION XIN !THE ARRAY FROM WHICH THE INPUTS TO THIS TYPE WILL BE RETRIEVED
DOUBLE PRECISION OUT !THE ARRAY WHICH WILL BE USED TO STORE THE OUTPUTS FROM THIS TYPE
DOUBLE PRECISION TIME !THE CURRENT SIMULATION TIME - YOU MAY USE THIS VARIABLE BUT DO NOT SET IT!
DOUBLE PRECISION PAR !THE ARRAY FROM WHICH THE PARAMETERS FOR THIS TYPE WILL BE RETRIEVED
DOUBLE PRECISION STORED !THE STORAGE ARRAY FOR HOLDING VARIABLES FROM TIMESTEP TO TIMESTEP
DOUBLE PRECISION T !AN ARRAY CONTAINING THE RESULTS FROM THE DIFFERENTIAL EQUATION SOLVER
DOUBLE PRECISION DTDT !AN ARRAY CONTAINING THE DERIVATIVES TO BE PASSED TO THE DIFF.EQ. SOLVER
DOUBLE PRECISION TIME0,TFINAL,DELT
INTEGER*4 INFO(15) !THE INFO ARRAY STORES AND PASSES VALUABLE INFORMATION TO AND FROM THIS TYPE
INTEGER*4 NP,NI,NOUT,ND !VARIABLES FOR THE MAXIMUM NUMBER OF PARAMETERS,INPUTS,OUTPUTS AND DERIVATIVES
INTEGER*4 NPAR,NIN,NDER !VARIABLES FOR THE CORRECT NUMBER OF PARAMETERS,INPUTS,OUTPUTS AND DERIVATIVES
INTEGER*4 IUNIT,ITYPE !THE UNIT NUMBER AND TYPE NUMBER FOR THIS COMPONENT
INTEGER*4 ICNTRL !AN ARRAY FOR HOLDING VALUES OF CONTROL FUNCTIONS WITH THE NEW SOLVER
INTEGER*4 NSTORED !THE NUMBER OF VARIABLES THAT WILL BE PASSED INTO AND OUT OF STORAGE
CHARACTER*3 OCHECK !AN ARRAY TO BE FILLED WITH THE CORRECT VARIABLE TYPES FOR THE OUTPUTS
CHARACTER*3 YCHECK !AN ARRAY TO BE FILLED WITH THE CORRECT VARIABLE TYPES FOR THE INPUTS
C-------------------------------------------------------------------------------------------------------------------------

C-------------------------------------------------------------------------------------------------------------------------
C    USER DECLARATIONS - SET THE MAXIMUM NUMBER OF PARAMETERS (NP), INPUTS (NI),
C    OUTPUTS (NOUT), AND DERIVATIVES (ND) THAT MAY BE SUPPLIED FOR THIS TYPE
      PARAMETER (NP=18,NI=16,NOUT=16,ND=0,NSTORED=0)
C-------------------------------------------------------------------------------------------------------------------------

C-------------------------------------------------------------------------------------------------------------------------
C    REQUIRED TRNSYS DIMENSIONS
      DIMENSION XIN(NI),OUT(NOUT),PAR(NP),YCHECK(NI),OCHECK(NOUT),
1   STORED(NSTORED),T(ND),DTDT(ND)
      INTEGER NITEMS
C-------------------------------------------------------------------------------------------------------------------------
C-------------------------------------------------------------------------------------------------------------------------
C    ADD DECLARATIONS AND DEFINITIONS FOR THE USER-VARIABLES HERE


C    PARAMETERS
      DOUBLE PRECISION RDCONV,PI,LENGTH,WIDTH,THICK_ABSORBER,K_ABSORBER,
1   DIA_TUBE_I,DIA_TUBE_O,R_BOND,CP_FLUID,ABS_PLATE,EMISS_PLATE,
1   REFR_INDEX,KL_COVER,RHO_DIFFUSE,TAU_ALPHA_N,TAU_ALPHA,M,X,FR,
     1   T_FLUID_IN,FLOW_IN,T_AMB,T_SKY,WINDSPEED,GT,GH,GDH,RHO_GROUND,
1   ANGLE_INC,SLOPE,U_BACK,U_EDGES,H_FLUID,U_TOP,U_L,AREA,W,XKAT,
1   EFFSKY,EFFGND,COSSLOPE,FSKY,FGND,GDSKY,GDGND,XKATDS,XKATDG,
     1   XKATB,T_FLUID_OUT,T_PLATE_MEAN,H_CONV,H_RAD,H_RADIATION,T_K,
     1   T_FLUID_MEAN,TMC,TAC,F,C,STF1,STF2,F_PRIME,QU,Q_TOP,Q_BACK,
     1   Q_EDGES,FPP,T_FLUID_OUT_OLD,P_ATM,P_KPA,T_GLASS1,T_GLASS2,
1   T_GLASS1_OLD,T_GLASS2_OLD,AIRPROPS,EMISS_GLASS,HR_PC1,HR_C1C2,
1   HR_C2A,RADCOEPP,RADCOEPA,H_RAD_GRAY,K_VISC,THERM_COND,PRAND_N,
1   T_PG1,T_G1G2,RAYLEIGH_PG1,RAYLEIGH_G1G2,RAYLEIGH_NUM,DETA_TPG1,
1   DETA_TG1G2,DIS_PG1,DIS_G1G2,NUSSELT_PG1,NUSSELT_G1G2,
1   H_CONV_PG1,H_CONV_G1G2,NUSSELT_NUM,H_CONVECTION,
1   T_PLATE_MEAN_OLD,U_TOP2
INTEGER N_TUBES,ICOUNT,MODE_U,N_COVERS
CHARACTER(LEN=MAXMESSAGELENGTH)::MESSAGE1,MESSAGE2
DIMENSION AIRPROPS(5)
   IUNIT=INFO(1)
   ITYPE=INFO(2)
C-------------------------------------------------------------------------------------------------------------------------
```

```
C-------------------------------------------------------------------------------------------------------------------------
C    DATA STATEMENTS
      DATA RDCONV/0.017453292/
DATA MESSAGE1/'An illegal overall heat transfer coefficient has be
1en calculated by the model.  Please check the entering information
     1 carefully.'/
DATA MESSAGE2/'Unable to find a stable solution for the mean plate
1 temperature.'/
C-------------------------------------------------------------------------------------------------------------------------

C-------------------------------------------------------------------------------------------------------------------------
C    GET GLOBAL TRNSYS SIMULATION VARIABLES
      TIME0=getSimulationStartTime()
      TFINAL=getSimulationStopTime()
      DELT=getSimulationTimeStep()
C-------------------------------------------------------------------------------------------------------------------------

C-------------------------------------------------------------------------------------------------------------------------
C    SET THE VERSION INFORMATION FOR TRNSYS
      IF(INFO(7).EQ.-2) THEN
   INFO(12)=16
   RETURN 1
ENDIF
C-------------------------------------------------------------------------------------------------------------------------

C-------------------------------------------------------------------------------------------------------------------------
C    DO ALL THE VERY LAST CALL OF THE SIMULATION MANIPULATIONS HERE
      IF (INFO(8).EQ.-1) THEN
   RETURN 1
ENDIF
C-------------------------------------------------------------------------------------------------------------------------

C-------------------------------------------------------------------------------------------------------------------------
C    PERFORM ANY 'AFTER-ITERATION' MANIPULATIONS THAT ARE REQUIRED HERE
C    e.g. save variables to storage array for the next timestep
      IF (INFO(13).GT.0) THEN
   NITEMS=0
C    STORED(1)=... (if NITEMS > 0)
C        CALL setStorageVars(STORED,NITEMS,INFO)
   RETURN 1
ENDIF
C
C-------------------------------------------------------------------------------------------------------------------------

C-------------------------------------------------------------------------------------------------------------------------
C    DO ALL THE VERY FIRST CALL OF THE SIMULATION MANIPULATIONS HERE
      IF (INFO(7).EQ.-1) THEN

C       SET SOME INFO ARRAY VARIABLES TO TELL THE TRNSYS ENGINE HOW THIS TYPE IS TO WORK
         INFO(6)=NOUT
         INFO(9)=1
   INFO(10)=0 !STORAGE FOR VERSION 16 HAS BEEN CHANGED

C       SET THE REQUIRED NUMBER OF INPUTS, PARAMETERS AND DERIVATIVES THAT THE USER SHOULD SUPPLY IN THE INPUT FILE
C       IN SOME CASES, THE NUMBER OF VARIABLES MAY DEPEND ON THE VALUE OF PARAMETERS TO THIS MODEL....
         NIN=NI
   NPAR=NP
   NDER=ND
```

```
C       SET THE REQUIRED NUMBER OF INPUTS BASED ON THE MODE FOR THE TOP LOSSES
   MODE_U=JFIX(PAR(12)+0.5)
         IF(MODE_U.LT.1) CALL TYPECK(-4,INFO,0,12,0)
         IF(MODE_U.GT.2) CALL TYPECK(-4,INFO,0,12,0)

   IF(MODE_U.EQ.1) THEN
      NIN=15
   ELSE
      NIN=16
   ENDIF

C       CALL THE TYPE CHECK SUBROUTINE TO COMPARE WHAT THIS COMPONENT REQUIRES TO WHAT IS SUPPLIED IN
C       THE TRNSYS INPUT FILE
   CALL TYPECK(1,INFO,NIN,NPAR,NDER)

C       SET THE YCHECK AND OCHECK ARRAYS TO CONTAIN THE CORRECT VARIABLE TYPES FOR THE INPUTS AND OUTPUTS
         DATA YCHECK/'TE1','MF1','TE1','TE1','VE1','IR1','IR1','IR1',
1      'DM1','DG1','DG1','HT1','HT1','HT1','PR4','HT1'/
         DATA OCHECK/'TE1','MF1','PW1','DM1','DM1','PW1','PW1','PW1',
1      'TE1','TE1','DM1','HT1','DM1','DM1','TE1','TE1'/

C       CALL THE RCHECK SUBROUTINE TO SET THE CORRECT INPUT AND OUTPUT TYPES FOR THIS COMPONENT
C         CALL RCHECK(INFO,YCHECK,OCHECK)

C       SET THE NUMBER OF STORAGE SPOTS NEEDED FOR THIS COMPONENT
         NITEMS=0
C    CALL setStorageSize(NITEMS,INFO)

C       RETURN TO THE CALLING PROGRAM
         RETURN 1

      ENDIF
C-----------------------------------------------------------------------------------------------------------------------

C-----------------------------------------------------------------------------------------------------------------------
C    DO ALL OF THE INITIAL TIMESTEP MANIPULATIONS HERE - THERE ARE NO ITERATIONS AT THE INTIAL TIME
      IF (TIME .LT. (getSimulationStartTime() +
     . getSimulationTimeStep()/2.D0)) THEN

C       SET THE UNIT NUMBER FOR FUTURE CALLS
         IUNIT=INFO(1)
         ITYPE=INFO(2)

C       READ IN THE VALUES OF THE PARAMETERS IN SEQUENTIAL ORDER
         LENGTH=PAR(1)
   WIDTH=PAR(2)
   THICK_ABSORBER=PAR(3)
   K_ABSORBER=PAR(4)
   N_TUBES=JFIX(PAR(5)+0.5)
   DIA_TUBE_I=PAR(6)
   DIA_TUBE_O=PAR(7)
   R_BOND=PAR(8)
   CP_FLUID=PAR(9)
   ABS_PLATE=PAR(10)
   EMISS_PLATE=PAR(11)
   MODE_U=JFIX(PAR(12)+0.5)
   N_COVERS=JFIX(PAR(13)+0.5)
```

```
   REFR_INDEX=PAR(14)
         KL_COVER=PAR(15)
   EMISS_GLASS=PAR(16)
         DIS_PG1=PAR(17)
   DIS_G1G2=PAR(18)

C       CHECK THE PARAMETERS FOR PROBLEMS AND RETURN FROM THE SUBROUTINE IF AN ERROR IS FOUND
          IF(LENGTH.LE.0.) CALL TYPECK(-4,INFO,0,1,0)
          IF(WIDTH.LE.0.) CALL TYPECK(-4,INFO,0,2,0)
          IF(THICK_ABSORBER.LE.0.) CALL TYPECK(-4,INFO,0,3,0)
          IF(K_ABSORBER.LE.0.) CALL TYPECK(-4,INFO,0,4,0)
          IF(N_TUBES.LT.1) CALL TYPECK(-4,INFO,0,5,0)
          IF(DIA_TUBE_I.LE.0.) CALL TYPECK(-4,INFO,0,6,0)
          IF(DIA_TUBE_O.LE.0.) CALL TYPECK(-4,INFO,0,7,0)
          IF(DIA_TUBE_O*DBLE(N_TUBES).GT.WIDTH)
1      CALL TYPECK(-4,INFO,0,7,0)
          IF(DIA_TUBE_O.LE.DIA_TUBE_I) CALL TYPECK(-4,INFO,0,7,0)
          IF(R_BOND.LT.0.) CALL TYPECK(-4,INFO,0,8,0)
          IF(CP_FLUID.LE.0.) CALL TYPECK(-4,INFO,0,9,0)
          IF(ABS_PLATE.LE.0.) CALL TYPECK(-4,INFO,0,10,0)
          IF(ABS_PLATE.GT.1.) CALL TYPECK(-4,INFO,0,10,0)
          IF(EMISS_PLATE.LT.0.) CALL TYPECK(-4,INFO,0,11,0)
          IF(EMISS_PLATE.GT.1.) CALL TYPECK(-4,INFO,0,11,0)
          IF(N_COVERS.LT.0) CALL TYPECK(-4,INFO,0,13,0)
          IF(REFR_INDEX.LE.0.) CALL TYPECK(-4,INFO,0,14,0)
          IF(KL_COVER.LT.0.) CALL TYPECK(-4,INFO,0,15,0)

C       SET THE TRANSMITTANCE-ABSORPTANCE PRODUCT AT NORMAL INCIDENCE AND THE REFLECTANCE OF THE COVER
C       TO DIFFUSE RADIATION
   RHO_DIFFUSE=-1.
         TAU_ALPHA_N=TAU_ALPHA(N_COVERS,0.D0,KL_COVER,REFR_INDEX,
C          TAU_ALPHA_N=TAU_ALPHA(N_COVERS,45.,KL_COVER,REFR_INDEX,
1      ABS_PLATE,RHO_DIFFUSE)

C       CHECK THE PARAMETERS FOR PROBLEMS AND RETURN FROM THE SUBROUTINE IF AN ERROR IS FOUND
C         IF(...) CALL TYPECK(-4,INFO,0,"BAD PARAMETER #",0)

C       PERFORM ANY REQUIRED CALCULATIONS TO SET THE INITIAL VALUES OF THE OUTPUTS HERE
C       PERFORM ANY REQUIRED CALCULATIONS TO SET THE INITIAL VALUES OF THE OUTPUTS HERE
          OUT(1)=XIN(1)
          OUT(2:8)=0.
   OUT(9)=XIN(1)
       OUT(10)=XIN(1)
   OUT(11:12)=0.
   OUT(13)=RHO_DIFFUSE
   OUT(14)=TAU_ALPHA_N
   OUT(15)=0.
   OUT(16)=0.

C       PERFORM ANY REQUIRED CALCULATIONS TO SET THE INITIAL STORAGE VARIABLES HERE
          NITEMS=0
C     STORED(1)=...

C       PUT THE STORED ARRAY IN THE GLOBAL STORED ARRAY
C         CALL setStorageVars(STORED,NITEMS,INFO)

C       RETURN TO THE CALLING PROGRAM
          RETURN 1
```

```
      ENDIF
C-----------------------------------------------------------------------------------------------------------------------

C-----------------------------------------------------------------------------------------------------------------------
C    *** ITS AN ITERATIVE CALL TO THIS COMPONENT ***
C-----------------------------------------------------------------------------------------------------------------------

C-----------------------------------------------------------------------------------------------------------------------
C    RE-READ THE PARAMETERS IF ANOTHER UNIT OF THIS TYPE HAS BEEN CALLED
      IF(INFO(1).NE.IUNIT) THEN

C       RESET THE UNIT NUMBER
   IUNIT=INFO(1)
   ITYPE=INFO(2)

C       READ IN THE VALUES OF THE PARAMETERS IN SEQUENTIAL ORDER
         LENGTH=PAR(1)
   WIDTH=PAR(2)
   THICK_ABSORBER=PAR(3)
   K_ABSORBER=PAR(4)
   N_TUBES=JFIX(PAR(5)+0.5)
   DIA_TUBE_I=PAR(6)
   DIA_TUBE_O=PAR(7)
   R_BOND=PAR(8)
   CP_FLUID=PAR(9)
   ABS_PLATE=PAR(10)
   EMISS_PLATE=PAR(11)
   MODE_U=JFIX(PAR(12)+0.5)
   N_COVERS=JFIX(PAR(13)+0.5)
   REFR_INDEX=PAR(14)
         KL_COVER=PAR(15)
   EMISS_GLASS=PAR(16)
   DIS_PG1=PAR(17)
   DIS_G1G2=PAR(18)

      ENDIF
C-----------------------------------------------------------------------------------------------------------------------

C-----------------------------------------------------------------------------------------------------------------------
C    RETRIEVE THE CURRENT VALUES OF THE INPUTS TO THIS MODEL FROM THE XIN ARRAY IN SEQUENTIAL ORDER
      T_FLUID_IN=XIN(1)
FLOW_IN=XIN(2)
T_AMB=XIN(3)
T_SKY=XIN(4)
WINDSPEED=XIN(5)
      GT=XIN(6)
      GH=XIN(7)
      GDH=XIN(8)
      RHO_GROUND=XIN(9)
      ANGLE_INC=XIN(10)
      SLOPE=XIN(11)
      U_BACK=XIN(12)
      U_EDGES=XIN(13)
      H_FLUID=XIN(14)
P_ATM=XIN(15)

      IF(MODE_U.GT.1) THEN
```

```
        U_TOP=XIN(16)
ELSE
   U_TOP=0.
ENDIF
C-------------------------------------------------------------------------------------------------------------------------

C-------------------------------------------------------------------------------------------------------------------------
C    CHECK THE INPUTS FOR PROBLEMS
      IF(FLOW_IN.LT.0.) CALL TYPECK(-3,INFO,2,0,0)
      IF(WINDSPEED.LT.0.) CALL TYPECK(-3,INFO,5,0,0)
      IF(GT.LT.0.) CALL TYPECK(-3,INFO,6,0,0)
      IF(GH.LT.0.) CALL TYPECK(-3,INFO,7,0,0)
      IF(GDH.LT.0.) CALL TYPECK(-3,INFO,8,0,0)
      IF(RHO_GROUND.LT.0.) CALL TYPECK(-3,INFO,9,0,0)
      IF(RHO_GROUND.GT.1.) CALL TYPECK(-3,INFO,9,0,0)
      IF(U_BACK.LT.0.) CALL TYPECK(-3,INFO,12,0,0)
      IF(U_EDGES.LT.0.) CALL TYPECK(-3,INFO,13,0,0)
      IF(H_FLUID.LE.0.) CALL TYPECK(-3,INFO,14,0,0)
      IF(P_ATM.LE.0.) CALL TYPECK(-3,INFO,15,0,0)
      IF(P_ATM.GT.5.) CALL TYPECK(-3,INFO,15,0,0)
      IF(U_TOP.LT.0.) CALL TYPECK(-3,INFO,16,0,0)
IF(ERRORFOUND()) RETURN 1
C-------------------------------------------------------------------------------------------------------------------------


C-------------------------------------------------------------------------------------------------------------------------
C    RETRIEVE THE VALUES IN THE STORAGE ARRAY FOR THIS ITERATION
C      NITEMS=
C CALL getStorageVars(STORED,NITEMS,INFO)
C      STORED(1)=
C-------------------------------------------------------------------------------------------------------------------------
C-------------------------------------------------------------------------------------------------------------------------
C    CHECK THE INPUTS FOR PROBLEMS
C      IF(...) CALL TYPECK(-3,INFO,'BAD INPUT #',0,0)
C IF(IERROR.GT.0) RETURN 1
C-------------------------------------------------------------------------------------------------------------------------
C-------------------------------------------------------------------------------------------------------------------------
C    *** PERFORM ALL THE CALCULATION HERE FOR THIS MODEL. ***
C-------------------------------------------------------------------------------------------------------------------------

C ADD YOUR COMPONENT EQUATIONS HERE; BASICALLY THE EQUATIONS THAT WILL
C CALCULATE THE OUTPUTS BASED ON THE PARAMETERS AND THE INPUTS. REFER TO
C CHAPTER 3 OF THE TRNSYS VOLUME 1 MANUAL FOR DETAILED INFORMATION ON
C WRITING TRNSYS COMPONENTS.


C    SET PI
      PI=4*DATAN(1.D0)


C    CALCULATE THE AREA OF THE COLLECTOR
      AREA=LENGTH*WIDTH

C    CALCULATE THE TUBE-TO-TUBE DISTANCE
      W=WIDTH/DBLE(N_TUBES)

C    RETRIEVE THE TRANSMITTANCE ABSORPTANCE PRODUCT AT NORMAL INCIDENCE AND THE REFLECTANCE TO DIFFUSE
      RHO_DIFFUSE=OUT(13)
```

```
      TAU_ALPHA_N=OUT(14)

C    GET THE INCIDENCE ANGLE MODIFIER
      IF(N_COVERS.LT.1) THEN
   XKAT=1.

ELSE

C       USE THE RELATIONS OF BRANDEMUEHL TO GET THE EFFECTIVE INCIDENCE ANGLES FOR DIFFUSE RADIATION
         EFFSKY=59.68-0.1388*SLOPE+0.001497*SLOPE*SLOPE
         EFFGND=90.-0.5788*SLOPE+0.002693*SLOPE*SLOPE
         COSSLOPE=DCOS(SLOPE*RDCONV)
         FSKY=(1.+COSSLOPE)/2.
         FGND=(1.-COSSLOPE)/2.
         GDSKY=FSKY*GDH
         GDGND=RHO_GROUND*FGND*GH


C       USE THE TAU_ALPHA FUNCTION FOR THE COMPONENT IAM VALUES
         XKATDS=TAU_ALPHA(N_COVERS,EFFSKY,KL_COVER,REFR_INDEX,ABS_PLATE,
1      RHO_DIFFUSE)/TAU_ALPHA_N
         XKATDG=TAU_ALPHA(N_COVERS,EFFGND,KL_COVER,REFR_INDEX,ABS_PLATE,
1      RHO_DIFFUSE)/TAU_ALPHA_N
         XKATB=TAU_ALPHA(N_COVERS,ANGLE_INC,KL_COVER,REFR_INDEX,
1      ABS_PLATE,RHO_DIFFUSE)/TAU_ALPHA_N

C       CALCULATE THE OVERALL IAM
         IF(GT.GT.0.) THEN
            XKAT=(XKATB*(GT-GDSKY-GDGND)+XKATDS*GDSKY+XKATDG*GDGND)/GT
   ELSE
      XKAT=0.
   ENDIF
ENDIF

C    GUESS AN OUTPUT TEMPERATURE
      T_FLUID_OUT=T_FLUID_IN

C    GUESS THE MEAN PLATE ND MEAN FLUID TEMPERATURES
      T_PLATE_MEAN=(T_FLUID_IN+T_FLUID_OUT)/2.
C      T_PLATE_MEAN=100.
      T_FLUID_MEAN=(T_FLUID_IN+T_FLUID_OUT)/2.

C    GUESS THE GLASS1 AND GLASS2 TEMPERATURES
      T_GLASS1=T_PLATE_MEAN-5.
T_GLASS2=T_PLATE_MEAN-10.


C    INITIALIZE A FEW VARIABLES
      ICOUNT=1
T_FLUID_OUT_OLD=T_FLUID_OUT
T_GLASS1_OLD=T_GLASS1
T_GLASS2_OLD=T_GLASS2

C    SET THE TOP LOSS COEFFICIENCT
100   IF(MODE_U.EQ.1) THEN

C       SET THE TOP LOSS FROM CONVECTION AND RADIATION
   IF(N_COVERS.LT.1) THEN
```

```
  P_KPA=P_ATM*101.325
      T_K=T_AMB+273.15
            CALL WINDCOEF(WINDSPEED,LENGTH,WIDTH,T_K,P_KPA,H_CONV)
      H_CONV=H_CONV*3.6  !CONVERT W/M2/K TO KJ/H/M2.K

      H_RAD=H_RADIATION(T_PLATE_MEAN,T_SKY,EMISS_PLATE)
      U_TOP=H_CONV+H_RAD

C       USE KLEIN'S TOP LOSS CORRELATION
   ELSE

   LENGTH=PAR(1)
   WIDTH=PAR(2)
   THICK_ABSORBER=PAR(3)
   K_ABSORBER=PAR(4)
   N_TUBES=JFIX(PAR(5)+0.5)
   DIA_TUBE_I=PAR(6)
   DIA_TUBE_O=PAR(7)
   R_BOND=PAR(8)
   CP_FLUID=PAR(9)
   ABS_PLATE=PAR(10)
   EMISS_PLATE=PAR(11)
   MODE_U=JFIX(PAR(12)+0.5)
   N_COVERS=JFIX(PAR(13)+0.5)
   REFR_INDEX=PAR(14)
         KL_COVER=PAR(15)
   EMISS_GLASS=PAR(16)
   DIS_PG1=PAR(17)
   DIS_G1G2=PAR(18)

   T_FLUID_IN=XIN(1)
   FLOW_IN=XIN(2)
   T_AMB=XIN(3)
   T_SKY=XIN(4)
   WINDSPEED=XIN(5)
         GT=XIN(6)
         GH=XIN(7)
         GDH=XIN(8)
         RHO_GROUND=XIN(9)
         ANGLE_INC=XIN(10)
         SLOPE=XIN(11)
         U_BACK=XIN(12)
         U_EDGES=XIN(13)
         H_FLUID=XIN(14)
   P_ATM=XIN(15)


C          SET THE MEAN FLUID TEMPERATURE
  P_KPA=P_ATM*101.325
      T_K=T_AMB+273.15

            IF (GT.GE.50.) THEN
            CALL WINDCOEF(WINDSPEED,LENGTH,WIDTH,T_K,P_KPA,H_CONV)

            HR_PC1=H_RAD_GRAY(T_PLATE_MEAN,T_GLASS1,EMISS_PLATE,
     1            EMISS_GLASS)/3.6
            HR_C1C2=H_RAD_GRAY(T_GLASS1,T_GLASS2,EMISS_GLASS,
```

```
     1      EMISS_GLASS)/3.6
      HR_C2A=H_RADIATION(T_GLASS2,T_SKY,EMISS_GLASS)*
     1             DABS(T_GLASS2-T_SKY)/DABS(T_GLASS2-T_AMB+1.D-3)/3.6

  T_PG1=(T_PLATE_MEAN+T_GLASS1)/2.+273.15
      T_G1G2=(T_GLASS1+T_GLASS2)/2.+273.15
      DETA_TPG1=DABS(T_PLATE_MEAN-T_GLASS1)
      DETA_TG1G2=DABS(T_GLASS1-T_GLASS2)

      CALL AIRPROP(T_PG1,P_KPA,AIRPROPS)
           K_VISC=AIRPROPS(2)
      THERM_COND=AIRPROPS(4)
      PRAND_N=AIRPROPS(3)

      RAYLEIGH_PG1=RAYLEIGH_NUM(T_PG1,DETA_TPG1,DIS_PG1,K_VISC,
     1         PRAND_N)
  NUSSELT_PG1=NUSSELT_NUM(RAYLEIGH_PG1,SLOPE)
      H_CONV_PG1=H_CONVECTION(NUSSELT_PG1,THERM_COND,DIS_PG1)


      CALL AIRPROP(T_G1G2,P_KPA,AIRPROPS)
           K_VISC=AIRPROPS(2)
      THERM_COND=AIRPROPS(4)
      PRAND_N=AIRPROPS(3)

      RAYLEIGH_G1G2=RAYLEIGH_NUM(T_G1G2,DETA_TG1G2,DIS_G1G2,
     1   K_VISC,PRAND_N)
           NUSSELT_G1G2=NUSSELT_NUM(RAYLEIGH_G1G2,SLOPE)
      H_CONV_G1G2=H_CONVECTION(NUSSELT_G1G2,THERM_COND,DIS_G1G2)

      U_TOP=1/(H_CONV_PG1+HR_PC1)+1/(H_CONV_G1G2+HR_C1C2)+
     1   1/(H_CONV+HR_C2A+1.D-3)
           U_TOP=1/U_TOP

      T_GLASS1=T_PLATE_MEAN-U_TOP*(T_PLATE_MEAN-T_AMB)/
     1              (H_CONV_PG1+HR_PC1)
      T_GLASS2=T_GLASS1-U_TOP*(T_PLATE_MEAN-T_AMB+1.D-3)/
     1              (H_CONV_G1G2+HR_C1C2)

      U_TOP=U_TOP*3.6
           ENDIF

   ENDIF

ELSE
   U_TOP=XIN(16)

ENDIF

C    SET THE OVERALL LOSS COEFFICIENT
      U_L=U_TOP+U_BACK+U_EDGES
IF(U_L.LE.0.) THEN
         CALL MESSAGES(-1,MESSAGE1,'FATAL',IUNIT,ITYPE)
   RETURN 1
ENDIF

C    SET SOME REQUIRED VARIABLES
      M=(U_L/K_ABSORBER/THICK_ABSORBER)**0.5
```

```
F=DTANH((M*(W-DIA_TUBE_O)/2.))/(M*(W-DIA_TUBE_O)/2.)
      X=1./(U_L*(DIA_TUBE_O+(W-DIA_TUBE_O)*F))+R_BOND+
1   1./PI/DIA_TUBE_I/H_FLUID

C    SET THE COLLECTOR EFFICIENCY FACTOR
F_PRIME=1./U_L/W/X

C    CALCULATE THE COLLECTOR HEAT REMOVAL FACTOR
      IF(FLOW_IN.LE.0.) THEN
   FR=0.
ELSE
   FR=FLOW_IN*CP_FLUID/AREA/U_L*(1.-DEXP(-AREA*U_L*F_PRIME/
1      FLOW_IN/CP_FLUID))
      ENDIF

C    CALCULATE THE COLLECTOR USEFUL ENERGY GAIN
      QU=AREA*FR*(GT*TAU_ALPHA_N*XKAT-U_L*(T_FLUID_IN-T_AMB))

C    CALCULATE THE COLLECTOR OUTLET TEMPERATURE
      IF (FLOW_IN.LE.0.) THEN

T_FLUID_OUT=GT*XKAT*TAU_ALPHA_N/U_L+T_AMB

    T_PLATE_MEAN=T_FLUID_OUT
    T_FLUID_MEAN=T_FLUID_OUT

C      IF ((T_PLATE_MEAN-T_AMB).LE.0.) THEN
          IF (GT.LE.50.) THEN
    T_PLATE_MEAN_OLD=T_PLATE_MEAN


P_KPA=P_ATM*101.325
    T_K=T_AMB+273.15
         CALL WINDCOEF(WINDSPEED,LENGTH,WIDTH,T_K,P_KPA,H_CONV)
150      T_PLATE_MEAN=T_AMB-EMISS_GLASS*5.67D-8*((T_PLATE_MEAN+273.15)
     1           **4-(T_SKY+273.15)**4)/H_CONV
         IF (DABS(T_PLATE_MEAN-T_PLATE_MEAN_OLD).GT.0.001) THEN
         T_PLATE_MEAN_OLD=T_PLATE_MEAN
        GOTO 150
    ENDIF

    H_CONV=H_CONV*3.6
    T_GLASS1=T_PLATE_MEAN
    T_GLASS2=T_PLATE_MEAN

    T_FLUID_OUT=(T_PLATE_MEAN+T_AMB)/2.
    T_FLUID_MEAN=T_PLATE_MEAN

Q_TOP=AREA*H_CONV*(T_AMB-T_GLASS2)
    Q_BACK=AREA*U_BACK*(T_PLATE_MEAN-T_AMB)
    Q_EDGES=AREA*U_EDGES*(T_PLATE_MEAN-T_AMB)
    ENDIF


ELSE
   T_FLUID_OUT=T_FLUID_IN+QU/FLOW_IN/CP_FLUID
        FPP=FR/F_PRIME
   T_FLUID_MEAN=T_FLUID_IN+QU/AREA*(1.-FPP)/FR/U_L
```

```
   T_PLATE_MEAN=T_FLUID_IN+QU/AREA*(1.-FR)/FR/U_L

   Q_TOP=AREA*U_TOP*(T_PLATE_MEAN-T_AMB)
   Q_BACK=AREA*U_BACK*(T_PLATE_MEAN-T_AMB)
   Q_EDGES=AREA*U_EDGES*(T_PLATE_MEAN-T_AMB)

      ENDIF

C    SEE IF CONVERGENCE HAS BEEN REACHED
      IF((ICOUNT.LT.200).AND.(DABS(T_FLUID_OUT_OLD-T_FLUID_OUT
     1    ).GT.0.001).AND.(DABS(T_GLASS1_OLD-T_GLASS1).GT.0.001).
     1    AND.(DABS(T_GLASS2_OLD-T_GLASS2).GT.0.001)) THEN
   ICOUNT=ICOUNT+1
   T_FLUID_OUT_OLD=T_FLUID_OUT
   T_GLASS1_OLD=T_GLASS1
   T_GLASS2_OLD=T_GLASS2
   GOTO 100
ENDIF

C    WARN THE USER IF CONVERGENCE HAS NOT BEEN OBTAINED
      IF(ICOUNT.GE.50) THEN
         CALL MESSAGES(-1,MESSAGE2,'WARNING',IUNIT,ITYPE)
ENDIF


C------------------------------------------------------------------------------------------------------------------------

C------------------------------------------------------------------------------------------------------------------------
C------------------------------------------------------------------------------------------------------------------------
C    SET THE STORAGE ARRAY AT THE END OF THIS ITERATION IF NECESSARY
C      NITEMS=
C      STORED(1)=
C CALL setStorageVars(STORED,NITEMS,INFO)
C------------------------------------------------------------------------------------------------------------------------
C------------------------------------------------------------------------------------------------------------------------
C    REPORT ANY PROBLEMS THAT HAVE BEEN FOUND USING CALLS LIKE THIS:
C      CALL MESSAGES(-1,'put your message here','MESSAGE',IUNIT,ITYPE)
C      CALL MESSAGES(-1,'put your message here','WARNING',IUNIT,ITYPE)
C      CALL MESSAGES(-1,'put your message here','SEVERE',IUNIT,ITYPE)
C      CALL MESSAGES(-1,'put your message here','FATAL',IUNIT,ITYPE)
C------------------------------------------------------------------------------------------------------------------------
C------------------------------------------------------------------------------------------------------------------------
C    SET THE OUTPUTS FROM THIS MODEL IN SEQUENTIAL ORDER AND GET OUT

      OUT(1)=T_FLUID_OUT
      OUT(2)=FLOW_IN
      OUT(3)=QU
OUT(4)=F_PRIME
OUT(5)=FR
OUT(6)=Q_TOP
OUT(7)=Q_BACK
OUT(8)=Q_EDGES
OUT(9)=T_FLUID_MEAN
OUT(10)=T_PLATE_MEAN
OUT(11)=XKAT
OUT(12)=U_L
OUT(13)=RHO_DIFFUSE
OUT(14)=TAU_ALPHA_N
```

```
OUT(15)=T_GLASS1
OUT(16)=T_GLASS2


C---------------------------------------------------------------------------------------------------------------------------
C    EVERYTHING IS DONE - RETURN FROM THIS SUBROUTINE AND MOVE ON
      RETURN 1
      END
C---------------------------------------------------------------------------------------------------------------------------
```

References

D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. A learning algorithm for boltzmann machines. Cognitive Science, 9:147–169, 1985.

James A. Anderson. A simple neural network generating an interactive memory. Mathematical Biosciences, 14(34):197 – 220, 1972.

Michle Basseville. Detecting changes in signals and systemsa survey. Automatica, 24(3):309 – 326, 1988.

S. Bendapudi and J.E. Braun. A review of literatture on dynamic models for vapor compression equipment. Technical Report HL 2002-9, Report Number 4036-5, Ray Herrick Laboratories, Purdue University, 2002.

A. Bryson and Y. Ho. Applied optimal control: optimization, estimation, and control. Ginn/Blaisdell, 1969.

E. Cardozo and S.N. Talukdar. A distributed expert system for fault diagnosis. Power Systems, IEEE Transactions on, 3(2):641 –646, may 1988.

Gail A. Carpenter and Stephen Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. Computer Vision, Graphics, and Image Processing, 37(1):54 – 115, 1987.

Gail A. Carpenter, Stephen Grossberg, and John H. Reynolds. Artmap: Supervised

real-time learning and classification of nonstationary data by a self-organizing neural network. Neural Networks, 4(5):565 – 588, 1991a.

Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks, 4(6):759 – 771, 1991b.

T.P. Caudell and D.S. Newman. An adaptive resonance architecture to define normality and detect novelties in time series and databases. In Proceedings of the INNS World Congress on Neural Networks, volume IV, pages 166–176, Portland, July 1993.

T.P. Caudell, S.D.G. Smith, R. Escobedo, and M. Anderson. Nirs: Large scale art-1 neural architectures for engineering design retrieval. Neural Networks, 7: 1339–1350, 1994.

E. Chow and A. Willsky. Analytical redundancy and the design of robust failure detection systems. Automatic Control, IEEE Transactions on, 29(7):603 – 614, Jul 1984.

R.N. Clark. A simplified instrument failure detection scheme. Aerospace and Electronic Systems, IEEE Transactions on, AES-14(4):558 –563, July 1978a.

R.N. Clark. Instrument fault detection. Aerospace and Electronic Systems, IEEE Transactions on, AES-14(3):456 –465, May 1978b.

DOE and EIA. Solar thermal collector manufacturing activities 2009. Technical report, U.S. Department of Energy, U.S. Energy Information Administration, 2010.

DOE and EIA. Annual energy review 2010. Technical report, U.S. Department of Energy, U.S. Energy Information Administration, 2011.

J. Du and Meng Joo Er. An artificial intelligence approach towards fault diagnosis of an air-handling unit. In Control Conference, 2004. 5th Asian, volume 3, pages 1594 – 1601 Vol.3, july 2004a.

J. Du and Meng Joo Er. Fault diagnosis in air-handling unit system using dynamic fuzzy neural network. In Proceedings of the 6th International FLINS Conference,Blankenberge, Belgium, pages 483–488, 2004b.

J. Du, Meng Joo Er, and Leszek Rutkowski. Fault diagnosis of an air-handling unit system using a dynamic fuzzy-neural approach. In Proceedings of the 10th international conference on artificial intelligence and soft computing, pages 58–65, 2010.

J.A. Duffie and W.A. Beckman. Solar engineering of thermal processes. John Wiley & Sons, New York, USA, second edition, 1991.

P. Fairey and D. Parker. A review of hot water draw profiles used in performance analysis of residential domestic hot water systems. Technical Report FSEC-RR-56-04, Florida Solar Energy Center/University of Central Florida, 2004.

P. M. Frank, S. X. Ding, and T. Marcu. Model-based fault diagnosis in technical processes. Transactions of the Institute of Measurement and Control, 22(1):57–101, 2000.

Paul M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results. Automatica, 26(3):459 – 474, 1990.

Lisa Frantzis and Shalom Goffri. Solar water heater supply chain market analysis: Study for the city of milwaukee. Technical report, Navigant Consulting, Inc., 2010.

Jerry B. Fussell, Gary J. Powers, and R. G. Bennetts. Fault trees - state of the art discussion. Reliability, IEEE Transactions on, R-23(1):51 –55, april 1974.

Janos Gertler and David Singer. A new structural framework for parity equation-based failure detection and isolation. Automatica, 26(2):381 – 388, 1990.

S. Grossberg. Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. Biological Cybernetics, 23: 121–134, 1976a. ISSN 0340-1200.

S. Grossberg. Adaptive pattern classification and universal recoding: Ii. feedback, expectation, olfaction, illusions. Biological Cybernetics, 23:187–202, 1976b. ISSN 0340-1200.

S. Haykin. Neural Networks: A Comprehensive Foundation. Prentice Hall, 1998.

H. He, D. Menicucci, T.P. Caudell, and A. Mammoli. Real-time fault detection for solar hot water systems using adaptive resonance theory neural networks. In Proceedings of ASME 2011 5th International Conference on Energy Sustainability & 9th Fuel Cell Science, Engineering and Technology Conference, Washington DC,ESFuelCell2011-54885, 2011.

D.O. Hebb. The organization of behavior: A neuropsychological theory. New York, Wiley, 1949.

Jr. Hessian, R.T., B.B. Salter, and E.F. Goodwin. Fault-tree analysis for system design, development, modification, and verification. Reliability, IEEE Transactions on, 39(1):87 –91, apr 1990.

J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, 79(8):2554–2558, 1982.

J.M. House, H. Vaezi-Nejad, and J.M. Whitcomb. An expert rule set for fault detection in air-handling units. ASHRAE Transactions, 107:858–871, 2001.

R. Isermann. Supervision, fault-detection and fault-diagnosis methods an introduction. Control Engineering Practice, 5(5):639 – 652, 1997.

Rolf Isermann. Process fault detection based on modeling and estimation methodsa survey. Automatica, 20(4):387 – 404, 1984.

Rolf Isermann. Model-based fault-detection and diagnosis status and applications. Annual Reviews in Control, 29(1):71 – 85, 2005.

S. Kaldorf and P. Gruber. Practical experiences from developing and implementing an expert system diagnostic tool. ASHRAE Transactions, 108:826–840, 2002.

Soteris Kalogirou, Sylvain Lalot, Georgios Florides, and Bernard Desmet. Development of a neural network-based fault diagnostic system for solar thermal applications. Solar Energy, 82(2):164 – 172, 2008.

S. Katipamula, R.G. Pratt, D.P. Chassin, Z.T. Taylor, K. Gowri, and M.R. Brambley. Automated fault detection and diagnostics for outdoor-air ventilation systems and economizers:methodology and results from field testing. ASHRAE Transactions, 105:555–567, 1999.

Srinivas Katipamula, Michael R. Brambley, and Larry Luskay. Automated proactive techniques for commissioning air-handling units. Journal of Solar Energy Engineering, 125(3):282–291, 2003.

A. Kaylani, A. Al-Daraiseh, M. Georgiopoulos, M. Mollaghasemi, G.C. Anagnostopoulos, and A.S. Wu. Genetic optimization of art neural network architectures. In Neural Networks, 2007. IJCNN 2007. International Joint Conference on, pages 379 –384, aug. 2007.

J. Kilma. An expert system to aid trouble-shooting of operational problems in solar hot water systems. ASHRAE Transactions, 96:1530–1538, 1990.

T. Kohonen. The self-organizing map. Proceedings of the IEEE, 78(9):1464 –1480, sep 1990.

Teuvo Kohonen. Correlation matrix memories. Computers, IEEE Transactions on, C-21(4):353 –359, april 1972.

Hiromitsu Kumamoto, Kenji Ikenchi, Koichi Inoue, and Ernest J. Henley. Application of expert system techniques to fault diagnosis. The Chemical Engineering Journal, 29(1):1 – 9, 1984.

Won-Yong Lee, John M. House, and Nam-Ho Kyong. Subsystem level fault diagnosis of a building's air-handling unit using general regression neural networks. Applied Energy, 77(2):153 – 170, 2004.

W.Y. Lee, C. Park, J.M. House, and G.E. Kelly. Fault diagnosis of an air-handling unit using artificial neural networks. ASHRAE Transactions, 102(1):540–549, 1996.

W.Y. Lee, J.M. House, and D.R. Shin. Fault diagnosis and temperature sensor recovery for an air-handling unit. ASHRAE Transactions, 103(1):621–633, 1997.

S. Li. A Model-Based Fault Detection and Diagnostics Methodology for Secondary HVAC Systems. PhD thesis, Drexel University, 2009.

X. Li, V. Hossein, and J. Visier. Development of a fault diagnosis method for heating system using neural networks. ASHRAE Transactions, 102:607–614, 1996.

X. Li, J. Visier, and H. Vaezi-Nejad. A neural network prototype for fault detection and diagnosis of heating systems. ASHRAE Transactions, 103:634–644, 1997.

Larry Luskay, Michael Brambley, and Srinivas Katipamula. Methods for automated and continuous commissioning of building systems. Technical report, ARTI Report ARTI-21CR/610-30040-01. Arlington, Virginia: Air Conditioning and Refrigeration Technology Institute, 2003.

Mano Ram Maurya, Raghunathan Rengaswamy, and Venkat Venkatasubramanian. A systematic framework for the development and analysis of signed digraphs for chemical processes. 1. algorithms and analysis. Industrial & Engineering Chemistry Research, 42(20):4789–4810, 2003a.

Mano Ram Maurya, Raghunathan Rengaswamy, and Venkat Venkatasubramanian. A systematic framework for the development and analysis of signed digraphs for chemical processes. 2. control loops and flowsheet analysis. Industrial & Engineering Chemistry Research, 42(20):4811–4827, 2003b.

W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biology, 5:115–133, 1943.

M.G. McKellar. Failure diagnosis for a household refrigerator. Master's thesis, Purdue University, West Lafayette, Indiana., 1987.

D. Menicucci, H. He, A. Mammoli, T.P. Caudell, and J. Burch. Testing and evaluation for solar hot water reliability. Technical report, Sandia National Lab, 2011.

M. Minsky and S. Papert. Perceptrons: an introduction to computational geometry. Cambridge, Mass., MIT Press, 1969.

O. Morisot and D. Marchio. Fault detection and diagnosis on hvac variable air volume system using artificial neural network. In Proc. IBPSA Building Simulation 1999, Kyoto, Japan, 1999.

W.R. Nelson. Reactor: an expert system for diagnosis and treatment of nuclear reactor accidents. In Proceedings of the Second National Conference on Artificial Intelligence, Los Altos, California, pages 296–301, 1982.

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, 8(1):41–71, 2002.

Mario L. Ortiz. A trnsys model of a solar thermal system with thermal storage and absorption cooling. Master's thesis, University of New Mexico, 2008.

Sanjiv A. Patel and Ali K. Kamrani. Intelligent decision support system for diagnosis and maintenance of automated systems. Computers and Industrial Engineering, 30(2):297 – 319, 1996.

L.F. PAU. Survey of expert systems for fault detection, test generation and maintenance. Expert Systems, 3(2):100–110, 1986.

A. Punitha, C.P. Sumathi, and T. Santhanam. A combination of genetic algorithm and art neural network for breast cancer diagnosis. Asian Journal of Information Technology, 6(1):112–117, 2007.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6):386 – 408, 1958.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. Nature, 323:533–536, 1986.

D.E. Rumelhart and J.L. McClelland. Parallel distributed processing: explorations in the microstructure of congnition. Foundations, volume 1. MIT Press, 1986a.

D.E. Rumelhart and J.L. McClelland. Parallel distributed processing: Explorations in the microstructure of cognition. Psychological and biological models, volume 2. MIT Press, 1986b.

Jeffrey Schein and Steven T. Bushby. A hierarchical rule-based fault detection and diagnostic method for hvac systems. HVAC&R Research, 12(1):111–125, 2006.

Jeffrey Schein, Steven T. Bushby, Natascha S. Castro, and John M. House. A rule-based fault detection method for air handling units. Energy and Buildings, 38(12): 1485 – 1492, 2006.

J. Shiozaki and F. Miyasaka. A fault diagnosis tool for hvac systems using qualitative reasoning algorithms. In Proceedings of the Building Simulation 1999, 6th IBPSA conference, Kyoto, Japan, 1999.

L.A. Stallard. Model based expert system for failure detection and identification of household refrigerators. Master's thesis, Purdue University, West Lafayette, Indiana., 1989.

W. C. Swinbank. Long-wave radiation from clear skies. Quarterly Journal of the Royal Meteorological Society, 89(381):339–348, 1963.

S.A. Tassou and I.N. Grace. Fault diagnosis and refrigerant leak detection in vapour compression refrigeration systems. International Journal of Refrigeration, 28(5): 680 – 688, 2005.

B.T. Thumati, M.A. Feinstein, J.W. Fonda, A. Turnbull, F.J. Weaver, M.E. Calkins, and S. Jagannathan. An online model-based fault diagnosis scheme for hvac systems. In Control Applications (CCA), 2011 IEEE International Conference on, pages 70 –75, sept. 2011.

Venkat Venkatasubramanian, Raghunathan Rengaswamy, and Surya N Kavuri. A review of process fault detection and diagnosis: Part ii: Qualitative models and search strategies. Computers and Chemical Engineering, 27(3):313 – 326, 2003b.

Venkat Venkatasubramanian, Raghunathan Rengaswamy, Surya N. Kavuri, and Kewen Yin. A review of process fault detection and diagnosis: Part iii: Process history based methods. Computers and Chemical Engineering, 27(3):327 – 346, 2003c.

Venkat Venkatasubramanian, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. A review of process fault detection and diagnosis: Part i: Quantitative

model-based methods. Computers and Chemical Engineering, 27(3):293 – 311, 2003a.

J. Wagner and R. Shoureshi. Failure detection diagnostics for thermofluid systems. Journal of Dynamic Systems, Measurement, and Control, 114(4):699–706, 1992.

Shengwei Wang and Fu Xiao. Ahu sensor fault diagnosis using principal component analysis method. Energy and Buildings, 36(2):147 – 160, 2004.

B. Widrow and M.E. Hoff. Adaptive switching circuits. In 1960 IRE WESCON Convention Record Part IV: Computers: Man-machine Systems, pages 96–104, Los Angeles, 1960.

S. Wilcox and W. Marion. Users manual for tmy3 data sets. Technical Report NREL/TP-581-43156, National Renewable Energy Laboratory, 2008.

Alan S. Willsky. A survey of design methods for failure detection in dynamic systems. Automatica, 12(6):601 – 611, 1976.

Yonghua Zhu, Xinqiao Jin, and Zhimin Du. Fault diagnosis for sensors in air handling unit based on neural network pre-processed by wavelet and fractal. Energy and Buildings, 44(0):7 – 16, 2012.