A REAL-TIME SIMULATION TOOL FOR FAULT DETECTION AND DIAGNOSIS OF HVAC SYSTEMS

YUE MA

A Thesis

in

Department of Building, Civil & Environmental Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

January 2006

Your file Votre référence
ISBN: 0-494-14239-1
Our file Notre référence
ISBN: 0-494-14239-1

ABSTRACT

A REAL-TIME SIMULATION TOOL FOR FAULT DETECTION AND DIAGNOSIS OF HVAC SYSTEMS

YUE MA

Faults tend to degrade the performance of HVAC systems by causing occupants' discomfort, increased energy consumption, and shorter life of equipment. Accordingly, it is very important to develop fault detection and diagnosis (FDD) tools for HVAC systems.

In this thesis, in consideration of the fact that many faults could be traced back to oversized/undersized equipment, an interactive design tool is first developed to simulate the steady-state performance of a multi-zone variable-air-volume terminal reheat (VAV-TRH) HVAC system. This program is used for verifying HVAC system parameters to ensure that HVAC components are correctly sized. Then, a software program is developed to simulate the dynamic performance of a two-zone variable-air-volume terminal reheat (VAV-TRH) HVAC system. This program runs in two modes: off-line and on-line. Using this program, two control strategies – an optimal control strategy and a reheat control strategy, are studied. Simulation results show that the optimal control strategy leads to more energy savings than the reheat control strategy and is valid most of

the operation time. The program is also useful for on-line fault detection and diagnosis for the HVAC system. In order to achieve this goal, an expert rule set for on-line FDD is established using knowledge-based approach. The expert rule set, in which IF-THEN clauses are applied, consists of simple rules that can successfully identify some fairly obvious problems that are often overlooked. The simulation experiments are performed to examine the performance of expert rules using data from real-time simulations. Simulation results show that expert rules are efficient in on-line detection and diagnosis of HVAC faults.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Zaheer-uddin, for his initiating this study, his excellent guidance, and his suggestions and encouragement in all phases of this research work. Also, I appreciate the financial support.

Special thanks are given to my colleagues, for their helpful discussions and constructive suggestions in my study. My sincere appreciations also go to my father. It was he who encouraged me to enter into the thesis option.

Finally, I would like to dedicate this thesis to my wife, Wei, and my lovely daughter, Jie Yi. Without their patience, continual encouragement, and moral support, I would have never finished this study.

Table of Contents

List of Figures

List of Tables

Nomenclature

A	total surface area of the fin and base (ft^2)
A_d, A_{duct}	area of duct walls (m^2)
A_f	fin surface area (ft^2)
A_i	total water-side heat-transfer area (ft^2)
A_o	total air-side heat-transfer area (ft^2)
A_t	area of bare tubes without fins (ft^2)
C_a	air capacitance rate (Btu/hr-F)
C_m	total thermal capacitance of coil material (Btu/hr-F)
$C_{\max}$	maximum of C_a and C_w (Btu/hr-F)
$C_{\min}$	minimum of C_a and C_w (Btu/hr-F)
c_{pa}	specific heat of air (Btu/lb_m-F)
C_w	water capacitance rate (Btu/hr-F)
c_w, c_p	water specific heat (Btu/lb_m-F)
CFM_{oad}	outside air intake for design conditions (cfm)
CFM_{sad}	supply airflow rate for design conditions (cfm)
D_i	tube inside diameter (ft)
D_t, D_o	outside tube diameter (in.)
E_{ch}	chiller power (Hp)
E_{rf}	return fan power (Hp)

E_{sf}	supply fan power (Hp)
E_t	total power (Hp)
e	error used for PI controller
G_c	air mass velocity referred to the minimum flow area inside the coil (lbm/hr-ft^2)
H_{da}	enthalpy of discharge air (Btu/lb$_m$ dry)
H_{re}	reheat load (Kw)
H_z	enthalpy of zone air (Btu/lb$_m$ dry)
h_i	water-side heat-transfer coefficient (Btu/hr-ft^2-F)
h_o	air-side heat-transfer coefficient (Btu/hr-ft^2-F)
j	Colburn heat transfer j-factor
j_4	Colburn heat transfer j-factor for 4-row cooling coil
k	factor describing the thermal capacity of zone
k_p	proportional gain
k_i	integral gain
$\dot{k}$	thermal conductivity of the fin material (Btu-ft)/ (hr-ft^2-F)
$\bar{k}$	air thermal conductivity (Btu/ft-F)
$\hat{k}$	water thermal conductivity (Btu/ft-F)
$M_{as}, M_a, \dot{m}_a$	supply air mass flow rate (kg/s)
$MinR_{oa}$	minimum ratio of outdoor air intake to supply airflow rate
$\dot{m}_{ea}$	exhaust air mass flow rate (lb$_m$/hr)

$\dot{m}_{oa}$	outside air mass flow rate (lb_m/hr)
$\dot{m}_{oad}$	outside air mass flow rate for design conditions (lb_m/hr)
$\dot{m}_{ra}$	recirculated air mass flow rate (lb_m/hr)
$\dot{m}_{rad}$	recirculated air mass flow rate for design conditions (lb_m/hr)
$\dot{m}_{sa}, \dot{m}_a$	supply air mass flow rate (lb_m/hr)
$\dot{m}_{sad}$	supply air mass flow rate for design conditions (lb_m/hr)
$\dot{m}_w$	chilled water mass flow rate (lb_m/hr)
$\dot{m}_{wd}$	chilled water mass flow rate for design conditions (lb_m/hr)
N_r	number of rows
Pr	Prandtl number (1/h)
Q	airflow rate delivered by fan (cfm)
Q_s, q_s	instantaneous sensible load (Kw or Btu/hr)
Q_{sd}	peak sensible load (Kw or Btu/hr)
Q_{tc}, Q_{td}	peak total load (Kw or Btu/hr)
R	ratio of minimum to maximum capacitance
R_{ct}	unit contact resistance (hr-ft^2-F/Btu)
R_f	fouling factor (hr-ft^2-F/Btu)
R_l	factor to describe predicted leakage rates for ducts
R_{osact}	actual ratio of outdoor air to supply airflow rate

$R_{os\,\exp}$	expected ratio of outdoor air to supply airflow rate
Re_{Di}	Reynolds number based on the tube inside diameter
Re_{Do}	Reynolds number based on the tube outside diameter
Re_{s}	Reynolds number based on fin spacing
Re_{Xb}	Reynolds number based on X_b
RH_{da}	relative humidity of discharge air
RH_{z}	relative humidity of zone air
r	tube outside radius (in.)
s	fin spacing (in.)
T_{ai}	entering coil air temperature (F)
T_{ao}	air dynamic outlet temperature (F)
T_{aoss}	steady state air outlet temperature (F)
T_{br}	temperature of air entering VAV box (°C)
T_{da}	discharge air temperature (F)
T_{daset}	discharge air temperature setpoint (F)
T_{ma}	mixed air temperature (F)
T_{oa}	outdoor air temperature (F or °C)
T_{ra}	return air temperature (F)
T_{sa}	supply air temperature (F)
T_{z}	zone temperature (F or °C)
T_{zset}	zone temperature setpoint (F)

T_{wi}	inlet water temperature (F)
T_{wo}	water dynamic outlet temperature (F)
T_{woss}	steady state water outlet temperature (F)
t	time (hour)
t_i	temperature of air entering ducts (^{o}C)
T_{surr}, t_r	surrounding room temperature (^{o}C)
U_{ea}	exhaust air damper position [0, 1]
U_{oa}	outside air damper position [0, 1]
U_o	controller output with no error
U_{ov}	overall heat-transfer coefficient, based on the air-side area (Btu/hr-ft^2-F)
U_{ra}	recirculated air damper position [0, 1]
U_{trh}	reheat coil valve position
U_{wact}	actual control signal to the cooling coil valve
$U_{w\exp}$	expected control signal to the cooling coil valve
V_{th}	threshold percentage of supply airflow rate
V_w	water velocity (fpm)
V_z	zone air volume (ft^3)
W_{sh}	fan shaft horsepower (Hp)
X_b	longitudinal tube spacing (in.)
y	fin thickness (in.)

ΔP	total pressure difference across fan (in.WG or KPa)
ΔT_d	temperature rise in ducts (^{o}C)
ΔT_f	temperature rise across fan (F or ^{o}C)

Greek letters

ρ	water density (lb/ft^3)
ρ_a	air density (lb_m/ft^3)
τ	time constant (seconds)
ε	heat exchanger effectiveness
η	fin efficiency
η_s	surface effectiveness
η_f	fan total efficiency
μ	air dynamic viscosity (lb/ft-sec)
μ_w	water dynamic viscosity (lb/ft-sec)

CHAPTER 1 INTRODUCTION

1.1 Introduction

Faults in HVAC systems can be grouped into two categories: *operational faults* and *component/equipment faults*. The former type of faults could be due to poor tuning of controllers, incorrect setpoints, and contradictory time scheduled operations. The latter type of faults includes stuck or leaky dampers and valves, broken sensors or actuators, scale build-up on coils, and etc. Detecting equipment faults is somewhat easier than the operational faults. It is well known that the presence of faults in the HVAC system could increase energy consumption, shorten equipment life, and lead to thermal discomfort in buildings. To this end, several researchers have developed fault detection and diagnosis (FDD) tools for HVAC systems (House and Shin 1999, Han et al. 1999, Katipamula et al. 1999).

Generally, two approaches are used to develop FDD systems. The first method uses a model-based approach. In this method, the system model must reflect the real system's behavior under all normal operating conditions. Model-based approach compares the output of a model of fault-free operation with actual conditions. By defining threshold levels of differences, the presence of a fault is detected. Knowledge-based approaches, also referred to as expert systems, are also used in HVAC applications. Most

knowledge-based approaches use IF-THEN rules. The expert knowledge base is used to diagnose and isolate faults.

So far, numerous studies report the use of knowledge-based approaches to diagnose faults in HVAC components and systems. These studies have used both off-line simulations and experimental techniques to assess specific faults. However, what are lacking are on-line FDD tools that can simulate real-time HVAC system operation, so that HVAC system operators can use such tools to simulate faults and study its impact on system outputs and therefore operate the system more efficiently. To this end, the following methodology is used in this thesis. First, a dynamic model of a variable-air-volume terminal reheat (VAV-TRH) HVAC system is developed with a view to use it as a platform for both off-line and real-time simulation tool for assessing the dynamic performance of the HVAC system. Second, HVAC control strategies are designed to study the impact of control strategies on the system performance. Third, expert rules for on-line FDD, some of which are adapted from the literature, are established. Fourth, a real-time FDD software platform is developed. Finally, expert rules are tested in real-time and the corresponding results are analyzed.

1.2 Objectives

The major objective of this thesis is to develop a real-time monitoring, control, and fault detection system in Matlab/Simulink environment useful for on-line fault detection and

diagnosis (FDD) and for analyzing optimal control strategies for VAV HVAC systems.

The following tasks will be carried out in order to achieve the above objectives:

1. To establish a dynamic model for a two-zone VAV-TRH HVAC system,
2. To develop a software program for the model using MatLab/Simulink environment,
3. To design HVAC control strategies for the operation of the system,
4. To develop a real-time FDD tool by adapting from the literature common faults in the HVAC system, and
5. To test the expert rules by performing real-time simulations and analyzing the results.

1.3 Organization

This thesis is divided into seven chapters. A review of literature is conducted in Chapter 2. In Chapter 3, a software program is developed in Visual C++ 6.0 environment to simulate steady-state performance of multi-zone VAV-TRH HVAC systems. In Chapter 4, the dynamic model of a two-zone VAV-TRH HVAC system is developed. Also, PI control strategies for the VAV system are used to simulate closed loop performance of the system. In Chapter 5, the development process of software program to perform off-line and on-line simulations is explained in detail. Several cases of off-line simulations and real-time experiments are also performed to study the dynamic behavior of the HVAC system. In Chapter 6, expert rules for on-line fault detection and diagnosis are defined. Results from the application of the rules to data from real-time simulations are presented and analyzed. Chapter 7 summarizes conclusions and provides recommendations for future work.

1.4 Contributions

The work presented in this thesis will help

1. HVAC designers double-check whether or not the equipment is correctly sized,
2. control engineers see whether or not the PI controller is correctly tuned,
3. building administrator implement appropriate operation strategies, and
4. facility staff detect and diagnose HVAC faults on-line.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

Buildings consume one-third of all energy use and therefore it is important to consider energy-efficient concepts to reduce energy consumption. The topic of buildings' energy conservation is multidisciplinary in nature. From the point of view of HVAC engineering, the methods to reduce building energy consumption mainly include (1) *equipment retrofits to improve the energy efficiency of heating and cooling components*, (2) *utilization of thermal storage to provide free cooling and reduce peak demand*, (3) *recommissioning of the HVAC system to rectify faulty operation*, and (4) *application of control systems*. The HVAC control strategy is the most frequently used method for saving building energy use. It is well accepted that energy use can significantly be reduced by applying HVAC control systems. Two approaches can be used for HVAC control strategies: *local process oriented approach* and *system oriented approach*. The control system based on the first approach is referred to as local loop control, which remains quite popular today. The local loop control mainly includes (1) *two-position (or on-off) control*, (2) *proportional control*, (3) *proportional-integral control*, and (4) *proportional-integral-derivative control*. All of these control schemes are effective. However, the local loop control is simply individual component control, failing to take advantage of the interaction of the system variables. Accordingly, maximum energy savings cannot be achieved. Increased concerns for the

production of energy and popular recognition that energy resources are not infinite are two important motivations to develop better HVAC control systems to conserve building energy use to the greatest extent. This goal can be achieved using the optimal control strategy based on system oriented approach. As the name indicates, system oriented approach considers the HVAC system as a whole. Compared to local loop approach, system oriented approach takes into accounts the interactive nature of the HVAC components, the building system, and their associated variables so that it can be used to implement optimal control strategy. According to House and Smith (1995), the system oriented control strategy has four advantages. First, more energy savings can be acquired in comparison to a local loop control. Second, it has the capability to accommodate a variety of system conditions, including time-varying loads, occupancy schedules, system constraints, and different setpoints in different zones. Third, it is flexible to allow the importance of comfort versus energy use to be adjusted based on the particular needs of a given HVAC system. Fourth, it can be implemented using the zone air and cooling coil temperature responses as setpoint values for local feedback controls. In summary, optimal control scheme can be implemented using system oriented approach.

The heating, ventilating, and air-conditioning (HVAC) system is a complex electromechanical system. In such a system, faults appear virtually everywhere, develop rather quickly or very slowly, and present different kinds of symptoms under different conditions. Faults in HVAC systems comprise a wide range of problems. The outcome

caused by HVAC faults may be categorized into three classifications: (1) *increased energy consumption*, (2) *worn equipment*, and (3) *less comfortable conditions*. However, one point that is often ignored is that optimal control strategies can be partially or completely disabled by faults. When this happens, optimal control strategy will not save energy as expected. In addition, optimal control strategy can also be used to detect HVAC faults (Pape et al. 1991). Therefore, the optimal control has a great deal to do with fault detection and diagnosis.

To this end, this literature review attempts to summarize the developments in two aspects: (1) *optimal control strategies*, and (2) *fault detection and diagnosis*.

2.2 Optimal Control Strategies

An optimal control strategy is one by which, for given building loads and ambient conditions, the total energy requirement for an HVAC system is minimized while maintaining acceptable comfort conditions in the building. The total energy use is the sum of all the power required to run HVAC components, which mainly consist of (1) *chillers*, (2) *boilers*, (3) *cooling towers*, (4) *chilled water circulating pumps*, (5) *condenser water circulating pumps*, and (6) *fans*.

The implementation of the optimal control strategy for an HVAC system requires detailed knowledge of the performance characteristics of every HVAC component. Numerous studies have shown that optimal control strategies based on system oriented approach lead to improved system responses and reduced energy use compared to local loop control. Moreover, further improvements can be achieved when multiple control variables are optimized simultaneously. Kaya et al. (1982) developed an optimal model that included a linearized dynamic model for a thermal space and an associated cost function. The improvement in control and reduction in energy consumption from controlled temperature, humidity, and air velocity simultaneously rather than individually were demonstrated. The optimal control solution consisted of static and dynamic optimization. Their findings indicated that the proposed control system, in which the interaction of control variables was considered, brought about reduction in energy use.

Nizet et al. (1984) developed an optimal control method for a simplified dynamic model for one floor of a typical multi-story building. The objective of the study was to find the optimal air-conditioning policy. It was implemented by minimizing a cost function, which was composed of energy costs for HVAC components and a thermal comfort penalty. This function was solved using a conjugate gradient method. At last, it was found that energy savings of 12% to 30% were achieved using the optimal control policy.

Cumali (1988) presented a practical case of energy efficiency retrofit for a high-rise building complex. The author faced the dilemma where the cost of controls alone would have far exceeded the total budget for the retrofit project if all terminal devices had been converted to variable-air-volume controls. He proposed a control system where enough sampling points were used to identify the interactions among the building systems and then applied this information to the real-time simulations of operation to select control strategies that were near optimal with respect to energy consumption. It was concluded that the case of partial loads provided more opportunities for energy savings relative to that of peak loads. For low loads, the limiting factors were not the fan and pump capacity limits, but the low limit on chilled water temperature and the high limit on supply air temperature.

Braun et al. (1989a) developed a methodology to determine optimal control that utilized component performance information from measurements, manufacturer's data, and mechanistic models. This information was combined into a representation of the performance of the entire system. Then the optimal strategy was determined using nonlinear optimization techniques. The optimal set of control variables that minimized energy consumption at any time was found. It seemed feasible to utilize detailed performance data for each of the HVAC components, but the information requirement was extremely large. Considering this difficulty, Braun et al. (1989b) also developed a methodology for determining the so-called 'near-optimal' control strategy based on the

power consumption of the system as a whole rather than that of every component. The near-optimal control strategy was simpler to determine and a set of control variables that was very close to the true optimal set was found. The basis for the near-optimal control strategy was the use of quadratic relations for power consumption. In the follow-up study, Braun (1990) noted that the state of the building thermal capacitance could be controlled by dynamically adjusting the zone temperature setpoints over time within the thermal comfort region. Therefore, building thermal storage was used as a means of reducing the operation costs while maintaining adequate comfort conditions in buildings. The dynamic optimization techniques were used to determine the minimum operation costs assuming that future ambient conditions and internal gains were known in advance. Finally, it was concluded that both energy costs and peak electrical demand could be significantly reduced through optimal control of the intrinsic thermal storage within building structures.

House et al. (1991) regarded a cost function as an important feature of the optimal control scheme. The cost function consisted of costs associated with energy consumption and a cost penalty for the zone temperature not being at the setpoint. An optimal control methodology that emphasized an overall system approach to the optimal control problem was developed. The solution approach, which was referred to as DTM (discrete time method), involved discretizing the optimal control problem in time and then determining optimal values of the control variables at discrete times using a nonlinear optimization technique. It was concluded that the optimal control scheme produced significantly lower

total costs and a better system performance. In the follow-up study, House and Smith (1995) refined the optimal control strategy and made it much closer to the practical engineering. First, an HVAC system with five zones subjected to time-varying loads and occupancy schedules was used. Second, constraints were imposed on the system variables like upper and lower bounds on the discharge air temperature. Finally, the optimal control strategy was based on maintaining the predicted mean vote (PMV) rather than the dry-bulb temperature within a comfort range.

So et al. (1995) stated that local loop HVAC control system had slow response rates and long transient period might bring about extra energy consumption. The new technology in computer vision was employed to improve the performance of HVAC control systems. This computer vision system served two purposes. First, it counted the number of occupants within an air-conditioned space, on the basis of which the optimal quantity of supply air could be adjusted to reduce the response time so that energy savings would be achieved. Second, it communicates the HVAC control system of the distribution of occupants so that real-time zone control became possible. Therefore, the thermal loads of different zones within the conditioned space could be estimated so that the supply airflow rate would be adaptively controlled. However, the computer vision system used local loop control rather than system oriented approach because it only minimized energy use of air-handling unit (AHU) instead of that of the entire HVAC system.

Li et al. (1996) conducted researches in minimizing the energy cost associated with the control of VAV system. A dynamic VAV HVAC system was modeled and the energy consumption characteristics were analyzed. Experimental work was performed using a single-zone VAV system. Conventional PI, adaptive control, and optimal control algorithms were applied to the duct pressure control loop within the VAV system. It was stated that the optimal control strategy resulted in energy savings of up to 30% in comparison to a conventional PI control scheme.

2.3 HVAC Fault Detection and Diagnosis

It is extremely important for an HVAC system to have fault detection and diagnosis (FDD) tools. To achieve this goal, several classification techniques are employed, which include (1) *artificial neural network classifiers* (ANN), (2) *nearest neighbor classifiers*, (3) *nearest prototype classifiers*, (4) *a rule-based classifier*, and (5) *a Bayes classifier*. House and Shin (1999) summarized the characteristics of the above classifiers. The Bayes classifier was a good choice for fault detection. It was a straightforward method with limited storage requirement and less computational effort. A Bayes classifier minimized the cost or the probability of misclassification. In terms of minimizing classification errors, the Bayes classifier is considered to be a good choice. The rule-based classifier was a favorable and effective method for fault diagnosis when the various classes of faulty operation were distinct and could be distinguished by a single dominant symptom or feature. Artificial neural networks (ANNs) are powerful tools for mapping inputs to

outputs, especially when this mapping is nonlinear. For classification purposes, a feedforward ANN could be trained to produce a specific output pattern for a specific input pattern. ANNs are effective tools for pattern recognition. However, since ANNs basically function like black boxes, as such the reasoning behind decisions is difficult to understand.

It has been found during the literature review that artificial neural network (ANN), Bayes, and rule-based classifiers are most used for fault detection and diagnosis.

2.3.1 Model-based approach

Model-based approaches for fault detection and diagnosis are commonly employed in the literature, with most reporting on the application of a particular FDD method to simulation data. Both off-line and on-line fault detection and diagnosis may be implemented using model-based approach.

Pape et al. (1991) developed a methodology for fault detection in HVAC systems based on the use of a relation for the system power consumption under the optimal control strategy. The optimal control strategy was determined using information from an energy management and control system (EMCS). Once deviations were found by comparing the measured system power with the power predicted with the optimal control strategy, faults were considered to be occurring. Then individual relations for each power-consuming component were used for determination of the location of faults. However, this

methodology simply detected faults for power-consuming HVAC components.

Peitsman and Bakker (1996) described the application of black-box models for fault detection and diagnosis in HVAC systems. Two levels of black-box model were required: *system models* and *component models*. The system model was used to detect faulty behavior of an HVAC system, and component models were used to locate the defective component. The reason that the authors used black-box models rather than white-box models was that black-box models required less physical knowledge of the process. However, there existed a risk of users not accepting a classifier based on black box models.

Lee et al. (1997) described the use of a two-stage artificial neural network for fault diagnosis in a simulated air-handling unit (AHU). The AHU was classified into four subsystems: (1) *pressure control subsystem*, (2) *flow control subsystem*, (3) *cooling coil subsystem*, and (4) *mixing box damper subsystem*. In the first stage neural network was trained to identify the subsystem in which a fault occurs while in the second stage neural network to diagnose the specific cause of a fault at the subsystem level. Two prerequisites existed if ANN was used for fault diagnosis. First, the data had to be filtered by a steady-state detector, which included three variables: *supply air temperature*, *supply air pressure*, and *flow rate difference between the supply and return air ducts*. Second, residuals were normalized so that the dominant symptom residuals had approximately the

same magnitude for different fault causes.

Li et al. (1997) developed an artificial neural network (ANN) prototype for fault detection and diagnosis in complex heating systems. The prototype was implemented using the daily values obtained by a preprocessing procedure of the simulation data. Six different types of fault could be detected: two related to the gas-fired boilers, one to the control valve, and three to the control system. The study concluded that single ANN presented better performance compared to multiple ANN. A reliable database describing the behavior of heating systems in faulty and nonfaulty cases is needed to train the neural networks. Such a database is often difficult to obtain.

Model-based techniques have also been used to detect faults in refrigeration equipment. Grimmelius et al. (1995) developed a real-time monitoring and diagnostic system for compression refrigeration plants. Several symptoms of failure modes were defined. Normal performance of refrigeration equipment was predicted using a regression analysis model. The symptoms were deviations of selected variables from the predicted values. Then these symptoms were analyzed to identify fault patterns using a fuzzy classifier.

Peitsman and Soethout (1997) applied auto aggressive (ARX) models in real-time model-based diagnosis. Two hierarchical levels of modeling - *the system level* and *the component level* - were required for an HVAC system. The system level involved a system

model considering the HVAC system as a whole. The component level involved all the models of the individual components. The system model was used to detect faults in the system based performance degradation. After a fault was detected, the component models were used to locate the defective component. The shortcoming of this approach was the slow speed caused by complexity at the component level.

Han et al. (1999) presented a model-based fault detection and diagnosis tool for HVAC systems. Various issues of real implementation of the system and the processing of real-time on-line data in actual systems were discussed. On-line monitoring of HVAC system performance, detection of abnormalities, and diagnosis of system faults were realized. A pretest was recommended to perform to make sure that all the sensors were correct before using rules to diagnose the HVAC system.

2.3.2 Knowledge-based approach

Knowledge-based approaches are also known as expert systems. Expert systems are one field of application for the advanced programming techniques. They are computer programs that solve problems difficult enough to require human expertise using a previously assembled knowledge base and inference procedures (Barr and Feigenbaum 1982). Expert systems in HVAC applications have been widely used in recent years. The expert system provides a valuable service for both building administrators and maintenance personnel. To be exact, administrators can plan future energy consumption

budgets and project maintenance schedules. Maintenance personnel can plan their work and perform the necessary preventative maintenance, as recommended by the knowledge-based system.

Haberl and Claridge (1987) listed six important characteristics of expert systems. They include (1) *diagnostic capabilities*, (2) *on-line training capabilities*, (3) *user friendly*, (4) *easy to install*, (5) *easy to maintain*, and (6) *easy to update*. Also, a prototype expert system, which was referred to as Building Energy Analysis Consultant System, was developed. This system had two main components: *an energy consumption predictor* and *an expert system*. The predictor used the regression techniques to determine energy consumption. Then this information was passed to an expert system that analyzed daily energy consumption patterns, diagnosed abnormal consumption, and suggested possible causes.

Norford and Rabel (1987) used measured data to establish relationships between chiller efficiency and chiller load and between fan power and airflow. Duct temperature and flow data were also used to establish excessive or deficient outdoor air intake into the building. Control logic faults, such as chilled-water pumps in prolonged operation with chillers off, were detected. Fault detection was made using visual inspection and a rule-based system. In another study, Norford and Little (1993) adopted a steady-state approach to detect faults in a VAV HVAC system. They divided faults into four categories: (1) *failure to maintain*

supply air temperature setpoint, (2) *failure to maintain supply air pressure setpoint*, (3) *increased pressure drop*, and (4) *problems associated with the fan motor, coupling to the fan and fan capacity controls*. The methods of correlating fan power with independent variables – space load, supply air temperature setpoint, supply airflow rate, and fan speed control signal – were described.

Katipamula et al. (1999) developed a diagnostic system that automatically detected and diagnosed problems with outdoor air ventilation and economizer operation in commercial buildings using data available from building automation systems. The system also provided suggestions for correcting the problems. The diagnostics was based on rules derived from engineering models of proper and improper air-handler performance. However, this system had two disadvantages. First, it could not be used for the entire HVAC system because it simply detected ventilation and economizer operational problems. Second, it was impractical to assume that outside air intake was a constant fraction of the supply airflow rate.

Visier et al. (1999) presented a method to detect the most common faults occurring in the hydronic space heating systems in school buildings. This software, which was called EMMA (energy management at the municipal level), was composed of a preprocessor and a classifier. The preprocessor, which was based on a good analysis of the system, input hourly measurements of outdoor air temperature, indoor air temperature and hot water

supply temperature. The classifier based on if-then rules was designed to perform pattern recognition.

Similar to model-based approach, knowledge-based approach was used to detect faults in refrigeration equipment. Stylianou and Nikanpour (1996) used knowledge-based approach to diagnose chiller faults based on the start-up temperature transients. A steady-state module, in which if-then rules were employed, was used to detect and diagnose faults. In the follow-up study, Stylianou (1997) substituted the rule-based steady-state module with a statistical pattern recognition algorithm (SPRA). The SPRA provided an effective way of classifying patterns in multivariable, multiple problems encountered in commercial chillers without having to explicitly use a rule-based system.

Knowledge-based approaches may also be used for FDD of real-time systems. Anderson et al. (1989) developed a quasi-real-time expert system. The expert system consisted of two distinct software components – *a statistical analysis preprocessor* and *a rule-based expert system*. The system provided an hourly diagnostic analysis of an industrial HVAC system. Monitoring and diagnostic functions were performed on data collected by the existing data acquisition system. The statistical analysis preprocessor compared incoming data with predictors and computed a normalized variance for each channel of data. The data were searched for inconsistencies in order to find sensor failures. The variances were compared with set limits within the expert system as a check on consistent system

operation. It was believed that the greatest advantage of this system was its ability to perform HVAC diagnostics as a human expert would on a continuous basis. The heart of this system was predictor, which was created by regressing the data against independent variables that affected the building performance.

Dodier et al. (1998) described an automated fault detection and diagnosis scheme for HVAC equipment. This diagnostic system was to report briefs about the physical system of interest. Bayer's classifier was used to predict the state of operation of a fan-powered VAV box. That is, expert knowledge was interpreted as probability functions on the input space and these functions were used to compute probabilities (degrees of belief) of states of the physical system. Combining the probabilities with information about the costs of taking actions, facility staff decided which action to take. The algorithm included two steps. First, the probability that each possible system state occurred was estimated. Second, of all the possible actions, the action with the least risk was taken.

From the literature, it has been found that most knowledge-based systems use if-then rules. Jiang et al. (1995) stated that there was no way to accurately determine the threshold values that were required for if-then rules. A new idea, which was referred to as *fault direction space method*, was proposed to overcome this problem. First, a group of characteristic parameters (CP) was established according to the physical model of the component to be detected. The CP was determined from the measured data and kept

constant during operation at the faultless state. When a fault occurred, the CP would deviate from the normal value. Second, to classify faults, relationships between the deviations of each CP were employed in a normal reasoning procedure. A matrix was used to present fault direction space so as to locate the defective component.

2.4 Summary

This literature review indicates two points. First, significant reduction in energy use can be achieved without loss of comfort through the use of optimal control strategies. Second, a number of faults of the HVAC system can be detected using *fault detection and diagnosis (FDD) tool*. As indicated at the beginning of this chapter, the optimal control is closely associated with FDD. Studies combining FDD with control strategies appear to be lacking. Among the papers surveyed, only Pape et al. (1991) developed a methodology for fault detection in HVAC systems based on the optimal control. However, this methodology could not detect faults for all HVAC components.

The literature review also presents that in recent years, on-line expert system employing if-then rules for FDD has attracted more and more interest. The if-then rules are very simple and can be easily applied manually by an operator. The use of if-then rules is rather efficient because it can give a clear explanation to the fault detection.

For on-line system, if-then rules are applicable because of its simplicity. On the other hand, if-then rules can lead to false alarms in some cases. It is then necessary to add new rules. The main risk of if-then rules will probably consist in adding too many new rules to deal with specific cases. In that case, there could be a risk of losing one of the main advantages of the method - its simplicity (Visier et al. 1999). Despite these drawbacks, if-then rules are attractive because it is relatively easy to take a diagnostic output and trace backward through the rules to understand how that conclusion has been reached. To the end-user, this is a desirable attribute of an FDD tool.

The existing methods focusing on on-line expert system using if-then rules for FDD have two limitations. First, some common faults occurring in HVAC systems have not been defined in the literature. Second, the set of expert rules that can be used for on-line fault detection and diagnosis for an HVAC system has been far from complete.

Accordingly, in this thesis, some of the above limitations will be addressed. In addition, the control strategies will be studied. On the basis of these tasks, a real-time monitoring, control, and fault detection system will be developed. This system is useful for on-line fault detection and diagnosis, and as well, for analyzing optimal control strategies for VAV HVAC system.

CHAPTER 3

STEADY STATE SIMULATIONS OF HVAC SYSTEMS

In diagnosing HVAC system operations to detect faults, an important aspect not to be overlooked is to verify if the components of the HVAC system are properly selected and sized. This is due to the fact that many of the operational faults could be traced back to oversized/undersized equipment. To this end, the major objective of this chapter is to develop an interactive design tool to verify HVAC system parameters, on the basis of which HVAC components can be correctly sized. In this chapter, a software application used for analyzing steady state performance of multi (up to five)-zone variable-air-volume terminal reheat (VAV-TRH) HVAC system is developed.

This chapter consists of four sections. A software design and application, which is developed to perform steady-state simulations of HVAC systems, is introduced in Section 1. Section 2 elaborates the algorithm of the software program. Section 3 presents simulation results of a two-zone VAV-TRH HVAC system. Summary and conclusions are listed in Section 4.

3.1 Software Profile

The software program, *VAV Simulator*, is implemented in Visual C++ 6.0 environment and listed in *Appendix A*. It can be used to perform steady-state simulations of any multi-zone (up to five zones) VAV-TRH HVAC systems. Object oriented methodology was used to develop the software program. This method is explained in detail in Chapter 4 where another program, *HVAC Simulator*, is developed.

3.1.1 Input Windows

Two kinds of input windows were used in this application. The first type, shown in **Fig. 3-1**, was known as "*General information*" window, which meant that the parameters entered in this window were common for the entire HVAC system. Such parameters included outdoor air design conditions, chilled water supply temperature, and etc. The second type, shown in **Fig. 3-2**, was called "*Inputs of zone*" window, where parameters of the individual zones were entered such as indoor air design conditions and instantaneous space loads. There was only one "*General information*" window but there could be more than one "*Inputs of zone*" window. The number windows will be equal to the entered digit in the *Number of zones of interest* "*General information*" window.

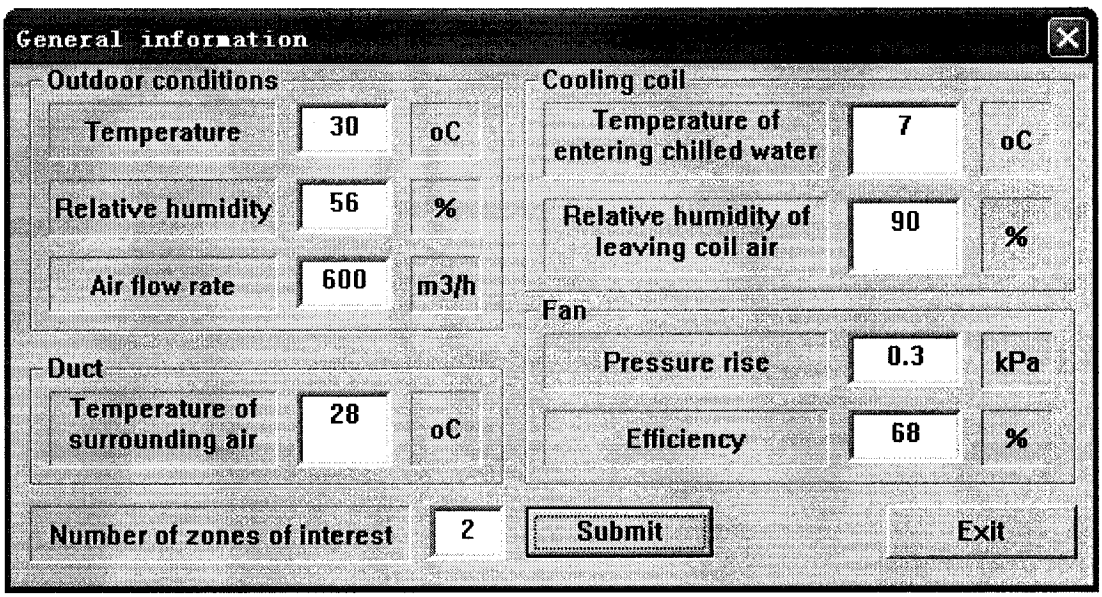


Figure 3-1 General information window

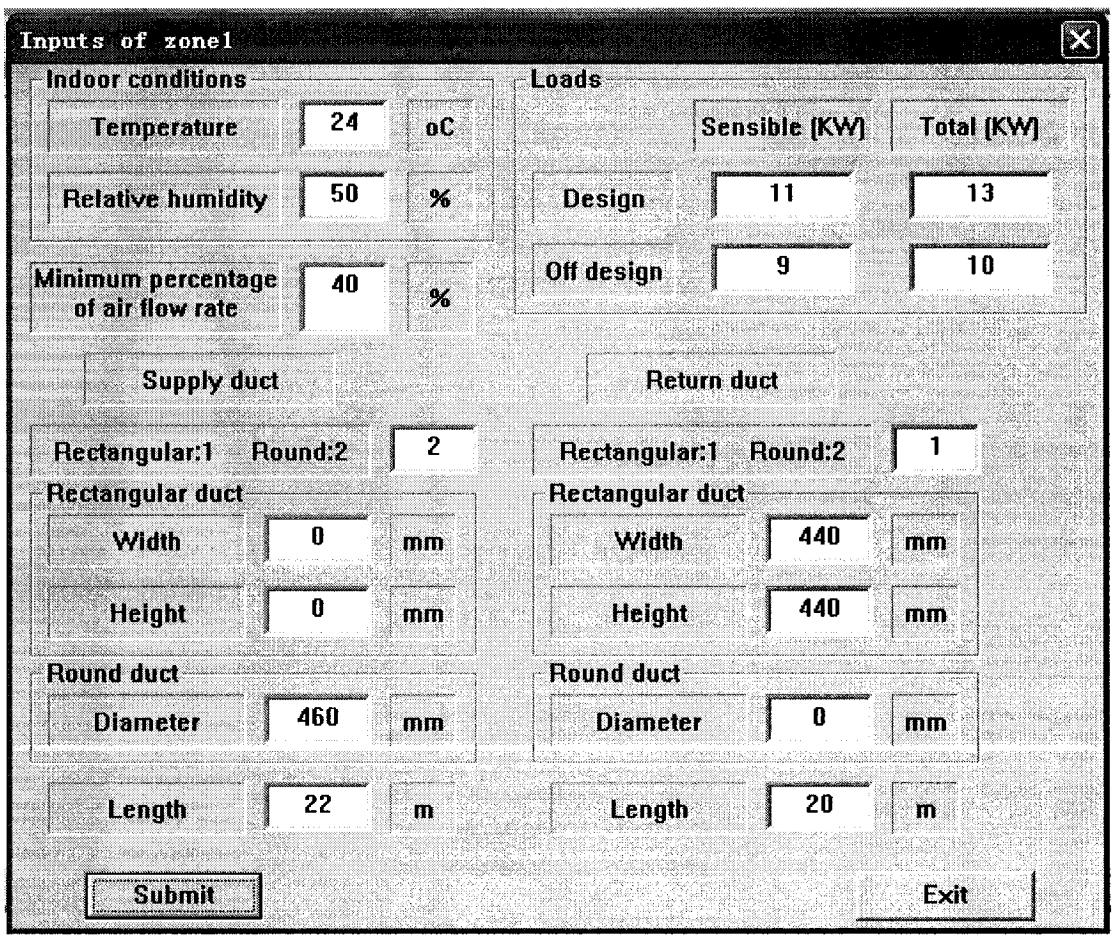


Figure 3-2 Inputs of zone window

3.1.2 Output Windows

Similar to input windows, there were two kinds of output windows. The first kind, shown in **Fig. 3-3**, is known as "*Outputs of general results*" window, which meant that the result parameters in this window were common for the entire HVAC system. Such parameters consisted of discharge air temperature, total supply airflow rate, and etc. The second kind, shown in **Fig. 3-4**, was called "*Outputs of zone*" window, where results of the individual zones were displayed like supply air temperature and supply airflow rate. Also, there was only one "*Outputs of general results*" window but could be more than one "*Outputs of zone*" window. The number of "*Outputs of zone*" window was the same as that of "*Inputs of zone*" window.

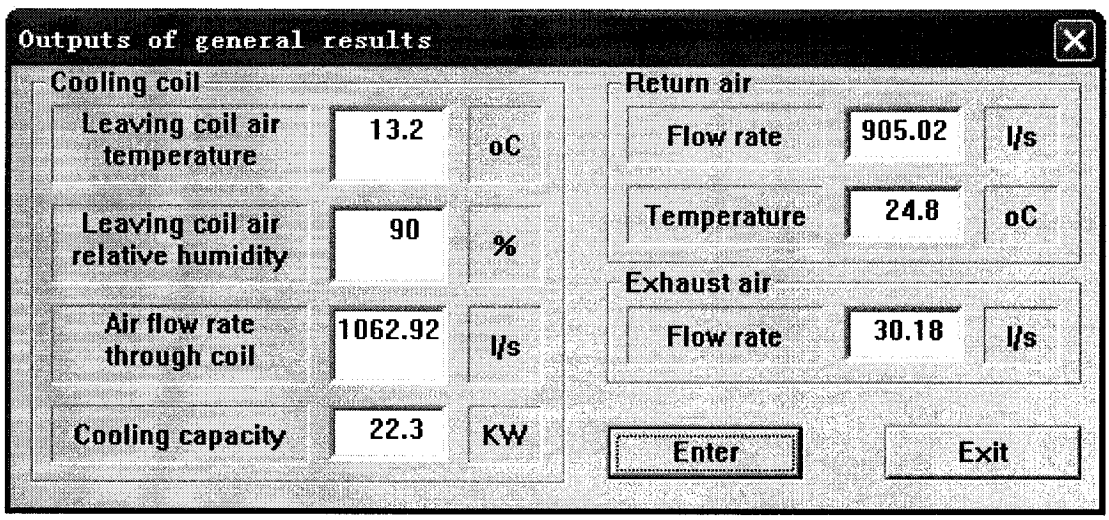


Figure 3-3 Outputs of general results window

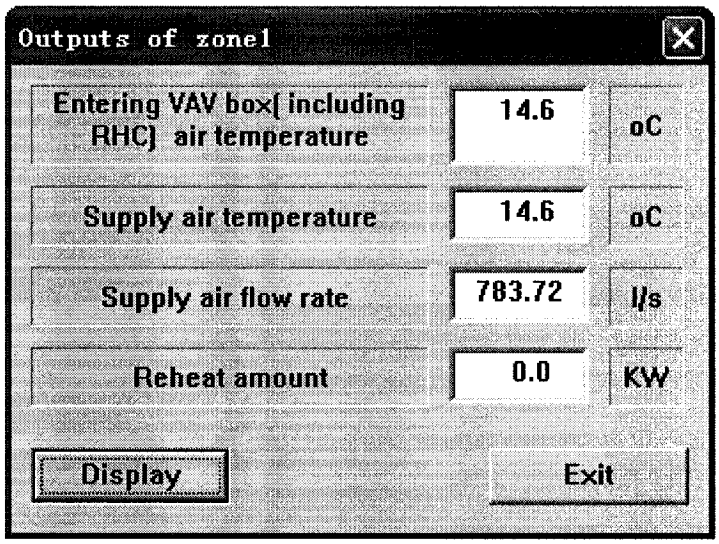


Figure 3-4 Outputs of zone window

3.2 Software Algorithm

3.2.1 Determination of Discharge Air Temperature

In this module, the discharge air temperature for a given set of conditions is determined. **Fig. 3-5** shows the algorithm for determining discharge air temperature. For a given set of peak sensible and total loads (Q_{sd} and Q_{td}), design indoor temperature and relative humidity (T_z and RH_z), relative humidity of air leaving the coil (RH_{da}), and chilled water supply temperature (T_{wi}), the discharge air temperature (T_{da}) matching the above operating conditions is determined. The algorithm is initiated by assuming a trial value of T_{da}. Then, two objects, *IndoorAir* and *DischargeAir*, are established using the *Class MoistAir*, on the basis of which enthalpies of indoor air and discharge air (H_z and H_{da}), and total supply air mass flow rate (M_{as}) are determined. The computed total load (Q_{tc}) is compared with the given total load (Q_{td}). If both are sufficiently close, i.e., the value of '*eh*' is very small, the discharge air temperature (T_{da}) is considered to have converged very close to the exact

solution. If not, the computed discharge air temperature (T_{da}) is updated by adding or subtracting a small value ($\triangle T$) and another iteration is performed until the converge is achieved.

3.2.2 Determination of Supply Airflow Rates and Reheat Loads

This process consists of three steps. First, air mass flow rates supplied to the individual zones based on space loads are determined. In this step, reheat is not considered. Then, the computed supply airflow rates are compared with the prescribed minimum airflow rates in order to check whether or not reheat is required. Finally, the appropriate values of supply airflow rates and reheat loads are determined.

Fig. 3-6 shows the algorithm for determining airflow rates supplied to the individual zones without taking into account the reheat loads. Instantaneous sensible load (Q_s), zone temperature (T_z), discharge air temperature (T_{da}), surrounding room temperature (T_{surr}), temperature rise across supply fan ($\triangle T_f$), and area of supply duct walls (A_{duct}) are supplied as inputs to this module. Temperature of air entering VAV box (T_{br}) is calculated by first ignoring temperature rise in ducts ($\triangle T_d$). With the knowledge of duct shape – rectangular or round, a factor (R_l) to describe predicted leakage rates for ducts is selected. Afterwards, supply air mass flow rate (M_{a1}) is determined, on the basis of which temperature rise in ducts ($\triangle T_d$) is calculated. After the actual temperature of air entering VAV box (T_{br}) is computed, supply air mass flow rate (M_{a2}) is determined and compared

with the previous one (M_{a1}). If both values are sufficiently close, supply air mass flow rate (M_{a1}) is considered to be the correct value. Otherwise, iteration is performed as shown in **Fig. 3-6**.

Fig. 3-7 shows the algorithm for determining airflow rates supplied to the individual zones taking into account the reheat loads. In this module, the prescribed minimum airflow rate (M_{mina}), temperature rise in ducts ($\triangle T_d$), supply air mass flow rate (M_a), and leakage factor (R_l), together with those used in **Fig. 3-6**, are used as inputs. Then, supply airflow rate (M_a) is compared with minimum airflow rate (M_{mina}) in order to determine if reheat is required. A negative value means that reheat is required. In this case, the supply air mass flow rate (M_a) is set equal to minimum airflow rate (M_{mina}). Based on this, temperature rise in ducts ($\triangle T_d$) is recalculated. Supply air temperature (T_{sa}) after reheating is determined. Temperature of air entering VAV box (T_{br}) is also calculated by including the temperature rise in ducts ($\triangle T_d$). Finally, the algorithm computes the reheat loads (H_{re}). A positive value means that no reheat is required. In this case, the input value of supply air mass flow rate (M_a) is correct and supply air temperature (T_{sa}) is the same as temperature of air entering VAV box (T_{br}).

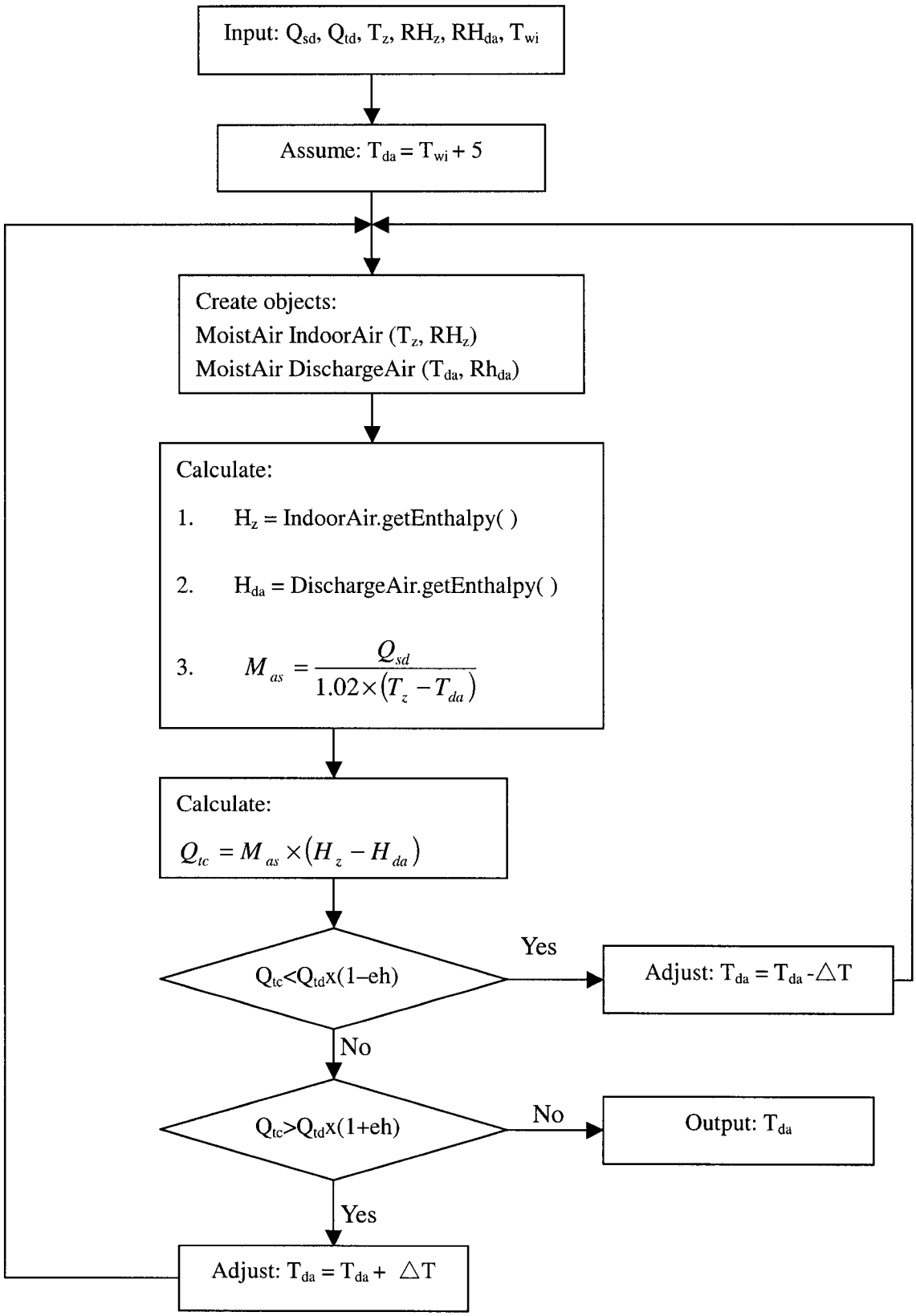


Figure 3-5 Algorithm for determining discharge air temperature

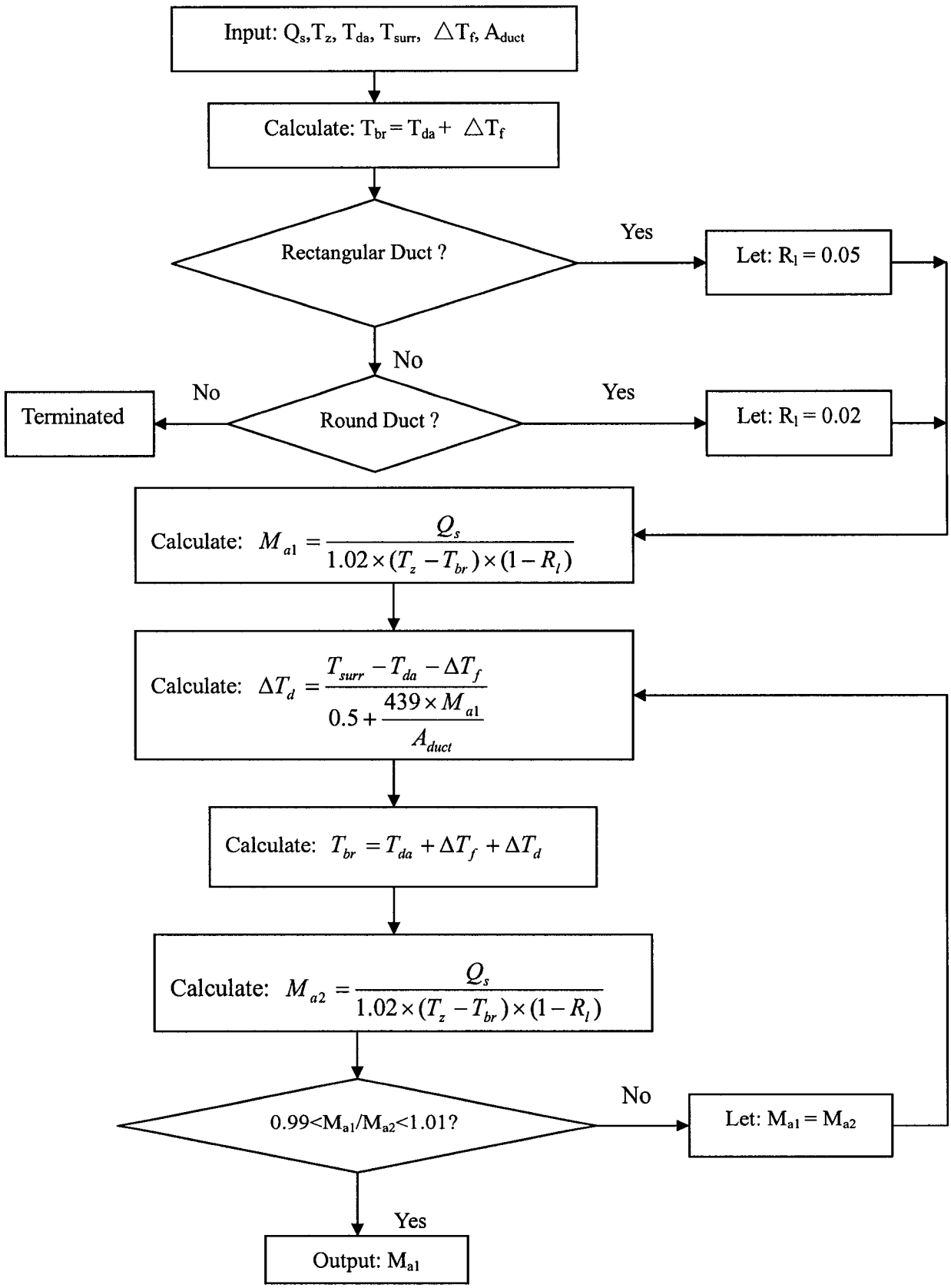


Figure 3-6 Algorithm for determining supply air mass flow rate

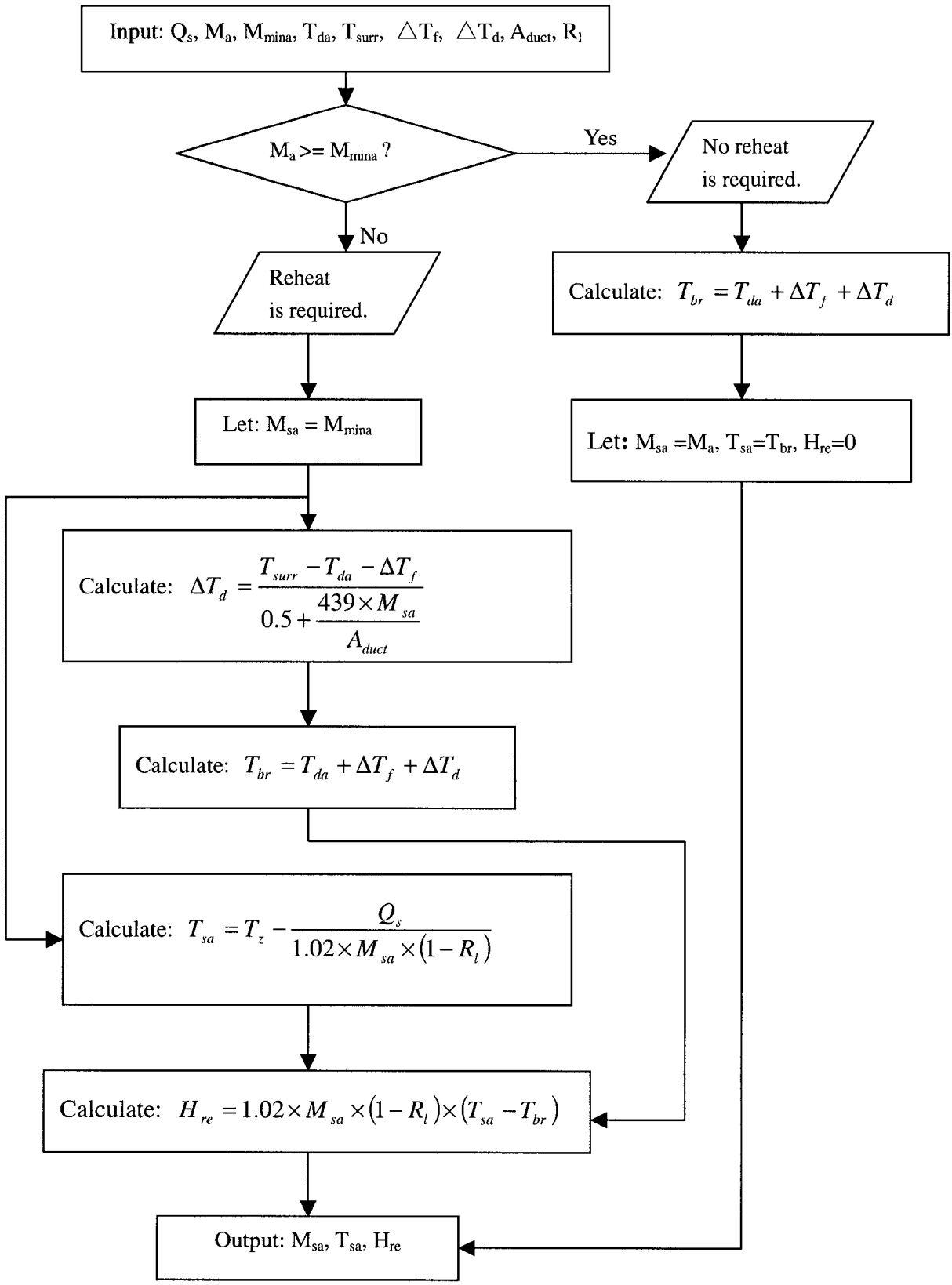


Figure 3-7 Algorithm for determining supply air mass flow rate, supply air temperature and reheat load

3.2.3 Practical Considerations

3.2.3.1 Temperature Rise across Fans

In HVAC applications, the temperature rise across fans may be found by the following equation (ASHRAE Handbook 2004):

$$\Delta T_f = \frac{0.829 \times \Delta P}{\eta_f} \qquad (3\text{-}1)$$

where

ΔT_f = temperature rise across fan (K)

ΔP = pressure rise across fan (kPa)

η_f = fan total efficiency (dimensionless)

3.2.3.2 Temperature Rise in Ducts

Temperature rise in insulated ducts may be determined by the following equation (Porges 2001):

$$\Delta T_d = \frac{t_r - t_i}{0.5 + \frac{439 \times \dot{m}_a}{A_d}} \qquad (3\text{-}2)$$

where

ΔT_d = temperature rise in ducts (°C)

t_r = surrounding room temperature (°C)

t_i = temperature of air entering ducts (°C)

$\dot{m}_a$ = air mass flow rate in ducts (kg/s)

A_d = area of duct walls (m^2)

3.2.3.3 Leakage Rates of Ducts

Following the guidelines given in (ASHRAE Handbook 1997), air leakage rates of 5% for rectangular metal ducts and 2% for round metal ducts were assumed.

3.3 Simulation Results

The simulated VAV system consisted of two zones measuring about 465 m^2 and 280 m^2 respectively in floor area. Design indoor conditions were set at 25°C, 50% for Zone 1 and 23°C, 55% for Zone 2. The minimum airflow rate allowed for either zone was assumed to be 40% of the associated maximum supply airflow rate. Ambient conditions and hourly loads of the design day were depicted in **Fig. 3-8** and **3-9** respectively. Note that Zone 1 had larger sensible and total loads. Accordingly, discharge air temperature was determined such that design conditions of Zone 1 were satisfied. The discharge air state point was located as shown in **Fig. 3-10**. By doing this, discharge air temperature was found to be 12.7°C. Once discharge air temperature was determined, it remained constant during all operating hours by modulating chilled water flow rate. Air quantities supplied to zones were determined by zone sensible loads and the temperature differences between zone air and supply air. When the calculated supply airflow rate was not less than the minimum airflow rate, no reheat would be required. Otherwise, reheat coil would be activated. The simulation results depicted in **Fig. 3-11** presented that for Zone 1, supply airflow rate was kept unchanged 8:00 thru 10:00. This was because sensible loads were much less than the associated peak loads. As a result, reheat coil was activated and quantities of air delivered

to Zone 1 remained the same, which was equal to the prescribed minimum airflow rates. Temperature of air supplied to Zone 1 was much higher than discharge air temperature 8:00 thru 10:00, which was seen in **Fig. 3-12**. This was because the temperature difference between supply air and discharge air was the summation of temperature rises across supply fan, in ducts and caused by reheat. On the other hand, no reheat was required during the period of 11:00 thru 18:00. Therefore, temperature difference between supply air and discharge air was relatively small. Similar trends occurred for Zone 2 as well.

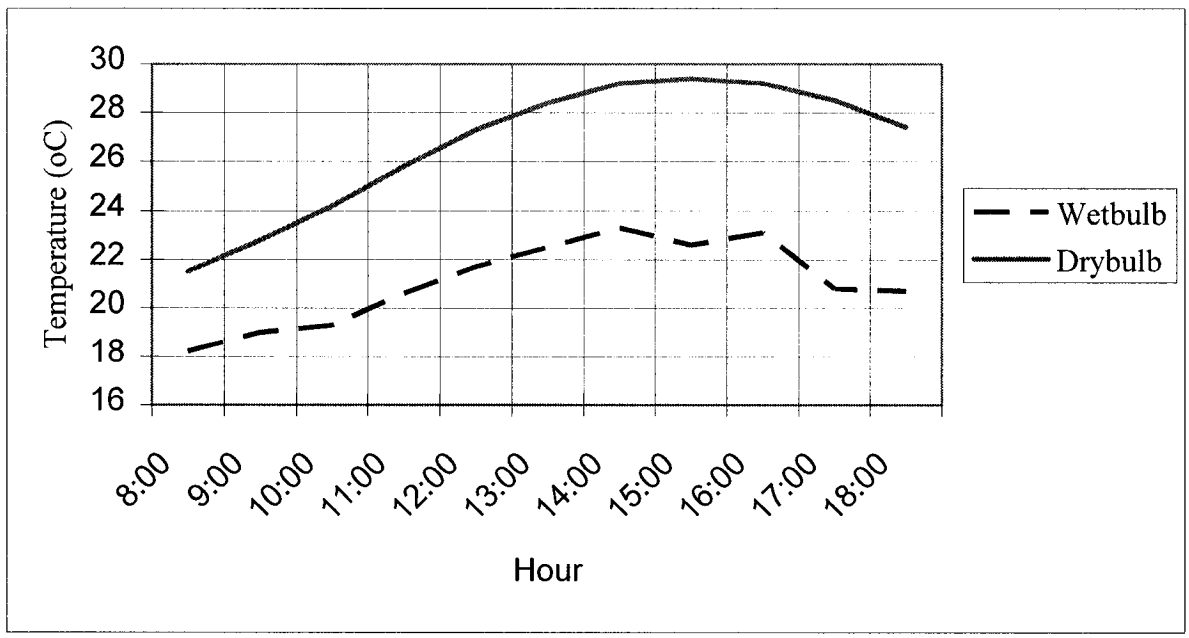


Figure 3-8 Hourly outdoor conditions of design day

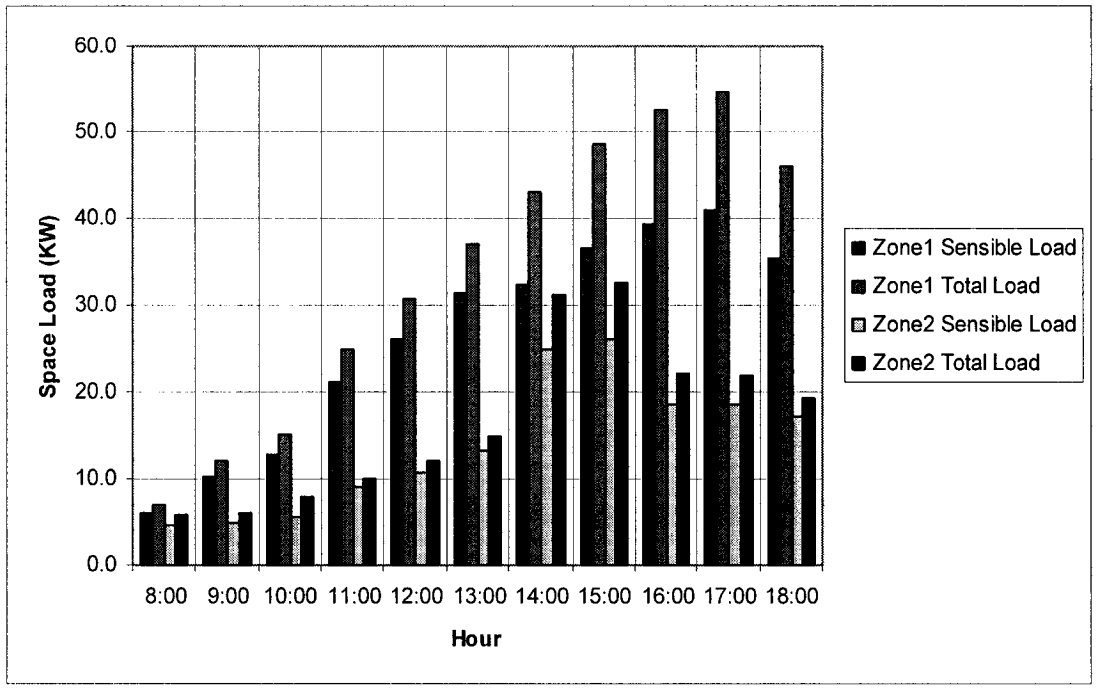


Figure 3-9 Hourly loads in design day

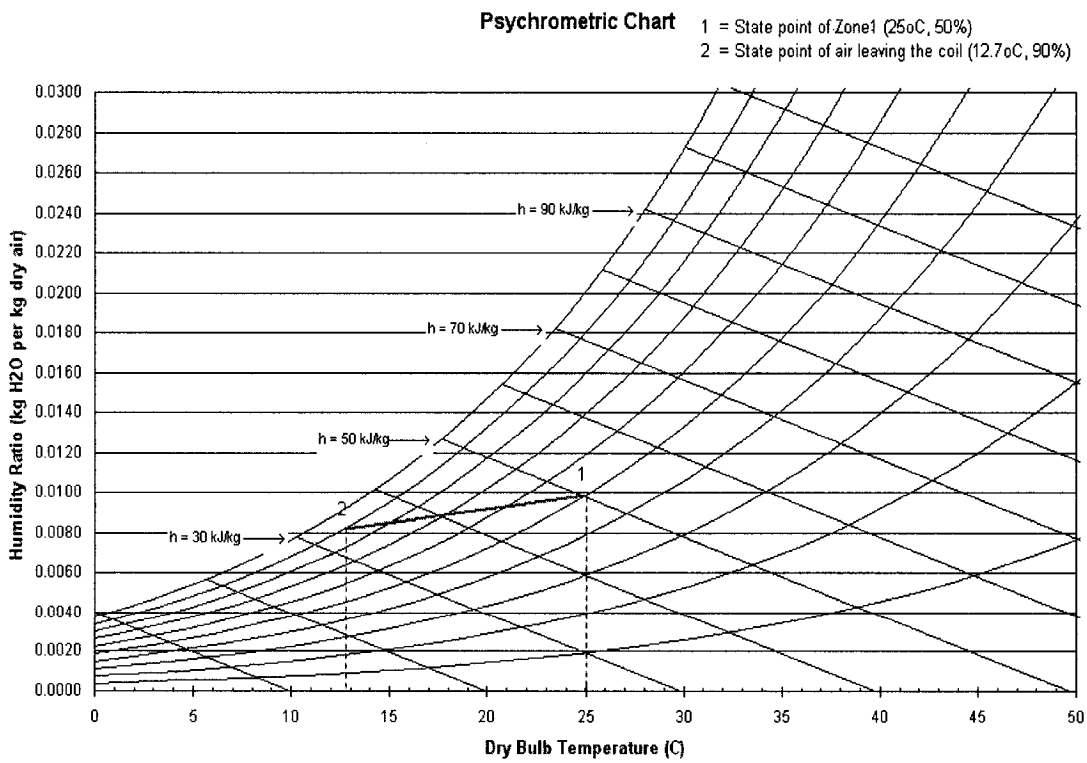


Figure 3-10 Locating discharge air state point

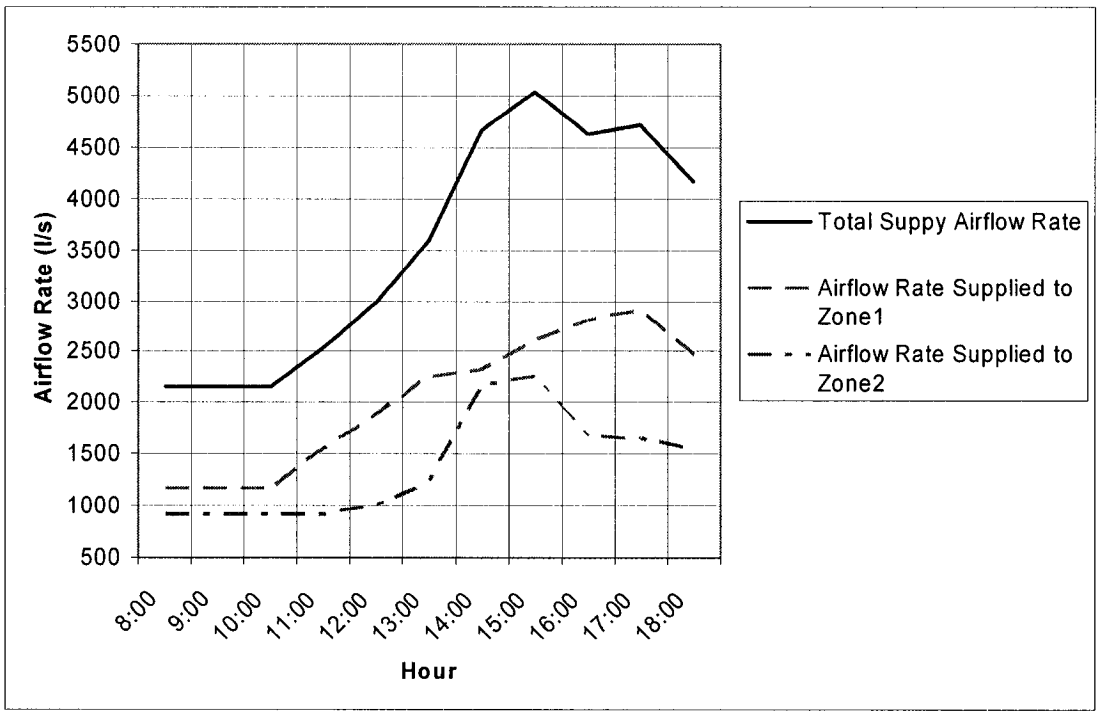


Figure 3-11 Supply airflow rate in design day

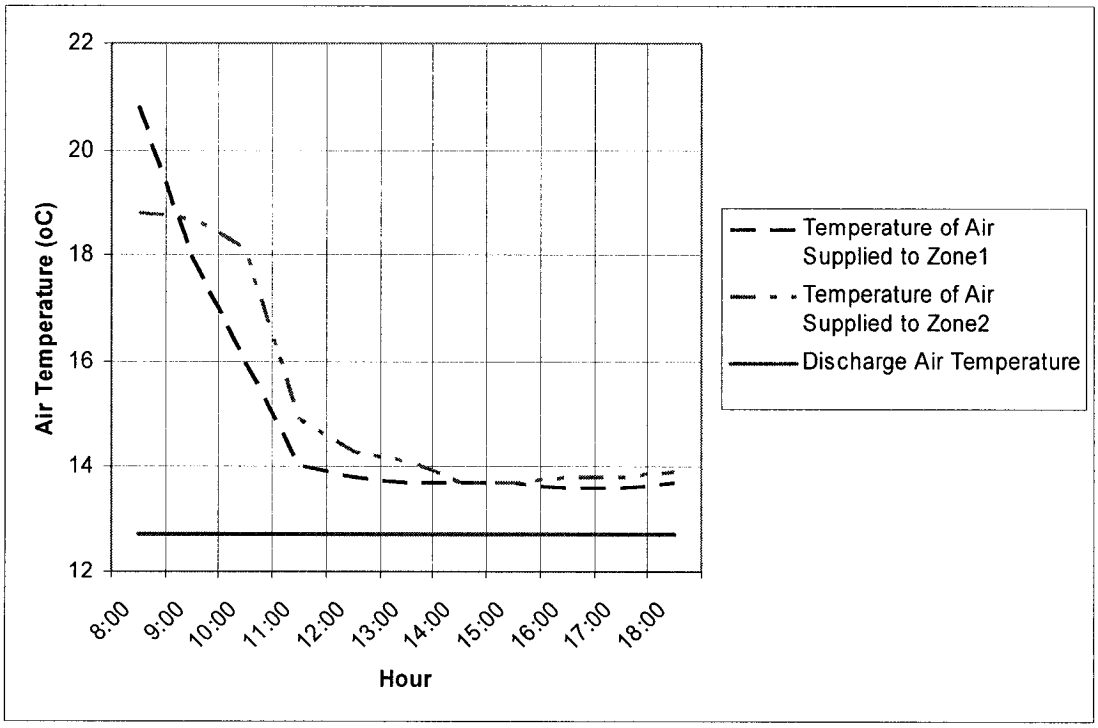


Figure 3-12 Supply (discharge) air temperature in design day

3.4 Conclusions

The simulation results reveal the fact that the developed software yields results that are consistent with the expected trends from a VAV HVAC system. Computations done on the example case show that VAV system can save air transportation energy relative to CVA system. The results present that on design day, 51% of fan power consumption could be saved. However, this kind of VAV system is not the best solution to energy conservation. Potential solutions for further energy savings can still be explored. In the simulated test case, it is noted that both zones need reheat 8:00 thru 10:00. In this case, energy will be wasted as a result of simultaneous heating and cooling operations. The reasonable solutions is to, rather than keep discharge air temperature constant, raise it such that supply air temperature satisfies at least one zone's condition without reheat. That is, at any operating time, no more than one zone requires reheat. Accordingly, energy cancellation may be partially eliminated. More important, raising discharge air temperature will lead to more energy savings, which will be discussed in Chapter 4 in detail.

CHAPTER 4

DYNAMIC MODELLING AND CONTROL STRATEGIES OF THE VAV HVAC SYSTEM

In order to perform real-time simulations of the HVAC system, an appropriate dynamic model of the HVAC system has to be developed. In this chapter, a two-zone variable-air-volume terminal reheat system (VAV-TRH) is chosen and a dynamic model is developed by adapting component models from the literature and as well developing some models using energy balance principles.

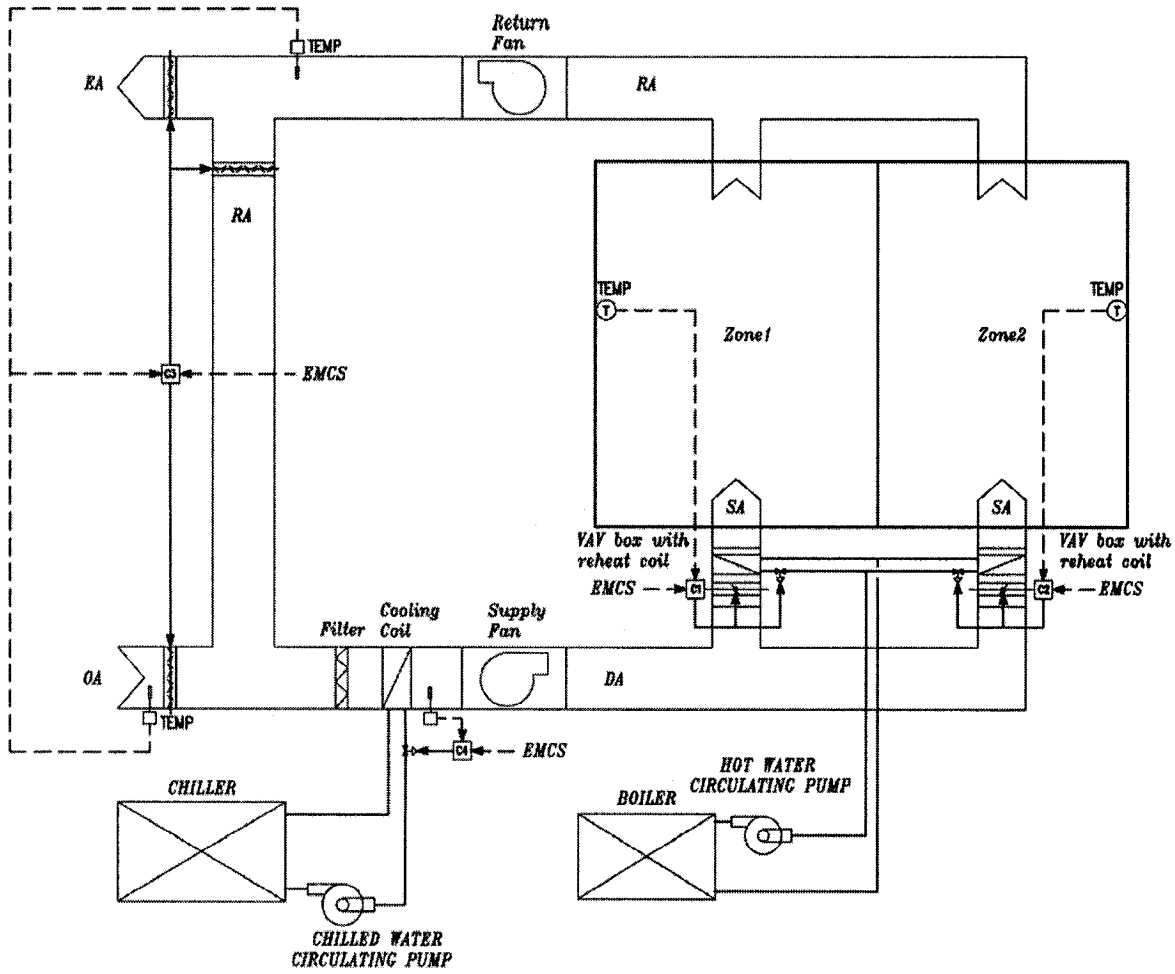
This chapter is organized in three sections. Section 1 describes the physical model of the chosen system. Section 2 designs two control strategies for the modeled system. Section 3 summarizes conclusions.

4.1 Physical Model of VAV-TRH System

The two-zone VAV-TRH HVAC system analyzed in this study is shown in **Fig. 4-1**. The major components of the system include (1) *two environmental zones*, (2) *a supply fan*, (3)

a return fan, (4) *a cooling and dehumidifying coil*, (5) *two VAV boxes with reheat coils*, (6) *a chiller*, (7) *a chilled water circulating pump*, (8) *a boiler*, (9) *a hot water circulating pump*, and (10) *ductwork.*

In the modeled system, outdoor air (**OA**) enters the system and is mixed with recirculated air (**RA**). Then mixed air passes through the cooling coil. After being conditioned in the cooling coil and the reheat coil (if required), the air is supplied to the thermal space – Zones 1 and 2. In response to demand for cooling from zone thermostats, volume flow rates of supply air (**SA**) to Zones 1 and 2 vary by modulating the VAV dampers, which are maneuvered by the two controllers – **C1** and **C2**. Air leaving zones, which is referred to as return air (**RA**), is drawn by the return fan, after which one portion is mixed with the outdoor air (**OA**) and the remainder may be exhausted out of the system. The controller **C3**, which is referred to as *Economizer*, modulates positions of outdoor air, recirculated air, and exhaust air dampers in order to minimize the mechanical cooling energy requirements by taking full advantage of the natural cooling effect of outdoor air. The controller **C4** is used to maintain discharge air temperature at a varying setpoint value by adjusting the chilled water control valve. In addition to the above four controllers, there is a supervisory control, which is referred to as ***E***nergy ***M***anagement and ***C***ontrol ***S***ystem (**EMCS**). This system, in which control strategies of HVAC system are stored, determines the setpoint values of zone temperatures and discharge air temperature and then supplies these values to the controllers as inputs.



OA	Outside Air	**RA**	Return/Recirculated Air
DA	Discharge Air	**EA**	Exhaust Air
EMCS	Energy Management and Control System	**SA**	Supply Air

Figure 4-1 Schematic diagram of a two-zone VAV-TRH HVAC system

4.1.1 Zone Model

An environmental zone is a complex thermal system, thus, a detailed analytical approach to formulating a model is not suitable for on-line applications. However, it is reasonable to simplify the thermal system to a space enclosed by a building envelope that is exposed to

certain outdoor conditions (Zhang and Nelson 1992). Assuming that outdoor air infiltration load is neglected, this simplified thermal system may be described by the following equation:

$$k\rho_a c_{pa} V_z \frac{dT_z(t)}{dt} = \dot{m}_{sa} c_{pa} [T_{sa}(t) - T_z(t)] + q_s \qquad (4\text{-}1)$$

where

k = factor describing the thermal capacity of zone ($k \geq 1$)

ρ_a = air density (lb_m/ft^3)

c_{pa} = specific heat of air (Btu/lb_m-F)

V_z = zone air volume (ft^3)

$\dot{m}_{sa}$ = supply air mass flow rate (lb_m/hr)

T_{sa} = supply air temperature (F)

T_z = zone temperature (F)

q_s = sensible load (Btu/hr)

t = time (hour)

It is also assumed that perfect mixing in the thermal space exists so that the air temperatures within and leaving the thermal zone are equal.

4.1.2 Coil Model

The coil is the most important interface between the primary plant (e.g., chiller or boiler) and the secondary air distribution system. Quite a few models have been developed to

study its transient response characteristics so far. In this section, the coil model presented in (Clark 1985) is adapted and some modifications related to coil design (McQuiston et al. 2000) are also included. First, a steady state model is established to determine the chilled water outlet temperature and the discharge air temperature in the steady state. Then, air and water dynamic outlet temperatures are both calculated by introducing first-order differential equations. It is assumed that there exists perfect mixing in the coil so that the air temperatures within and leaving the coil are the same.

4.1.2.1 Dynamic Model

Air and water dynamic outlet temperatures are both calculated using the following equations:

$$\frac{dT_{ao}}{dt} = \frac{T_{aoss} - T_{ao}}{\tau} \tag{4-2}$$

$$\frac{dT_{wo}}{dt} = \frac{T_{woss} - T_{wo}}{\tau} \tag{4-3}$$

$$\tau = \frac{C_m}{U_o A_o} \tag{4-4}$$

where

T_{ao} = air dynamic outlet temperature (F)

T_{wo} = water dynamic outlet temperature (F)

T_{woss} = steady state water outlet temperature (F)

T_{aoss} = steady state air outlet temperature (F)

A_o = total air-side heat-transfer area (ft^2)

U_o = overall heat-transfer coefficient, based on the air-side area (Btu/hr-ft^2-F)

C_m = total thermal capacitance of coil material (Btu/hr-F)

τ = time constant (seconds)

4.1.2.2 Steady State Model

➢ *Steady state air and water outlet temperatures*

The steady state air and water outlet temperatures may be found using the definition of effectiveness ε (McQuiston et al. 2000):

$$T_{aoss} = T_{ai} + \frac{\varepsilon C_{\min}}{C_a}\left(T_{wi} - T_{ai}\right) \qquad (4\text{-}5)$$

$$T_{woss} = T_{wi} - \frac{C_a}{C_w}\left(T_{aoss} - T_{ai}\right) \qquad (4\text{-}6)$$

where

T_{ai} = entering coil air temperature (F)

T_{wi} = inlet water temperature (F)

ε = heat exchanger effectiveness (dimensionless)

$C_{\min}$ = minimum of C_a and C_w (Btu/hr-F)

C_a = air capacitance rate (Btu/hr-F)

C_w = water capacitance rate (Btu/hr-F)

➢ *Heater exchanger effectiveness*

The cooling (or reheat) coil used an approximate equation for effectiveness as a function of NTU for a cross-flow heat exchanger with both fluids unmixed to determine steady state air and water outlet temperatures (McQuiston et al. 2000):

$$\varepsilon = 1 - \exp\left\{\frac{\exp[-Rn(NTU)]-1}{Rn}\right\} \qquad (4\text{-}7)$$

$$n = (NTU)^{-0.22} \qquad (4\text{-}8)$$

$$R = \frac{C_{\min}}{C_{\max}} \qquad (4\text{-}9)$$

where

R = ratio of minimum to maximum capacitance (dimensionless)

NTU = number of units (dimensionless)

$C_{\max}$ = maximum of C_a and C_w (Btu/hr-F)

➢ *Number of units(NTU)*

Number of transfer units (NTU) may be determined by the following equations (McQuiston et al. 2000):

$$NTU = \frac{U_o A_o}{C_{\min}} \tag{4-10}$$

$$C_a = \dot{m}_a \times c_{pa} \tag{4-11}$$

where

$\dot{m}_a$ = air mass flow rate (lb_m/hr)

c_{pa} = air specific heat (Btu/lb_m-F)

$$C_w = \dot{m}_w \times c_w \tag{4-12}$$

where

$\dot{m}_w$ = chilled water mass flow rate (lb_m/hr)

c_w = water specific heat (Btu/lb_m-F)

- *Overall heat-transfer coefficient*

From the standpoint of practical engineering, two factors must be taken into account when determining the overall heat-transfer coefficient. First, the fins are not rigidly attached to tubes so that the thermal contact resistance exists (McQuiston et al. 2000). Second, both inside and outside surfaces of tubes do not remain clean after some time of operation. Deposits are really a gradual build-up of layers of dirt due to impurities in the fluid, chemical reactions between the fluid and the metal, rust, etc. The deposits significantly affect the heat transfer efficiency. The effect of the deposits may be represented quantitatively by the fouling factor (Karkekar and Desmond 1982). Accordingly, overall heat-transfer coefficient may be determined by the following

equations:

$$\frac{1}{U_O} = \frac{1}{h_O \eta_s} + \frac{1}{h_i (\frac{A_i}{A_O})} + R_{ct} + R_f \qquad (4\text{-}13)$$

$$R_{ct} = 2.222 \times 10^{-6} \left[\frac{D_t (\frac{s}{y} - 1)^2}{y} \right]^{0.6422} \qquad (4\text{-}14)$$

where

h_o = air-side heat-transfer coefficient (Btu/hr-ft^2-F)

η_s = surface effectiveness (dimensionless)

h_i = water-side heat-transfer coefficient (Btu/hr-ft^2-F)

A_i = total water-side heat-transfer area (ft^2)

R_{ct} = unit contact resistance (hr-ft^2-F/Btu)

R_f = fouling factor (hr-ft^2-F/Btu)

D_t = outside tube diameter (in.)

s = fin spacing (in.)

y = fin thickness (in.)

- *Fin efficiency and surface effectiveness*

Fin efficiency and surface effectiveness may be determined by the following equations (McQuiston et al. 2000):

$$\eta_s = 1 - \frac{A_f}{A}(1-\eta) \qquad (4\text{-}15)$$

where

A = total surface area of the fin and base (ft^2)

A_f = fin surface area (ft^2)

η = fin efficiency (dimensionless)

$$\eta = \frac{\tanh(mr\phi)}{mr\phi} \qquad (4\text{-}16)$$

where

r = tube outside radius (in.)

$$m = \left(\frac{2h_o}{\dot{k}y}\right)^{1/2} \qquad (4\text{-}17)$$

where

$\dot{k}$ = thermal conductivity of the fin material (Btu-ft)/ ($hr\text{-}ft^2\text{-}F$)

$$\phi = \left(\frac{R_e}{r} - 1\right)\left[1 + 0.35\ln\left(\frac{R_e}{r}\right)\right] \qquad (4\text{-}18)$$

$$\frac{R_e}{r} = 1.27\psi(\beta - 0.3)^{1/2} \qquad (4\text{-}19)$$

where

$$\psi = \frac{M}{r}$$

$$\beta = \frac{L}{M}$$

➢ *Air-side heat-transfer coefficient*

The air-side heat-transfer coefficient may be determined by the following equations (McQuiston et al. 2000):

$$h_o = jG_c c_{pa} \Pr^{-2/3} \qquad (4\text{-}20)$$

where

j = j-factor (dimensionless)

G_c = air mass velocity referred to the minimum flow area inside the coil (lbm/hr-ft^2)

Pr = Prandtl number (1/h)

$$\Pr = \frac{3600\mu c_{pa}}{\bar{k}} \qquad (4\text{-}21)$$

where

μ = air dynamic viscosity (lb/ft-sec)

$\bar{k}$ = air thermal conductivity (Btu/ft-F)

$$j = \frac{1 - 1280 N_r \, \mathrm{Re}_{Xb}^{-1.2}}{1 - 5120 \, \mathrm{Re}_{Xb}^{-1.2}} j_4 \qquad (4\text{-}22)$$

where

N_r = number of rows

Re_{Xb} = Reynolds number based on X_b (dimensionless)

j_4 = Colburn heat transfer j-factor for 4-row cooling coil

$$\mathrm{Re}_{Xb} = \frac{G_c X_b}{\mu} \qquad (4\text{-}23)$$

where

X_b = longitudinal tube spacing (in.)

$$j_4 = 0.0013 + 0.267 JP \times J(s) \qquad (4\text{-}24)$$

$$JP = \mathrm{Re}_{Do}^{-0.4} \left(\frac{A}{A_t}\right)^{-0.15} \qquad (4\text{-}25)$$

where

Re_{Do} = Reynolds number based on the tube outside diameter (dimensionless)

A = total heat-transfer area (ft^2)

A_t = area of bare tubes without fins (ft^2)

$$\mathrm{Re}_{Do} = \frac{G_c D_o}{\mu} \qquad (4\text{-}26)$$

where

D_o = tube outside diameter (in.)

$$J(s) = 0.84 + 4 \times 10^{-5} \mathrm{Re}_s^{1.25} \qquad (4\text{-}27)$$

where

Re_s = Reynolds number based on fin spacing (dimensionless)

$$\mathrm{Re}_s = \frac{\mathrm{Re}_{Do}\, s}{D_o} \qquad (4\text{-}28)$$

➢ *Water-side heat-transfer coefficient*

The water-side heat-transfer coefficient may be calculated by the following equations (McQuiston et al. 2000):

$$h_i = 0.023 \mathrm{Re}_{Di}^{0.8} \mathrm{Pr}^n \frac{\hat{k}}{D_i} \qquad (4\text{-}29)$$

where

h_i = water-side heat-transfer coefficient (Btu/hr-ft^2-F)

Re_{Di} = Reynolds number based on the tube inside diameter (dimensionless)

$\hat{k}$ = water thermal conductivity (Btu/ft-F)

D_i = tube inside diameter (ft)

n = 0.4 for the heating coil (dimensionless)

= 0.3 for the cooling coil (dimensionless)

$$\mathrm{Re}_{Di} = \frac{\rho V_w D_i}{12\mu} \tag{4-30}$$

where

ρ = water density (lb/ft^3)

V_w = water velocity (fpm)

μ = water dynamic viscosity (lb/ft-sec)

$$\mathrm{Pr} = \frac{3600\mu c_p}{k} \tag{4-31}$$

where

c_p = specific heat of water at constant pressure (Btu/lb-F)

4.1.3 Fan Model

The relationship between fan shaft horsepower and delivered airflow volume may be described using the following equation (Huang 2003):

$$W_{sh} = \frac{Q\Delta P}{6350\eta_t} \tag{4-32}$$

where

W_{sh} = fan shaft horsepower (Hp)

ΔP = total pressure difference (in.WG)

Q = airflow rate delivered by fan (cfm)

η_t = fan total efficiency (dimensionless)

4.1.4 Economizer Model

The economizer control cycle is currently one of the most popular energy conservation measures for air-handling units (AHU) in commercial buildings. It reduces or eliminates the need for mechanical cooling by utilizing outdoor air when outdoor conditions are favorable. Parken et al. (1982) found that cooling energy savings varied from 1% to 77%, depending on climatic conditions.

The economizer control cycle may be classified into two types: *the temperature economizer* and *the enthalpy economizer* (Levenhagen and Spethmann 1993). In this study, the temperature economizer is applied in the modeled VAV-TRH system since only temperature control is of interest.

In the temperature economizer cycle, the percentage opening of outdoor air damper should be equal to that of exhaust air damper. The input data to the temperature economizer cycle include (1) *the outdoor air temperature,* (2) *the discharge air temperature setpoint,* and (3)

the return air temperature. The economizer outputs positions for outdoor air damper, exhaust air damper, and recirculated air damper so that outdoor air and recirculated air are mixed in appropriate quantities.

4.1.4.1 Economizer Control Strategies

In the case of cooling, the air handling unit (AHU) has three operation modes during occupied periods, which include (1) *mechanical cooling with minimum outdoor air intake* (***Mode1***), (2) *mechanical or natural cooling with 100% outdoor air* (***Mode2***), and (3) *natural cooling with outdoor air* (***Mode3***). Correspondingly, the economizer has three control modes. It should be noted that the outdoor air damper must always be open sufficiently to provide the minimum outdoor air for good indoor air quality no matter which mode is taken.

- *Mode#1*

  If the outdoor air temperature is greater than or equal to the return air temperature, the economizer cycle uses minimum outdoor air intake to maintain the mechanical cooling requirement at the lowest level.

- *Mode#2*

  If the outdoor air temperature is less than return air temperature but not less than the discharge air temperature setpoint minus the temperature rise across the supply fan,

the positions of the economizer dampers are at 0% return air, 100% exhaust air, and 100% outdoor air. In most cases, mechanical cooling is still required. However, when the outdoor air temperature happens to be equal to the discharge air temperature setpoint minus the temperature rise across the supply fan, there will be no need for mechanical cooling. In this case, natural cooling is used.

- *Mode#3*

  If the outdoor air temperature is less than the discharge air temperature setpoint minus the temperature rise across the supply fan, the economizer dampers are modulated to maintain mixed air temperature equal to the discharge air temperature setpoint. In this case, there is no need for the mechanical cooling.

4.1.4.2 Economizer Mathematical Model

- ***Mode #1**: Mechanical Cooling with Minimum Outdoor Air*

If $T_{oa} \geq T_{ra}$

$$\dot{m}_{oa} = MinR_{oa} \times \dot{m}_{sa}$$

Then $\dot{m}_{ea} = \dot{m}_{oa}$

$$\dot{m}_{ra} = \dot{m}_{sa} - \dot{m}_{oa}$$

➢ ***Mode #2**: Mechanical or Natural Cooling with 100% Outdoor Air*

If $T_{oa} \geq T_{daset} - \Delta T_{sf}$ and $T_{oa} < T_{ra}$

$$\dot{m}_{oa} = \dot{m}_{sa}$$

Then $\dot{m}_{ea} = \dot{m}_{sa}$

$$\dot{m}_{ra} = 0$$

➢ ***Mode #3**: Natural Cooling with Outdoor Air*

If $T_{oa} < T_{daset} - \Delta T_{sf}$

$$\dot{m}_{oa} = \frac{T_{ra} - T_{daset}}{T_{ra} - T_{oa}} \times \dot{m}_{sa}$$

Then $\dot{m}_{ea} = \dot{m}_{oa}$

$$\dot{m}_{ra} = \dot{m}_{sa} - \dot{m}_{oa}$$

where

T_{oa} = outdoor air temperature (F)

T_{ra} = return air temperature (F)

T_{daset} = discharge air temperature setpoint (F)

$\dot{m}_{oa}$ = outdoor airflow rate (lb_m/hr)

$\dot{m}_{ea}$ = exhaust airflow rate (lb_m/hr)

$\dot{m}_{sa}$ = supply airflow rate (lb_m/hr)

$\dot{m}_{ra}$ = recirculated airflow rate (lb_m/hr)

$MinR_{oa}$ = minimum ratio of outdoor air intake to supply airflow rate

4.1.4.3 Economizer Outputs

- *Outdoor air damper control signal*

$$U_{oa} = \frac{\dot{m}_{oa}}{\dot{m}_{sad}} \tag{4-33}$$

where:

U_{oa} = outdoor air damper position [0, 1]

$\dot{m}_{sad}$ = supply airflow rate required for design conditions (lb_m/hr)

- *Exhaust air damper control signal*

$$U_{ea} = \frac{\dot{m}_{ea}}{\dot{m}_{sad}} \tag{4-34}$$

where:

U_{ea} = exhaust air damper position [0, 1]

- *Recirculated air damper control signal*

$$U_{ra} = \frac{\dot{m}_{ra}}{\dot{m}_{rad}} \tag{4-35}$$

where:

U_{ra} = recirculated air damper position [0, 1]

$\dot{m}_{rad}$ = return airflow rate required for design conditions (lb_m/hr)

= $\dot{m}_{sad} - \dot{m}_{oad}$

$\dot{m}_{oad}$ = outdoor airflow rate required for design conditions (lb_m/hr)

4.1.5 Types of Control

Two types of control were used in the model: *two-position control*, and *modulating control*. Today, they are quite popular in HVAC applications.

4.1.5.1 Two-position Control

Two-position control, also referred to as on-off control, is the simplest and most common type. The control device can be positioned only to a maximum or minimum state, or can be either on or off. In the model, it is used to control the hot water valve of the reheat coil. A thermostat is designed with an error band, which centers on the zone temperature setpoint. When the zone temperature falls below the lower bound of the error band, the controller opens the valve. The hot water valve is closed by the controller when the zone temperature rises above the upper bound of the error band. That is to say, the on-off control is implemented using the following methodology:

$$\text{If } T_z > T_{zset} + \frac{\Delta T_b}{2} \qquad \text{then } U_{trh} = 0$$

$$\text{If } T_z < T_{zset} - \frac{\Delta T_b}{2} \qquad \text{then } U_{trh} = 1$$

where

T_z = zone temperature (F)

T_{zset} = zone temperature setpoint (F)

ΔT_b = error band (F)

U_{trh} = reheat coil valve position

Fig. 4-2 illustrates how the zone temperature changes with time with two-position control. It is assumed that T_{zset} is $75F$ and ΔT_b is $1.0F$.

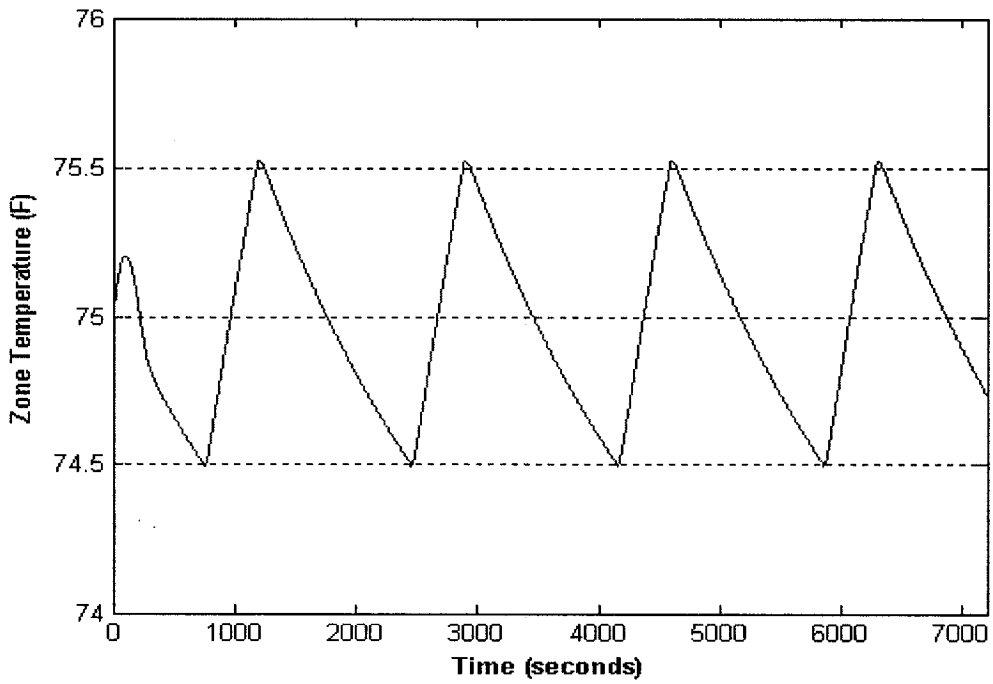


Figure 4-2 Two-position (on-off) control action

4.1.5.2 Modulating Control

Modulating control is comprised of (1) *proportional control* (**P**), (2) *proportional plus integral control* (**PI**), and (3) *proportional plus integral plus derivative* (**PID**). Proportional control is the simplest modulating control. The existence of an offset, which is a steady-state difference between the setpoint and the actual value of the control variable, is the major weakness of this control action. Offset depends on the load acting on the system and is an unavoidable consequence. PI control eliminates this offset using the

integral component. The integral term generates a control signal that is proportional to the time integral of the error signal. A continuous offset will therefore eventually produce a correction action, which will continue to accumulate until it has reached zero. Most electronic controllers and many pneumatic controllers use PI, and computers can be easily programmed for this mode. PID control brings a faster response and greater stability compared to PI control by adding the derivative term. However, it is more sensitive to noisy signals and harder to tune than PI control (ASHRAE Handbook 1997). In addition, most HVAC systems are relatively slow in response to changes in controller output, and PID control may overshoot. Therefore, PID control is usually not used in HVAC control (McQuiston et al. 2000). Accordingly, PI algorithm is used to control supply air dampers of VAV boxes and chilled water valve of the cooling coil in this thesis.

Fig. 4-3 depicts a closed-loop system of PI controller. A sensor provides feedback signal of a control variable. This variable is then compared with its setpoint and the difference (i.e., error) is multiplied by a constant (i.e., proportional gain) and summed with the product of the integral of the error and another constant (i.e., integral gain) to obtain an output. The output signal is sent to the actuator; thus initiating the regulation of the process.

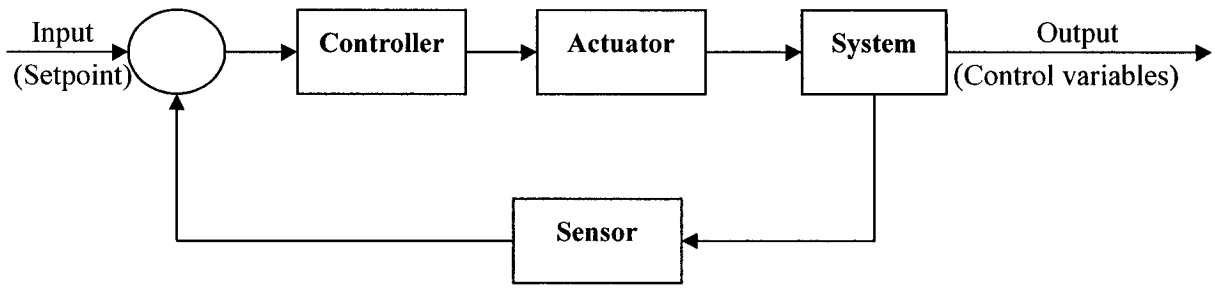


Figure 4-3 Closed-loop control mechanism

The two-zone VAV-TRH HVAC system studied in this thesis involves three closed control loops. The control variables are zone temperatures (i.e., T_{z1} and T_{z2}) and discharge air temperature (i.e., T_{da}). The variables to be modulated are air mass flow rates supplied to Zones 1 and 2 (i.e., $\dot{m}_{sa1}$ and $\dot{m}_{sa2}$) and chilled water supply mass flow rate (i.e., $\dot{m}_w$). Loops 1 and 2 are supply airflow rate control loops. Zone temperature error signals the associated controller to tailor the airflow rates by modulating VAV box damper openings. A transient zone temperature above the corresponding setpoint produces an increment of airflow rate by increasing the damper opening while a zone temperature below the setpoint reduces airflow rate by decreasing the damper opening. Loop 3 is chilled water flow rate control loop. The modulation of chilled water mass flow rate is achieved by adjusting the valve position. When the discharge air temperature is higher than the associated setpoint, the actuator accordingly opens the valve further until the temperature equals to its setpoint. On the other hand, when the discharge air temperature is less than the setpoint, the valve is closed a little bit.

PI control may be described by:

$$U = U_o + K_p e + K_i \int e dt \qquad (4\text{-}36)$$

where

U = controller output [0, 1]

= $\frac{\dot{m}_w}{\dot{m}_{wd}}$ for the cooling coil

= $\frac{\dot{m}_{sa}}{\dot{m}_{sad}}$ for the VAV box

U_o = controller output with no error, a constant

e = error

= $T_{da} - T_{daset}$ for the cooling coil

= $T_z - T_{zset}$ for the VAV box

K_p = proportional gain constant

K_i = integral gain constant

$\dot{m}_w$ = chilled water flow rate (gpm)

$\dot{m}_{wd}$ = chilled water flow rate for design conditions (gpm)

$\dot{m}_{sa}$ = supply airflow rate (cfm)

$\dot{m}_{sad}$ = supply airflow rate for design conditions (cfm)

4.2 Control Strategies of HVAC Systems

For the modeled VAV-TRH HVAC system, a *reheat control strategy* and an *optimal control strategy* were designed. The former is much related to the design issue and the latter has a great deal to do with the operation issue. Either of the strategy focuses on the determination of the discharge air temperature setpoint value due to the fact that the discharge air temperature is the most critical among the control variables, which has great impacts on the power requirements of the HVAC system.

4.2.1 Reheat Control Strategy

A reheat control strategy is one in which the discharge air temperature setpoint is determined when the supply airflow rate is minimum. Therefore, this strategy simply minimizes the air transportation power but might increase the reheat energy.

4.2.1.1 Reheat Control Methodology

An application of this strategy has been implemented in SIMULINK, which is referred to as *Reheat Analyzer*. In this strategy, zone temperature setpoints are kept constant, being equal to the design indoor temperatures (e.g., 78F for Zone 1 and 75F for Zone 2). It then outputs discharge air temperature setpoint and an integral for the given space loads and outdoor air conditions. This integral is called *reheat mode*, indicating whether or not reheat is required and which reheat coil has to be activated. The flowchart for the

algorithm corresponding to this strategy is shown in **Fig. 4-4**.

It should be noted that the fact that both zones need reheating means that the discharge air temperature is too low. In this case, the discharge air temperature setpoint must be raised so that no more than one zone requires reheat at any time. The following equation may be used for resetting the discharge air temperature setpoint.

$$T_{daset} = T_{zset} - \frac{Q_s}{1.1 \times (V_{th} \times CFM_{sad})} \qquad (4\text{-}37)$$

where

Q_s = zone sensible load (Btu/hr)

V_{th} = threshold percentage for supply airflow rate

= 30% in this study

CFM_{sad} = supply airflow rate for the design conditions (cfm)

4.2.1.2 Reheat Control Analysis

Table 4.1 lists an example application of reheat control strategy. In the reheat mode of 22, the first digit means that the discharge air temperature will be reset and the second indicates that the reheat coil of Zone 2 will be activated. Apparently, reheat control strategy does not result in the maximum energy savings. First, there exits cancellation between cooling energy and heating energy most of the time when the reheat coil is activated. Second, it does not take the chiller power consumption into account; thus this

strategy cannot minimize the total energy consumed by the entire HVAC system. Optimal control strategy mitigates these two disadvantages; therefore, it will potentially save much more energy than reheat control strategy.

4.2.2 Optimal Control Strategy

The level of the control variables impacts the power consumption of the HVAC system. Different control variable settings may produce the same thermal comfort and indoor air quality but at different energy requirements. An optimal control strategy is one in which the total cost of operation is minimized while maintaining comfort conditions in the building for a given building load and ambient conditions. It generates an optimal set of control variables. Savings of 10% to 20% in building energy use may be obtained through optimal control (Pape et al. 1991).

4.2.2.1 Optimization Methodology

The method used for optimization in this thesis is referred to as *constrained nonlinear optimization*, which finds a constrained minimum of a scalar cost function of several variables starting at an initial estimate. First, an objective function has to be built that includes all of the energy consumption elements as well as the relationships between it and the associated variables. Second, upper/lower bounds and initial values of the variables should be given. Finally, linear and/or nonlinear constraints with equality and inequality relationships must be defined. A computer program, which is referred to as *Optimizer* and

listed in *Appendix B*, is used to find the optimal solution. The objective function is the HVAC system power requirement, which consists of energy consumed by chillers and supply (return) fans. It is acknowledged that the power consumed by chillers occupies nearly half of the entire HVAC power requirements. The optimization solution gives the optimal set of control variables that minimizes power consumption at a given time. Therefore, the following objective function was chosen

$$E_t = E_{ch} + E_{sf} + E_{rf} \tag{4-38}$$

where

E_t = total power (Hp)

E_{ch} = chiller power (Hp)

E_{sf} = supply fan power (Hp)

E_{rf} = return fan power (Hp)

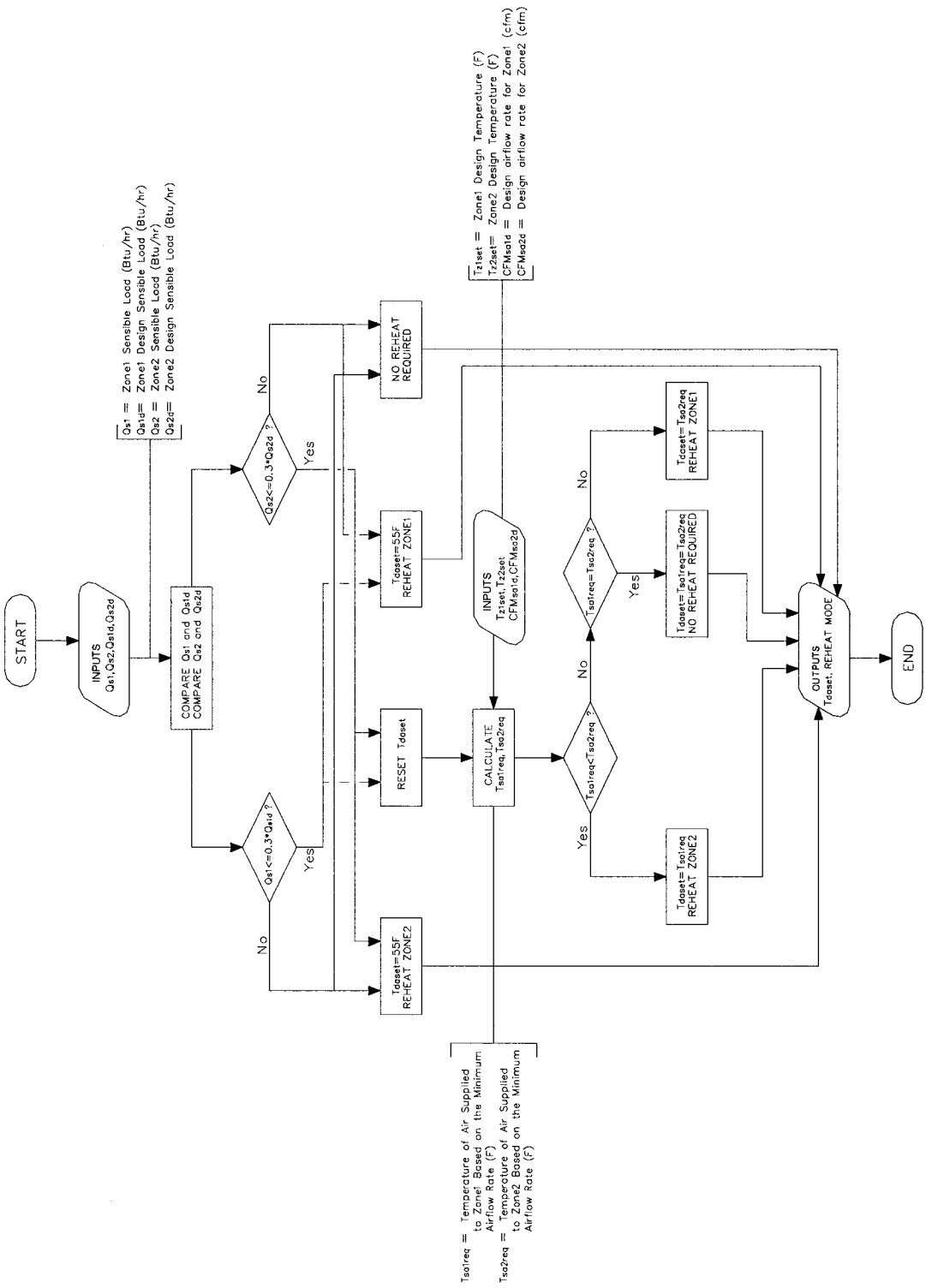


Figure 4-4 Algorithm of reheat control strategy (*Reheat Analyzer*)

Table 4.1 An example application of reheat control strategy

INPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Zone1	Sensible load	Q_{s1}	Btu/hr	30,560
	Peak sensible load	Q_{s1d}	Btu/hr	139,890
	Max. supply airflow rate	CFM_{sa1d}	cfm	5,530
	Temperature setpoint	T_{z1set}	F	78
Zone2	Sensible load	Q_{s2}	Btu/hr	15,980
	Peak sensible load	Q_{s2d}	Btu/hr	88,820
	Max. supply airflow rate	CFM_{sa2d}	cfm	4,040
	Temperature setpoint	T_{z2set}	F	75
OUTPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Discharge air temperature setpoint		T_{daset}	F	61.3
Reheat mode				22

The optimal solution and the associated initial values, lower bound and upper bound may be presented by vectors ***X***, ***X0***, ***lb*** and ***ub*** respectively. That is:

$$X = [T_{z1set}, T_{z2set}, T_{daset}, T_{wi}]$$

$$X0 = [78,75,55,44]$$

$$lb = [73,73,55,44]$$

$$ub = [78,78,64,53]$$

In addition, the objective function is subject to the following constraints:

$$lb \leq X \leq ub$$

$$V_{th} \times CFM_{sa1d} \leq CFM_{sa1} \leq CFM_{sa1d}$$

$$V_{th} \times CFM_{sa2d} \leq CFM_{sa2} \leq CFM_{sa2d}$$

The optimization problem was defined as the minimization of E_t subject to steady state constraints.

4.2.2.2 Optimal Control Analysis

An example application of optimal control strategy is shown in **Table 4.2**. It is found that a significant advantage of this strategy is that it eliminates the reheat requirement by raising the discharge air temperature while maintaining acceptable comfort conditions. Accordingly, the heating energy will definitely be saved. Furthermore, the chilled water supply temperature may be raised as the discharge air temperature goes up, which makes a great sense. It is stated that approximately two percent of input energy is saved for one degree Fahrenheit that the chilled water temperature is raised. Moreover, raising the chilled water temperature generally does not create any risks to equipment and costs little or nothing to accomplish (Wulfinghoff 1999). However, when the space load is far below

the associated peak value, the set of control variables given by *Optimizer* does not satisfy all the constraints. Accordingly, optimal control strategy is invalid in such cases and therefore the reheat control strategy is needed.

Table 4.2 An example application of optimal control strategy

INPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Sensible loads	Zone 1	Q_{s1}	Btu/hr	30,560
	Zone 2	Q_{s2}	Btu/hr	15,980
Outdoor air temperature		T_{oa}	F	70
Outdoor air intake		CFM_{oa}	cfm	1,120
OUTPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Zone temperature setpoints	Zone 1	T_{z1set}	F	78
	Zone 2	T_{z2set}	F	77
Discharge air temperature setpoint		T_{daset}	F	64
Chilled water supply temperature		T_{wi}	F	53
AHU operation mode				2

4.3 Conclusions

By comparing **Tables 4.1** and **4.2**, it is found that under the same operation conditions, the optimal control strategy results in 7% energy savings in comparison to the reheat control strategy by eliminating the need for reheat and raising the discharge air temperature setpoint. Therefore, the optimal control strategy is an ideal solution to energy conservation. Simulation experiments in this study have also shown that optimal control strategy may be applied most of the operation time.

CHAPTER 5

SOFTWARE DEVELOPMENT AND REAL-TIME SIMULATIONS OF THE VAV HVAC SYSTEM

A dynamic model of a two-zone variable-air-volume terminal reheat (VAV-TRH) HVAC system was developed in Chapter 4. Also, two control strategies – the optimal control strategy and the reheat control strategy – were designed. In this chapter, based on all these model equations, a computer program will be developed to perform both off-line and real-time simulations for the two-zone VAV-TRH HVAC system. This chapter is composed of three sections. The process of software development is explained in detail in Section 1. Section 2 presents several cases of off-line simulations and real-time experiments to study the dynamic behavior of the HVAC system. Section 3 summarizes conclusions.

5.1 Software Development

5.1.1 Development Environment

The software is done at the system-level and implemented in MATLAB/SIMULINK environment. MATLAB/SIMULINK is for modeling, simulation, and analyzing dynamic

systems and is widely used in academia and industry for modeling and simulating dynamic systems. It supports linear and nonlinear systems and can be modeled in continuous time, sampled time, or a hybrid of the two. Moreover, for modeling, MATLAB/SIMULINK provides a graphical user interface (*GUI*) for building models as block diagrams.

5.1.2 Software Requirements

The developed system, which is referred to as *HVAC Simulator* in this thesis, is an air-conditioning simulation system that can be used to help facility staff realize optimal operation and researchers conduct virtual HVAC experiments as well. In Chapter 6, expert rules will be tested and analyzed based on real-time virtual experiments using this system. Object oriented analysis (OOA) was used to specify requirements.

The analysis methodology used in this thesis is the so-called *use-case driven approach* created by Jacobson (1992). ***Use case*** is a structured and reasonably formal notation for capturing the high-level system functionality: how the system interacts with the actor to deliver its functionality (Ormandjieva, 2003). It is used to describe a set of sequences of actions (scenarios) a system performs that yields a result of value to a particular actor. This technique has become very widely used in recent years. It is the dominant requirements analysis and specification method in the software world of business systems. Cooling (2003) stated that use case techniques possess six functions: (1) *analyze client*

requirements, (2) *organize and present requirements in a way that is useful, meaningful and complete*, (3) *minimize confusion and misunderstanding between clients and suppliers*, (4) *validate system-level*, (5) *develop specifications for the software system itself*, and (6) *define the outlines of system acceptance tests (for function, performance and usage)*.

The following subsections give a general description of the *HVAC Simulator*, list the functionalities that exist in the system, depict the types of users of the system, describe general constraints, and indicate the assumptions and dependencies. In addition, the external interfaces to the system, performance requirements, design constraints, quality attributes and other requirements are also explained.

5.1.2.1 General Description

HVAC Simulator has two running modes: off-line simulation and real-time (i.e., on-line) application. When off-line mode is in effect, this system acts like a simulation tool, which is the process of running a dynamic system in nonreal time to observe its behavior (MATLAB Manuals 1997). Running off-line simulations lets the user observe the behavior of the model in nonreal-time. In addition, simulations are interactive, so the user can change parameters on the fly and immediately see what happens. Therefore, it can be used to predict the dynamic behavior of the modeled HVAC system. In this situation, this program, together with *Optimizer*, which produces the optimal set of the control variables, will be a valuable tool for HVAC operators. For example, once future space loads and

ambient conditions are known or predicted, the HVAC operator inputs the data to *Optimizer*, which, in return, either outputs optimal control setpoint values for chilled water supply temperature, discharge air temperature setpoint and zone temperature setpoints or tells that reheat is actually needed. If the optimal control strategy is viable, the operator enters the settings to *HVAC Simulator* to check how the HVAC system will be performing. If the performance is acceptable, the HVAC operator will then reset the control variables for the future hour. In this way, the optimal operation will be realized and maximum savings will be obtained.

When on-line mode is selected, real-time simulations are realized. The real-time simulation refers to code that is ready to run in real time with the kernel (MATLAB Manuals 1997). The user can create a real-time application to let the system run while synchronized to a real-time clock. This allows the system to control or interact with an external system. In this case, the program will become a virtual HVAC system. Experiments may be conducted exactly the same as experiments are performed in a physical HVAC system. The expert rules used to on-line detect and diagnose HVAC faults can be tested by this kind of experiment.

A use case diagram is generated in terms of the general descriptions, which is shown in **Fig. 5-1**.

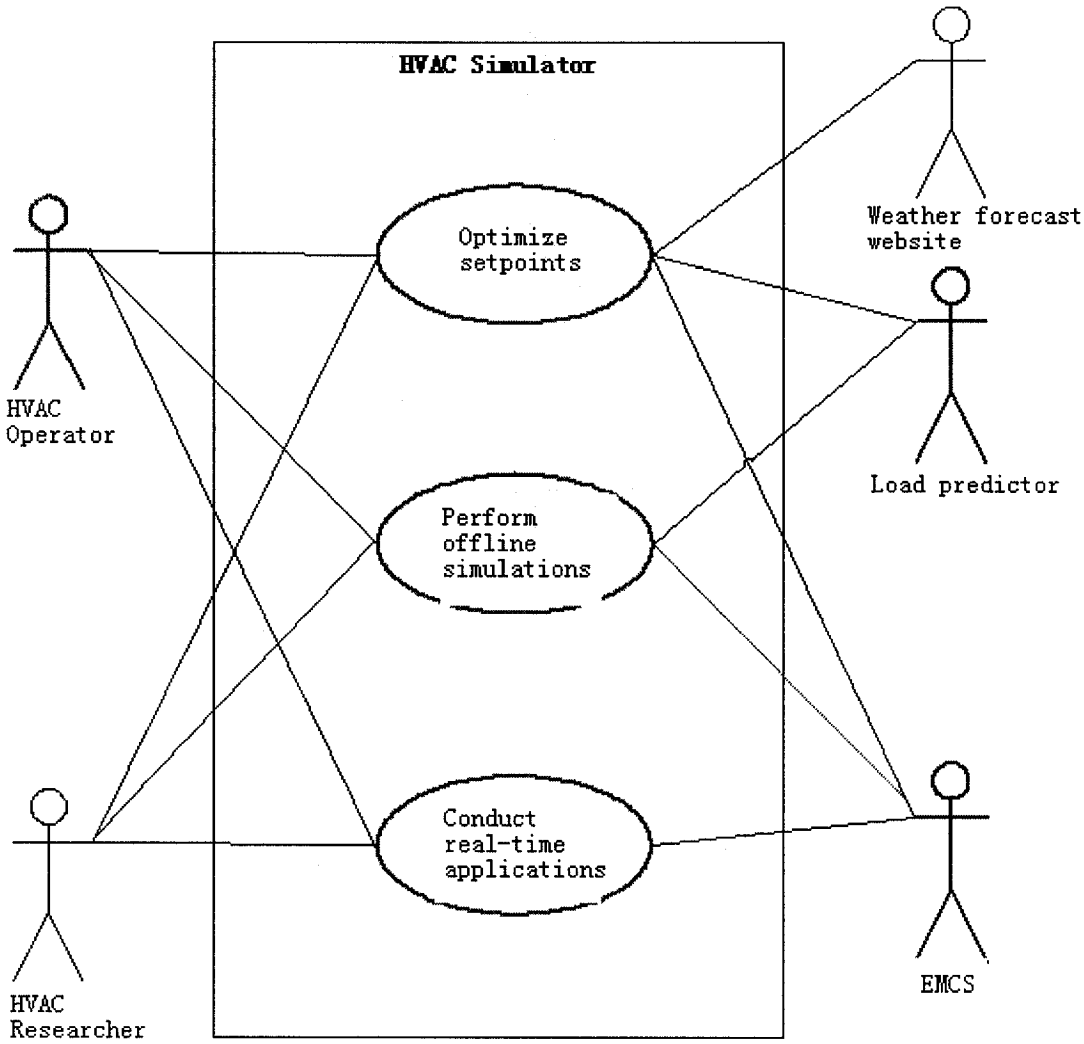


Figure 5-1 Use case diagram

5.1.2.2 Product Perspective

The *HVAC Simulator* will not be tied to any pre-existing system or database. It is independent and totally self-contained. The tool requires to run under a Windows environment: Windows 98/ME/2000/XP platforms.

For ease of use, the *HVAC Simulator* user interface will be implemented in a graphical user interface (GUI). The *HVAC Simulator* will print the output on the screen.

5.1.2.3 User Characteristics

Table 5.1 is a list of the primary actors in the system and their descriptions, corresponding to **Fig. 5-1**. ***Actor*** is a role that people or devices play as the system operates. ***Primary actor*** is an actor who interacts to achieve required system function and derive the intended benefit from the system. ***Secondary actor*** is the actor who supports the system so that primary actors can do their work (Ormandjieva, 2003).

Table 5.1 Descriptions of primary actors

Actors	Description
Operator	The HVAC operator has the ability to determine the optimal set of control variables. The HVAC operator also has the ability to perform off-line and on-line simulations.
Researcher	The same as Operator.

5.1.3 Use Case Analyses

Each use case is a set of scenarios. ***Scenario*** is an instance of the use case. ***Basic scenario*** is the normal flow for the use case. ***Alternative scenario*** is the what-if logic of the use case (Ormandjieva, 2003).

5.1.3.1 Use Case 1: Optimize setpoints

Use case 1, which is *Optimize Setpoints*, is listed in **Table 5.2** and shown in **Fig. 5-2a&b**.

Table 5.2 Descriptions of use case 1

Description	System determines the optimal set of control variables
Status	Detailed Description
Actors	HVAC operator, Weather forecast website, Load predictor Energy management and control system
Pre-Conditions	1. System must be loaded 2. Weather forecast website outputs outdoor air temperature 3. Load predictors outputs building loads
Flow of Events	
Basic Scenario	1. A dialog box prompts HVAC operator to enter outdoor air temperature and building loads 2. HVAC operator enters the required data 3. System determines the optimal set of control variables 4. System displays the optimal set of control variables
Alternative Scenario	Step 3a: System cannot determine the optimal set of control variables based on the given information Step 4a: System displays the message that the optimal set is not available and reheat control strategy should be used.
Post-Conditions	System displays the optimal set of control variables
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

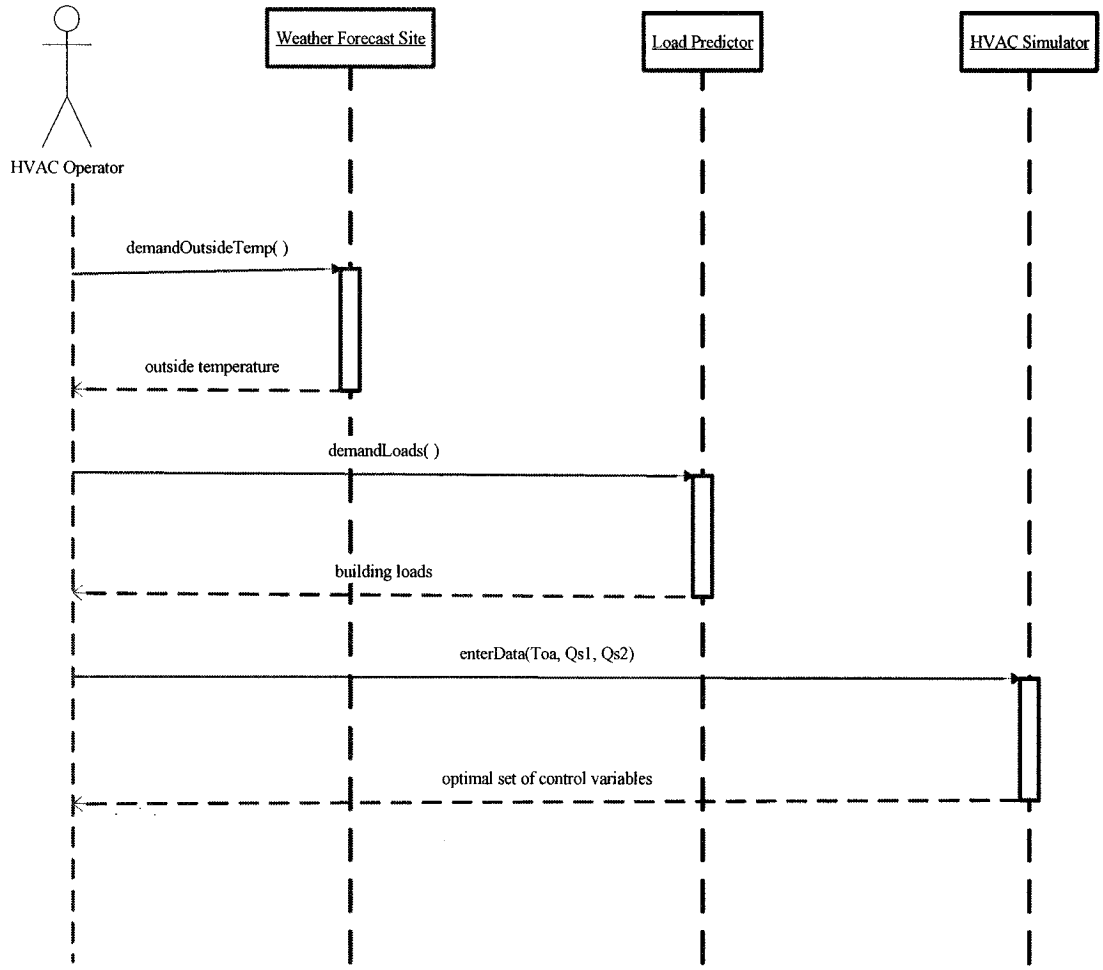


Figure 5-2a System sequence diagram (use case 1: basic scenario)

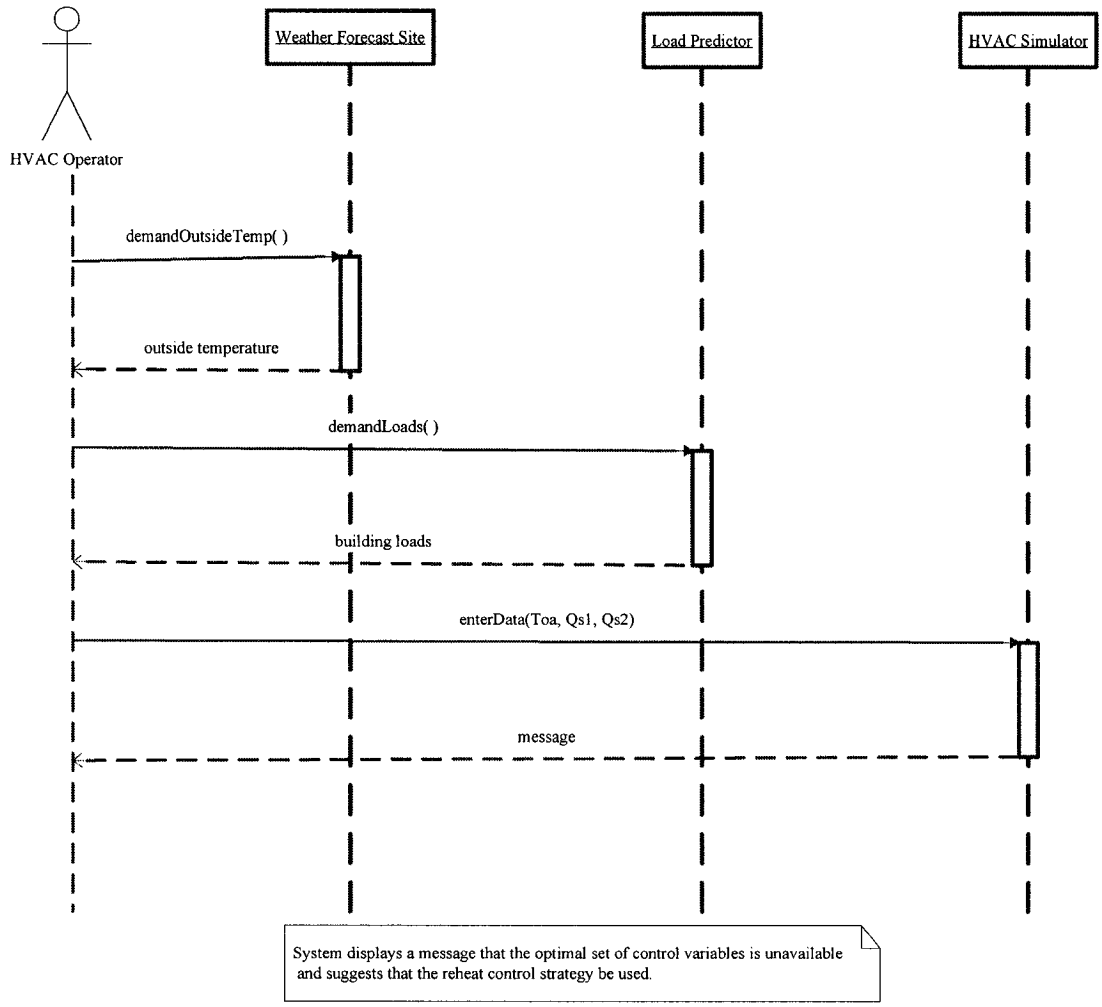


Figure 5-2b System sequence diagram (use case 1: alternative scenario)

5.1.3.2 Use Case 2: Perform off-line simulations

Use case 2, which is *Perform Off-line Simulations*, is listed in **Table 5.3** and shown in **Fig. 5-3a&b**.

Table 5.3 Descriptions of use case 2

Description	HVAC operator performs offline simulations using the optimal set of control variables
Status	Detailed Description
Actors	HVAC operator, Energy management and control system
Pre-Conditions	The optimal set of control variables must be available
Flow of Events	
Basic Scenario	1. HVAC operator enters discharge air temperature setpoint, zone temperature setpoints, chilled water supply temperature, outside air temperature and zone sensible loads 2. HVAC operator selects optimal control strategy 3. HVAC operator selects simulation mode 4. HVAC operator enters the required data 5. System simulates performance of the modeled HVAC system 6. System displays discharge air temperature and zone temperatures
Alternative Scenario	Step 2a: HVAC operator selects reheat control strategy
Post-Conditions	Diagrams of discharge air temperature and zone temperatures are displayed in separate windows
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

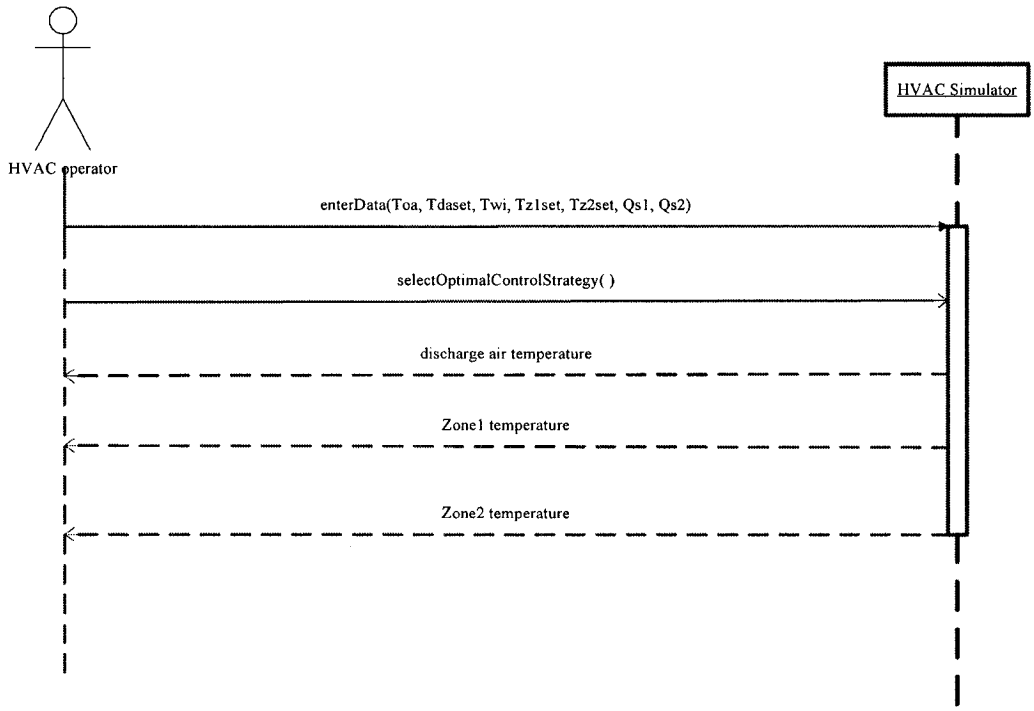


Figure 5-3a System sequence diagram (use case 2: basic scenario)

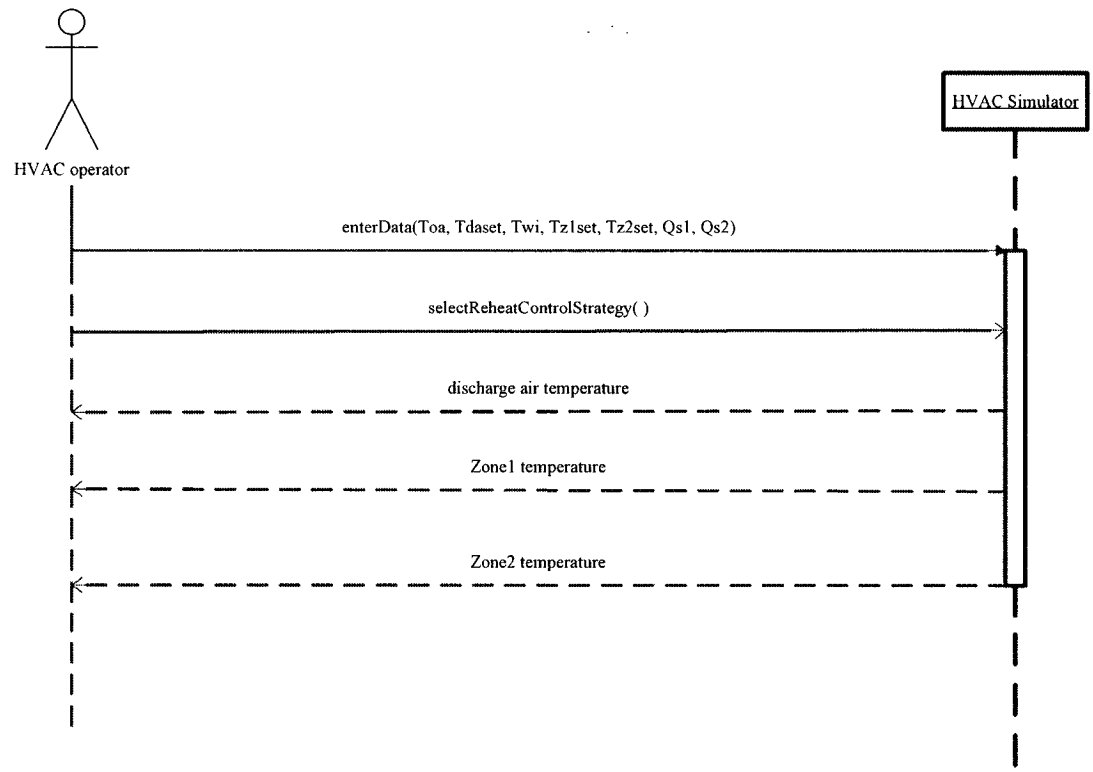


Figure 5-3b System sequence diagram (use case 2: alternative scenario)

5.1.3.3 Use Case 3: Conduct real-time applications

Use case 3, which is *Conduct real-time applications*, is listed in **Table 5.4** and shown in **Fig. 5-4**.

Table 5.4 Descriptions of use case 3

Description	HVAC researcher conducts virtual experiments using real-time applications
Status	Detailed Description
Actors	HVAC researcher, Energy management and control system
Pre-Conditions	None
Flow of Events	
Basic Scenario	1. HVAC researcher selects real-time mode 2. HVAC researcher enters the required data 3. System displays discharge air temperature and zone temperatures while synchronized to the real clock
Alternative Scenario	None
Post-Conditions	Discharge air temperature and zone temperatures are displayed in separate windows.
Related Use Cases	
Used Use Cases	None
Extending Use Cases	None

5.1.4 Interface Requirements

This section describes the user, hardware, software, and communications interfaces for *HVAC Simulator*.

5.1.4.1 User Interface

The user interface, which is shown in **Fig. 5-5**, includes three areas: (1) *information area*,

(2) *running mode selection area*, and (3) *result display area*. The information area includes three parts: Zone1 information, Zone2 information, and air handling unit information. From this area, the user can input the required information. From the running mode area the user selects a running mode: off-line simulations or real-time applications. The result display area consists of three buttons, which can be used to display discharge air temperature, Zone1 temperature and Zone2 temperature respectively.

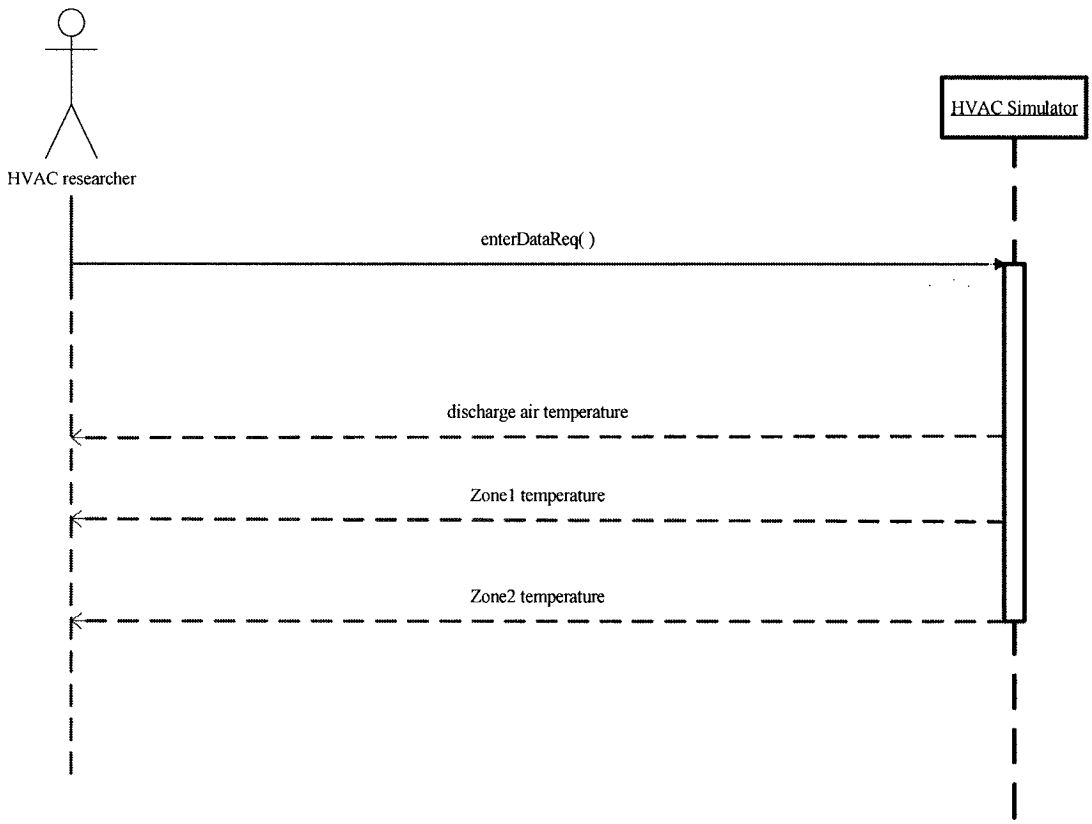


Figure 5-4 System sequence diagram (use case 3)

5.1.4.2 Hardware Interface

HVAC Simulator interfaces with a printer to print the result and a monitor to display it.

5.1.4.3 Software Interface

HVAC Simulator runs under Microsoft Windows 98/ME/2000/XP platform.

5.1.4.4 Communication Interface

There is no direct communication interface connected with *HVAC Simulator*.

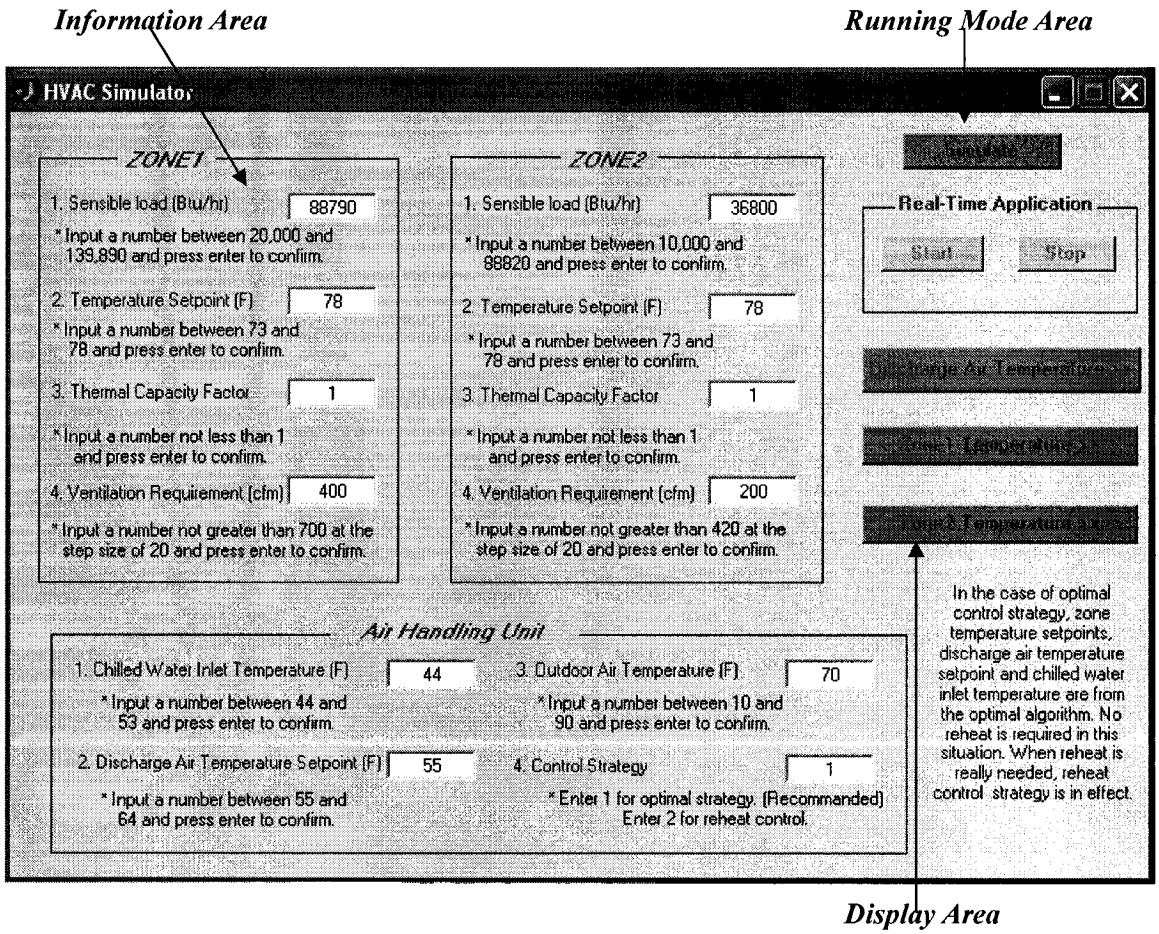


Figure 5-5 Graphical user interface

5.2 Results and Analyses

The modeled coil in this study is a typical counter-across flow water-to-air heat exchanger of the continuous plate-fin-tube. It uses 0.5 in. copper tubes and aluminum fins in a triangular layout. The coil parameters are shown in **Table 5.5**.

The modeled system consists of two office rooms, which are located in Montreal, Canada. Zone 1 has one south-facing external wall with three double-glazing windows, one west-facing external wall and two interior walls. Each window is 5.9 ft high and 7.9 ft wide. Zone 2 has one south-facing external wall and three interior walls. The system parameters are listed in **Table 5.6**.

Table 5.5 Coil characteristics parameters

Items		**Face area** (sq.ft)	**Number of fins per inch**	**Number of tubes per row**	**Number of rows**
Cooling coil		11.2	12	23	10
Reheat coil	***Zone1***	2	8	3	2
	Zone2	2	4	2	2

Table 5.6 Zone design parameters

Items		Symbols	Units	Magnitudes
Zone1	Floor area	A_1	sq.ft	5,000
	Volume	V_1	cu.ft	50,000
	Number of people		persons	35
	Design temperature	T_{z1}	F	78
	Design relative humidity	Φ_1	%	50
	Outdoor air requirement	CFM_{oa1d}	cfm	700
	Sensible load	Q_{s1d}	Btu/hr	139,890
	Supply airflow rate	CFM_{sa1d}	cfm	5,530
Zone2	Floor area	A_2	sq.ft	3,000
	Volume	V_2	cu.ft	30,000
	Number of people		persons	21
	Design temperature	T_{z2}	F	75
	Design relative humidity	Φ_2	%	50
	Outdoor air requirement	CFM_{oa2d}	cfm	420
	Sensible load	Q_{s2d}	Btu/hr	88,820
	Supply airflow rate	CFM_{sa2d}	cfm	4,040

5.2.1 Off-line Simulations

5.2.1.1 Case 1

A two-zone VAV-TRH system was simulated. The operating conditions correspond to the design day weather conditions. The design parameters are listed in **Tables 5.5** and **5.6**. Under these conditions, the optimization tool - *Optimizer* was invoked. The results from *Optimizer* and the corresponding inputs are depicted in **Table 5.7**. It is found that *Optimizer* determined that AHU operation be Mode 1 (i.e., mechanical cooling with minimum outdoor air intake). Since this case corresponds to design conditions, *Optimizer* chose the zone temperature setpoints to be the upper bound (i.e., 78F) and discharge air temperature setpoint to be the lower bound (i.e., 55F). By supplying these values as setpoints to the off-line simulator, the system performance was simulated. The results are depicted in **Fig. 5-6**. Note that the temperatures reach steady state in about 2,000 seconds.

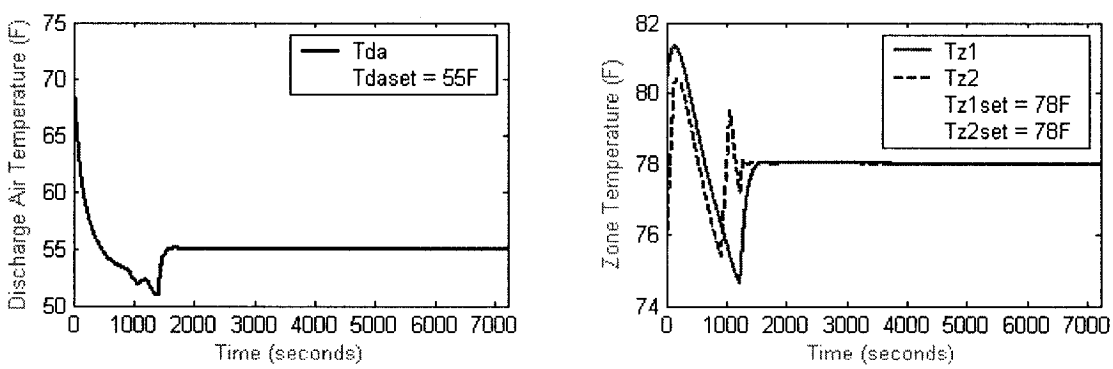


Figure 5-6 Response of temperatures

Table 5.7 Inputs and outputs of *Optimizer*

INPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Sensible loads	Zone 1	Q_{s1}	Btu/hr	124,220
	Zone 2	Q_{s2}	Btu/hr	88,820
Outdoor air temperature		T_{oa}	F	85
Outdoor air intake		CFM_{oa}	cfm	1,120
OUTPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Zone temperature setpoints	Zone 1	T_{z1set}	F	78
	Zone 2	T_{z2set}	F	78
Discharge air temperature setpoint		T_{daset}	F	55
Chilled water supply temperature		T_{wi}	F	44
AHU operation mode				1

5.2.1.2 Case 2

This case simulated the situation where mechanical cooling was required during the intermediate seasons (i.e., spring and fall). The inputs and outputs of *Optimizer* are listed in **Table 5.8**. In this case, the optimal control strategy was practical. *Optimizer* determined that AHU operation be Mode 2 (i.e., mechanical or natural cooling with 100% outdoor air).

Moreover, chilled water supply temperature was raised from 44F to 51.5F. Accordingly, 15% of energy was saved in the chiller. The dynamic performance of the modeled system is shown in **Fig. 5-7**. It is noted that mechanical cooling would still be required since outdoor air temperature (i.e., 70F) was greater than discharge air temperature setpoint (i.e., 62.5F).

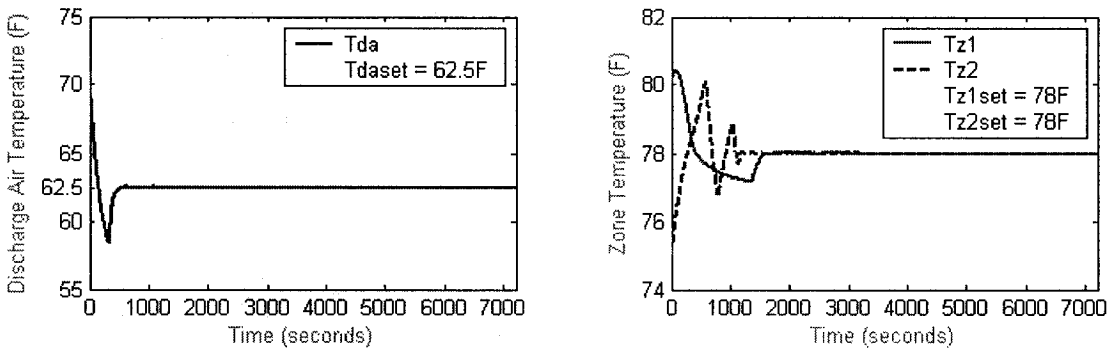


Figure 5-7 Response of temperatures

5.2.1.3 Case 3

This case simulated the situation where cooling was still required in the cold days (e.g., winter). This is practical because the interior zone of a large commercial building requires cooling all year round. The inputs and outputs of *Optimizer* are listed in **Table 5.9**. In this case, the optimal control strategy was still valid. *Optimizer* determined that AHU operation be Mode 3 (i.e., natural cooling with outdoor air). In this mode, since no mechanical cooling is required, *Optimizer* will give the discharge air temperature setpoint such that supply (return) fan power requirement were minimized. It is apparent that the optimal strategy chooses larger setpoint for Zone 1 and lower setpoint for Zone 2 to

minimize overall energy. The zone temperature responses are depicted in **Fig. 5-8**.

Table 5.8 Inputs and outputs of *Optimizer*

INPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Sensible loads	Zone 1	Q_{s1}	Btu/hr	88,790
	Zone 2	Q_{s2}	Btu/hr	36,800
Outdoor air temperature		T_{oa}	F	70
Outdoor air intake		CFM_{oa}	cfm	800
OUTPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Zone temperature setpoints	Zone 1	T_{z1set}	F	78
	Zone 2	T_{z2set}	F	78
Discharge air temperature setpoint		T_{daset}	F	62.5
Chilled water supply temperature		T_{wi}	F	51.5
AHU operation mode				2

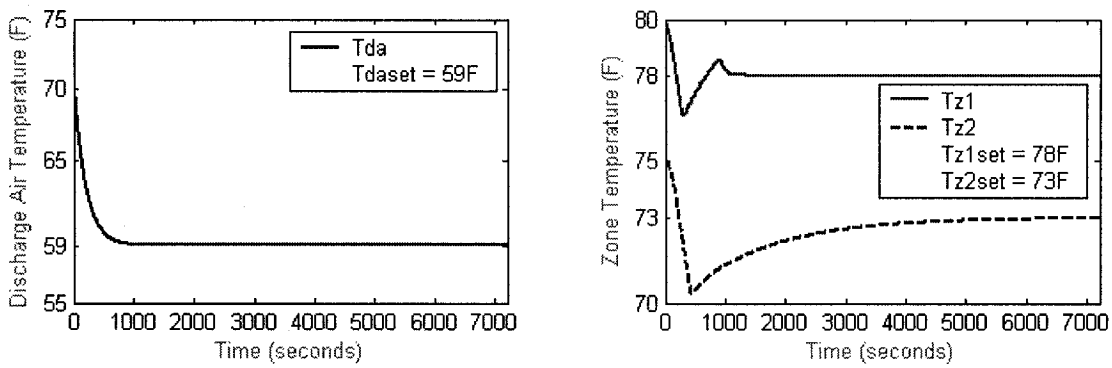


Figure 5-8 Response of temperatures

Table 5.9 Inputs and outputs of *Optimizer*

INPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Sensible loads	Zone 1	Q_{s1}	Btu/hr	43,860
	Zone 2	Q_{s2}	Btu/hr	18,740
Outdoor air temperature		T_{oa}	F	52
Outdoor air intake		CFM_{oa}	cfm	600
OUTPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Zone temperature setpoints	Zone 1	T_{z1set}	F	78
	Zone 2	T_{z2set}	F	73
Discharge air temperature setpoint		T_{daset}	F	59
AHU operation mode				3

5.2.1.4 Case 4

This case simulated the case where space loads were far less than the peak values. The input and the output of *Optimizer* are listed in **Table 5.10**. It was obvious that Zone 2 temperature setpoint was beyond the acceptable range (i.e., 73 thru 78F), which meant optimal control strategy was not viable. When this occurs, reheat control strategy (i.e., *Reheat Analyzer*) has to be utilized, which determines the discharge air temperature setpoint and which zone requires reheat. **Table 5.11** lists the application of this program. **Fig. 5-9** shows how the HVAC system behaved when reheat coil of Zone 2 had been activated. Since a two-position (on-off) control was used in the reheat coil, Zone 2 temperature shows typical on-off cyclic variations.

Table 5.10 Inputs and outputs of *Optimizer*

INPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Sensible loads	Zone 1	Q_{s1}	Btu/hr	21,000
	Zone 2	Q_{s2}	Btu/hr	10,000
Outdoor air temperature		T_{oa}	F	52
Outdoor air intake		CFM_{oa}	cfm	600
OUTPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Zone temperature setpoints	Zone 1	T_{z1set}	F	76.8
	Zone 2	T_{z2set}	F	72.8
Discharge air temperature setpoint		T_{daset}	F	64
AHU operation mode				3

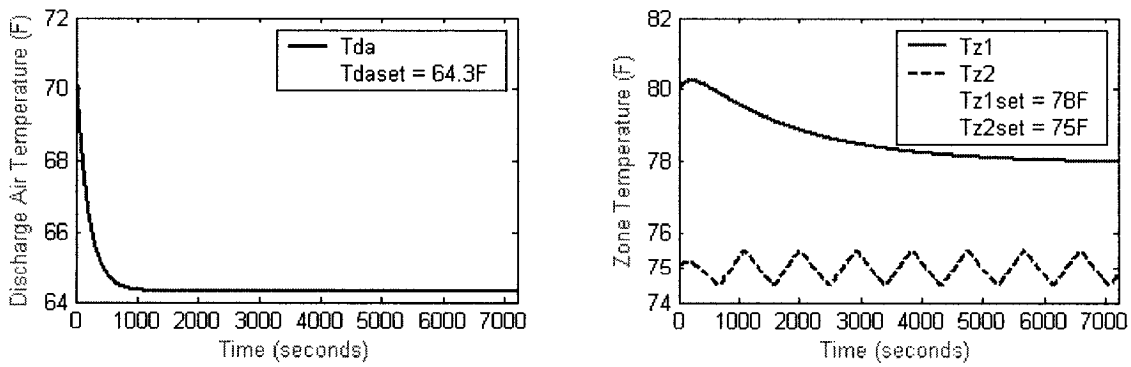


Figure 5-9 Response of temperatures

Table 5.11 Application of *Reheat Analyzer*

INPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Zone1	Sensible load	Q_{s1}	Btu/hr	21,000
	Peak sensible load	Q_{s1d}	Btu/hr	139,890
	Max. supply airflow rate	CFM_{sa1d}	cfm	5,530
	Temperature setpoint	T_{z1set}	F	78
Zone2	Sensible load	Q_{s2}	Btu/hr	10,000
	Peak sensible load	Q_{s2d}	Btu/hr	88,820
	Max. supply airflow rate	CFM_{sa2d}	cfm	4,040
	Temperature setpoint	T_{z2set}	F	75
OUTPUTS				
Items		**Symbols**	**Units**	**Magnitudes**
Discharge air temperature setpoint		T_{daset}	F	64.3
Reheat mode				22

5.2.2 Real-Time Simulations

A real-time application was realized to carry out virtual experiments that give graphical results that mimic experimental test. In this way, visualization of system dynamical behavior was plotted over a chosen time range, which was two hours, used in this thesis.

Two virtual experiments were conducted to test dynamic behavior of the two-zone VAV-TRH HVAC system when different space loads and ambient conditions were acting, each lasting two hours. The details of the experiments are listed in **Tables 5.12** and **5.13** and the corresponding results are shown in **Fig. 5-10** and **5-11** respectively.

5.2.2.1 Experiment 1

In this experiment, the outdoor temperature varied from 87 to 75F over a period of two hours as shown in **Table 5.12**. For example, T_{oa} = 87F for the first 2,000 seconds and T_{oa} = 85F from 2,003 to 3,600 seconds, and so on. The assumed zone load variations and outdoor airflow rates are also listed in **Table 5.12**. Given these inputs, the optimization tool – *Optimizer* selects AHU mode of operation and determines optimal setpoints. The optimal zone temperatures were 78F and discharge air temperature setpoint was 55F. By selecting the real-time operation from the menu, a real-time simulation was conducted. The results are depicted in **Fig. 5-10**. Note that even though outdoor air temperatures and loads acting on the zones are changing, the PI controllers are able to regulate zone temperatures and discharge air temperature close to their respective setpoints throughout

two-hour period. It should be noted that the real-time simulations shown in **Fig. 5-10** and **5-11** correspond to clock time and therefore the plots are dynamic in nature and require two-hour clock time to complete. This is quite useful in monitoring and control of HVAC systems as they are operated in real time using EMCS systems.

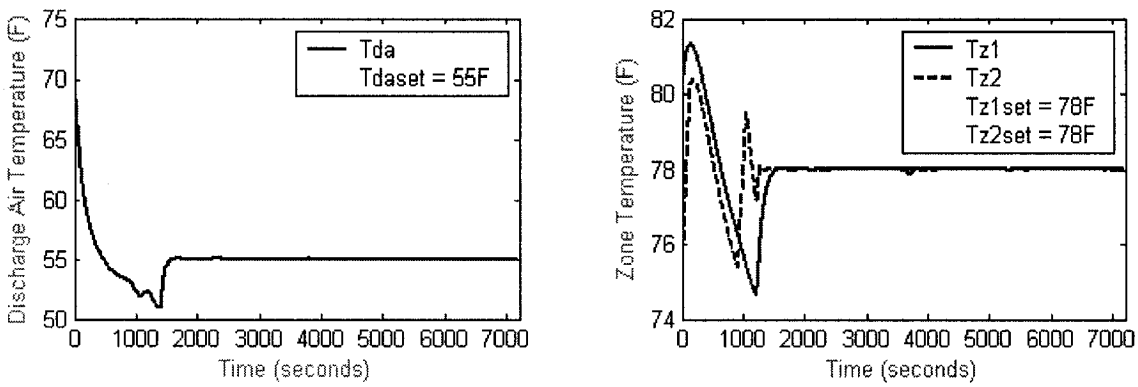


Figure 5-10 Response of temperatures

5.2.2.2 Experiment 2

This experiment simulated a transition from low-load to high-load operation of the two-zone VAV-TRH HVAC system. Note that over two-hour operation the discharge air temperature setpoint as computed by *Optimizer* decreased from 64 to 55F as shown in **Table 5.13** and the corresponding PI control response is shown in **Fig. 5-11**.

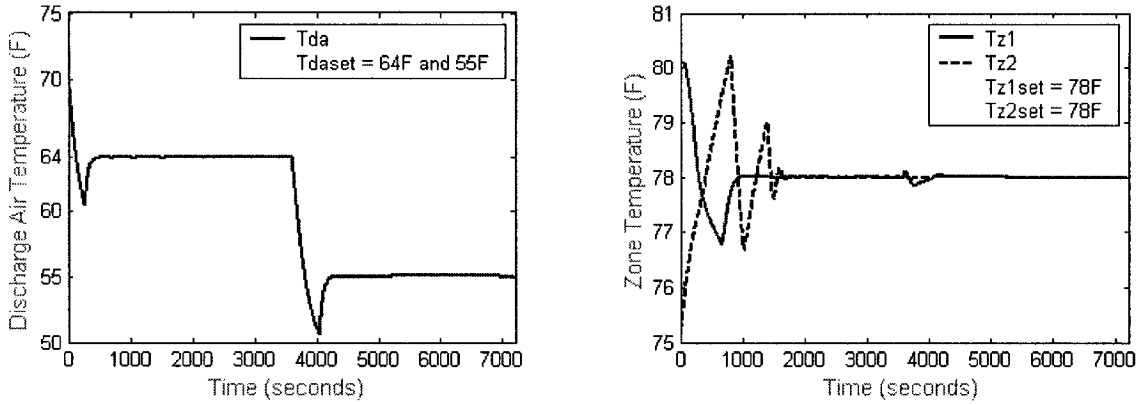


Figure 5-11 Response of temperatures

5.3 Summary and Conclusions

A Simulink model of a two-zone VAV-TRH HVAC system was developed so that simulations could be performed to see the dynamic behaviors of the HVAC system. The simulation model combined with *Optimizer* - the program based on the optimal control strategy - is a powerful tool for the HAVC staff to realize the optimal operation. In addition, the simulation model can be used for commissioning for the HVAC system. On the basis of the Simulink model, a virtual HVAC system of this kind was implemented. As a result, virtual experiments can be performed just as real experiments are performed in physical HVAC systems. In Chapter 6, this virtual system will be used to help test the expert rules for on-line fault detection and diagnosis of HVAC systems.

Table 5.12 Inputs and outputs of Experiment 1

Time (seconds)			**Start**	0	2,003	3,603	5,203
			End	2,000	3,600	5,200	7,200
INPUTS							
Items		**Symbols**	**Units**	**Magnitudes**			
Outside air temp.		T_{oa}	F	87	85	81	75
Outside air intake		CFM_{oa}	cfm	1,120			
Sensible loads	Zone 1	Q_{s1}	Btu/hr	124,220		134,320	
	Zone 2	Q_{s2}	Btu/hr	88,820		63,750	
OUTPUTS							
Zone temp setpoint	Zone 1	T_{z1set}	F	78			
	Zone 2	T_{z2set}	F	78			
Discharge air temp. setpoint		T_{daset}	F	55			
Chilled water inlet temp.		T_{wi}	F	44			
AHU operation mode				1			2

Table 5.13 Inputs and outputs of Experiment 2

Time (seconds)		**Start**	0	2,003	3,603	5,203
		End	2,000	3,600	5,200	7,200
INPUTS						
Items		**Symbols**	**Units**	**Magnitudes**		
Outside air temp.		T_{oa}	F	70	74	77	81
Outside air intake		CFM_{oa}	cfm	1,120		
Sensible loads	Zone 1	Q_{s1}	Btu/hr	71,960		88,790
	Zone 2	Q_{s2}	Btu/hr	30,660		36,800
OUTPUTS						
Zone temp setpoint	Zone 1	T_{z1set}	F	78		
	Zone 2	T_{z2set}	F	78		
Discharge air temp. setpoint		T_{daset}	F	64		55
Chilled water inlet temp.		T_{wi}	F	53		44
AHU operation mode				2		1

CHAPTER 6

ONLINE FAULT DETECTION AND DIAGNOSIS

6.1 Introduction

The heating, ventilating, and air-conditioning (HVAC) system is a complex electromechanical system. In such a system, faults tend to degrade the performance of the system. The performance of the HVAC system should be evaluated based on three aspects: (1) *occupant comfort*, (2) *energy consumption level*, and (3) *equipment life span*. Therefore, it can be concluded that occupants' feeling comfortable in a conditioned space does not necessarily mean that the HVAC system is running without any problems. For example, when the cooling coil is fouled, the cooling coil valve control signal increases in order to increase the chilled water flow rate; therefore, the heat transfer required to maintain the discharge air temperature at the setpoint is achieved. In this case, occupants will suffer no discomfort. However, the price of this fault is that energy consumption increases. Unfortunately, the latter two aspects have not been sufficiently emphasized because most problems with the HVAC system have been linked to occupant complaints. HVAC operators simply respond by checking zone temperatures or adjusting setpoints. This may work in some circumstances. However, the underlying causes of the problems are not diagnosed and let alone settled. As a result, the faults recur. If the staff is expected to find the reason, they will have to inspect equipment, sensors and even control

algorithms. The entire process might be rather tedious and time consuming. Furthermore, it depends much upon personal experience of the staff. Accordingly, it is extremely important for HVAC systems to have automatic fault detection and diagnosis tools to address these issues.

As its name indicates, the HVAC fault detection and diagnosis (FDD) system must possess two functions. First, an FDD system must distinguish a faulty condition from the normal condition, which is referred to as *detection*. It is realized by automatically and continuously checking the system performance and bringing the problem it has found to the attention of building operators (Katipamula et al. 1999). In addition to fault detection, an FDD system should do *diagnosis*. That is, it should be isolating the fault and suggesting the proper settlement. Compared to the first function, the second one is much harder to realize. In a typical HVAC system, there are a huge number of potential faults that can occur but few of them have unique symptoms. In most cases, faults and their symptoms are interrelated, i.e., one fault may have several symptoms and one symptom may be caused by several faults. Therefore, diagnostics of multiple faults is difficult and it is nearly impossible to determine which one is the original or primary source of the fault using an FDD system (Han et al. 1999). However, an FDD system will provide several possible explanations for HVAC operators to perform further investigations. This is still considered as an acceptable solution; otherwise many of these problems will go undetected. In summary, an FDD system helps improve HVAC operation and bring

comfort and lower operation costs.

In this chapter, expert rules specific for on-line monitoring HVAC system performance, detection of abnormalities, and diagnosis of system faults adapted from the literature were used to develop a real-time FDD tool. The next sections will present (1) summary of HVAC faults, (2) discussion of FDD approaches, (3) set-up of expert rules, (4) case studies, and (5) conclusions.

6.2 HVAC Faults

Table 6.1 summarizes the most common faults that a two-zone variable-air-volume terminal reheat (VAV-TRH) HVAC system could experience. Some faults are adapted from the literature (Han et al. 1999, House et al. 2002, Katipamula et al. 1999). Most faults listed in **Table 6.1** are self-explanatory and they are caused by mechanical failure of components. Other faults such as poor tuning of controllers and faults in sensors are the cause of control loop instabilities or result in less than optimal operation.

6.3 FDD Methodologies

One approach to FDD is *model-based* fault detection. The validity of this approach depends greatly upon the accuracy of the system model. The system model must reflect the real system's behavior under all normal conditions. An inaccurate model may result in an incorrect assessment of system performance, either by ignoring a fault or wrongly reporting a fault (Han et al. 1999). As a matter of fact, modeling an HVAC system accurately is an extremely difficult task, and sometimes it is impossible to realize. In this situation, another approach to FDD, the so-called *knowledge-based* fault detection, is more viable.

This approach uses if –then rules to give a very clear explanation of the fault detection. Therefore, users tend to accept it. Most FDD systems based on *knowledge-based* approach consist of two components: *a statistical analysis preprocessor* and *a rule-based expert system.* The first component screens incoming data and estimates system operating parameters. The second component analyzes the collected data in terms of the estimates and expected operating conditions and returns a warning if there is one (Anderson et al.1989). Sophisticated mathematical methods must be used in both components to do data filtering. As a result, the algorithm becomes very complex. Furthermore, through this kind of system, on-line fault detection and diagnosis cannot be implemented because the data that FDD has collected and analyzed is based on previous measurements. Accordingly,

there is a time lag between problem's occurring and problem's being detected. It is impossible for HVAC operators to respond to the problems in real time. Real-time fault detection and diagnosis (FDD) system using *knowledge-based* approach mitigates this shortcoming. Real time refers to events run by a computer at the same speed that they would occur in real life. In graphics animation, for example, a real-time program would display objects moving across the screen at the same speed that they would actually move. Real-time FDD system runs while synchronized to a real HVAC system, i.e., this system monitors on-line the HVAC system performance. In this type of system, the algorithm is much simpler because there is no need to introduce complicated mathematical theories to analyze data. More important, faults will be detected and diagnosed as soon as they occur so that HVAC operators can take timely actions. So far, some researchers have established a set of expert rules to detect and diagnose faults in AHUs. However, they are not suitable for real-time FDD system. Such a real-time FDD system appears to be lacking. In this chapter, expert rules for real-time fault detection and diagnosis will be built and tested. It should also be noted that it is impractical to establish a generic FDD rules to assess every kind of HVAC system. The rule set presented in this chapter is only used for two-zone VAV-TRH HVAC system and could be extended to the single duct HVAC system with reheat coils powered by hot water.

Table 6.1 Summary of HVAC faults

HVAC COMPONENTS	FAULTS	FAULT TYPES
Cooling Coil	1. Cooling coil valve stuck	Component fault
	2. Cooling coil valve leaking	Component fault
	3. Undersized cooling coil	Component fault
	4. Fouled cooling coil	Component fault
VAV Boxes	5. Zone 1 supply air damper stuck	Component fault
	6. Zone 2 supply air damper stuck	Component fault
Zones	7. Zone 1 design supply airflow rate too low	Component fault
	8. Zone 2 design supply airflow rate too low	Component fault
Reheat Coils	9. Zone 1 reheat coil valve stuck	Component fault
	10. Zone 1 reheat coil valve leaking	Component fault
	11. Zone 2 reheat coil valve stuck	Component fault
	12. Zone 2 reheat coil valve leaking	Component fault
Mixing Box	13. Return air damper leaking	Operational fault
	14. Return air damper stuck	Component fault
	15. Outdoor air damper stuck	Component fault
HVAC Controllers	16. Chilled water PI controller tuning error	Operational fault
	17. Zone 1 VAV box PI controller tuning error	Operational fault
	18. Zone 2 VAV box PI controller tuning error	Operational fault
Chilled Water Circulating Pump	19. Chilled water circulating pump fault	Component fault
	20. Undersized chilled water circulating pump	Component fault
Chiller	21. Chilled water supply temperature too high	Component fault
Supply Fan	22. Supply fan fault	Component fault
	23. Undersized supply fan	Component fault
Sensors	24. Outdoor air temperature sensor error	Operational fault
	25. Return air temperature sensor error	Operational fault
	26. Mixing air temperature sensor error	Operational fault
	27. Discharge air temperature sensor error	Operational fault
	28. Supply air temperature sensor error	Operational fault
	29. Zone temperature sensor error	Operational fault

6.4 Expert Rules

The expert rules are generated from several sources. Interviews are held with on-site personnel, building engineers, and technicians to identify the specific operating characteristics of the building. Also, general HVAC information is obtained from experts and facilities management personnel at other locations (Anderson et al.1989). In general, expert rules come from knowledge of HVAC experts. In this thesis, some expert rules are adapted from the work of House et al. (2002).

In this chapter, the expert rules are categorized according to the operation modes of the AHU. As defined in Chapter 4, the AHU has three operation modes: (1) *mechanical cooling with minimum outdoor air intake* (***Mode#1***), (2) *mechanical or natural cooling with 100% outdoor air* (***Mode#2***), and (3) *natural cooling with outdoor air* (***Mode#3***). Among these rules, those common for at least two operation modes are referred to as common rules while others are specific rules. All rules are written such that a fault or warning will be indicated if the expression is satisfied. They are easily understood by HVAC system operators. The rule set is enhanced in this thesis to develop a FDD tool for VAV HVAC systems.

The nomenclature corresponds to the one defined in this thesis. The set of equations which describe steady state energy balance on each local process are summarized below.

6.4.1 Common Rules

Twelve common rules are listed as follows:

1) Rule 1. $T_{sa} > T_{ma} + \Delta T_{sf} + \varepsilon_t$

2) Rule 2. $T_{da} > T_{ra} - \Delta T_{rf} + \varepsilon_t$

3) Rule 3. $T_{da} > T_{daset} + \varepsilon_t$

4) Rule 4. $T_{da} < T_{daset} - \varepsilon_t$

5) Rule 5. $T_{da} \neq const$

6) Rule 6. $T_{sa} > T_{da} + \varepsilon_t$

7) Rule 7. $T_{sa} < T_{da} - \varepsilon_t$

8) Rule 8. $T_{sa} = const$

9) Rule 9. $T_z > T_{zset} + \varepsilon_t$

10) Rule10. $T_z < T_{zset} - \varepsilon_t$

11) Rule11. $T_z \neq const$

12) Rule12. $U_{wact} > U_{w\exp}$

where

T_{da} = discharge air temperature (F)

T_{ma} = mixed air temperature (F)

ΔT_{sf} = temperature rise across the supply fan (F), s*pecified by the user*

ε_t = threshold parameter accounting for errors in temperature measurements (F), *specified by the user*

T_{ra} = return air temperature (F)

ΔT_{rf} = temperature rise across the return fan (F), *specified by the user*

T_{daset} = discharge air temperature setpoint (F)

T_{sa} = supply air temperature (F)

T_z = zone temperature (F)

T_{zset} = zone temperature setpoint (F)

U_{wact} = actual control signal to the cooling coil valve (dimensionless)

$U_{w\exp}$ = expected control signal to the cooling coil valve (dimensionless)

Rules 1&2 indicate inconsistencies in temperatures between the supply air and mixed air, and between the discharge air and return air, respectively. Rules 3&4 indicate that the discharge air temperature setpoint cannot be reached. Rules 5&11 indicate the instability in temperatures of the discharge air and zone air, respectively. Rules 6&7 indicate inconsistencies in temperatures between the discharge air and supply air. Rule 8 indicates that the supply air temperature does not fluctuate as it is supposed to when reheat is required. Rules 9&10 indicate that zone temperature setpoints cannot be reached. Rule 12 indicates inconsistencies between the actual control signal to the cooling coil valve and the expected value.

6.4.2 Rules for Mode#1

In Mode#1, the cooling coil valve is modulated to satisfy the discharge air temperature setpoint, and the mixing box dampers are set for minimum outdoor air intake.

1) Rule 13. $T_{oa} < T_{ra} - \varepsilon_t$

2) Rule 14. $R_{osact} \neq R_{os\,\exp}$

where

T_{oa} = outdoor air temperature (F)

R_{osact} = actual ratio of outdoor air to supply airflow rate (dimensionless)

$$= \frac{T_{ra} - T_{ma}}{T_{ra} - T_{oa}}$$

$R_{os\,\exp}$ = expected ratio of outdoor air to supply airflow rate (dimensionless)

$$= \frac{\dot{m}_{oa}}{\dot{m}_{sa}}$$

Rule 13 indicates that the outdoor air temperature is too low for this mode. Rule 14 indicates inconsistencies between the actual ratio of outdoor air intake to supply airflow rate and the expected ratio.

6.4.3 Rules for Mode#2

In Mode#2, the mixing box dampers are set for 100% outdoor air, 0% return air, and 100% exhaust air. When the outdoor air temperature is greater than the discharge air temperature setpoint, the cooling coil valve is modulated to satisfy the discharge air temperature setpoint. When the values of the above two temperatures are the same, the cooling coil valve is closed and natural cooling is utilized.

1) *Rule 15.* $T_{oa} < T_{daset} - \Delta T_{sf} - \varepsilon_t$

2) *Rule 16.* $T_{oa} > T_{ra} + \varepsilon_t$

3) *Rule 17.* $T_{oa} \geq T_{ma} + \varepsilon_t$

4) *Rule 18.* $T_{oa} \leq T_{ma} - \varepsilon_t$

Rules 15 &16 indicate that outdoor air temperature is too low and too high for this operation mode, respectively. Rules 17&18 indicate inconsistencies in temperatures between outdoor air and mixed air.

6.4.4 Rules for Mode#3

In Mode#3, the mixing box dampers are controlled to maintain the discharge air temperature at the setpoint value and no mechanical energy is required.

1) *Rule 19.* $T_{oa} > T_{daset} - \Delta T_{sf} + \varepsilon_t$

2) *Rule 20.* $T_{sa} < T_{ma} + \Delta T_{sf} - \varepsilon_t$

Rule 19 indicates that the outdoor air temperature is too high for this operation mode. Rule 20 indicates inconsistencies in temperatures between the supply air and mixed air.

6.4.5 Uncertainty in Threshold Values

The magnitudes of threshold values are normally obtained from operating experience. The normal range of ε_t used in the literature is 1.5°C (House et al. 2002). The uncertainty in the magnitudes of ε_t results in incorrect diagnosis of faults, which is referred to as false alarms. From a practical point of view, the range of uncertainty in ε_t is +10% to -20% from its normal range. A bias towards lower range is preferred to avoid missing the potential faults even though the frequency of false alarms could increase.

6.4.6 Possible Explanations

When a rule is satisfied, a fault must be occurring. However, in most cases, there is more than one cause for fault. That is to say, there are several explanations for one fault. Accordingly, a faulty HVAC component can seldom be uniquely located by only one satisfied rule. In this situation, the method of decision tree can be used to locate the

defective component or at least to reduce the scope of further investigations to the greatest degree. By referring to the work of House et al. (2002), the possible explanations of satisfied rules are tabulated, as shown in **Table 6.2**.

6.5 Case Studies

The validity and robustness of the expert rules were tested by embedding a fault in the model to simulate a faulty HVAC system. Since the expert rules are used for real-time FDD system, it is reasonable and reliable to conduct tests using real-time simulations rather than nonreal-time simulations. In this study, 100-minute virtual experiments were performed with data computed at 3-second time intervals. And the occurrence of simultaneous faults was assumed to be impossible. By this way, nineteen cases were studied. In this section six typical cases of fault experiments are examined and the other cases are presented in *Appendix C*.

Table 6.2 Possible Explanations of Satisfied Rules

Rule	Mode	Explanation											
		Cooling Coil Valve Stuck	Cooling Coil Valve Leaking	Undersized Cooling Coil	Fouled Cooling Coil	VAV Box Damper Stuck	Design Supply Airflow Rate Too Low	Reheat Coil Valve Stuck	Reheat Coil Valve Leaking	Return Air Damper Leaking	Return Air Damper Stuck	Outdoor Air Damper Stuck	Chilled Water PI Controller Tuning Error
1	1,2,3												
2	1,2,3												
3	1,2,3	■		■							■	■	■
4	1,2,3	■	■								■	■	■
5	1,2												■
6	1,2,3							■	■				
7	1,2,3												
8	1,2,3							■					
9	1,2,3	■		■	■	■	■	■				■	
10	1,2,3	■				■		■	■				
11	1,2,3												
12	1,2				■								
13	1												
14	1										■	■	
15	2												
16	2												
17	2												
18	2									■	■		
19	3												
20	3	■	■										

Table 6.2 Possible Explanations of Satisfied Rules (Continued)

Rule	Mode	Explanation											
		VAV Box PI Controller Tuning Error	Chilled Water Circulating Pump Fault	Undersized Chilled Water Pump	Chilled Water Temperature Too High	Supply Fan Fault	Undersized Supply Fan	Outdoor Air Temperature Sensor Error	Return Air Temperature Sensor Error	Mixing Air Temperature Sensor Error	Discharge Air Temperature Sensor Error	Supply Air Temperature Sensor Error	Zone Air Temperature Sensor Error
1	1,2,3									■	■		
2	1,2,3								■		■		
3	1,2,3		■	■	■						■		
4	1,2,3										■		
5	1,2												
6	1,2,3										■	■	
7	1,2,3											■	
8	1,2,3											■	
9	1,2,3	■	■	■	■	■	■						■
10	1,2,3	■											■
11	1,2,3	■											
12	1,2												
13	1							■	■				
14	1												
15	2							■					
16	2							■	■				
17	2							■		■			
18	2							■		■			
19	3							■					
20	3									■		■	

6.5.1 Case 1 – A Stuck Cooling Coil Valve

Since the control signal to the cooling coil valve modulates to maintain the discharge air temperature at its associated setpoint, this fault will eventually cause the control signal to saturate at one of its limit, depending on the relation between the actual cooling capacity and instantaneous space loads. When the cooling coil valve is stuck at such a position that less chilled water enters the cooling coil, the consequent cooling capacity will not suffice for space loads. As a result, air entering the coil will be undercooled, causing the discharge air temperature to rise. Then the control signal to VAV damper has to increase so that more conditioned air will be delivered to the zones to compensate this fault. Eventually, the discharge air temperature will be driven to a value well above the setpoint. In this situation, the control signal to the cooling coil will be saturating at the upper limit, i.e., 100% open. On the other hand, when the cooling coil valve is stuck at such a position that excessive chilled water enters the cooling coil, the consequent cooling capacity will be greater than required. As a result, air leaving the coil will be overcooled, causing the discharge air temperature to decrease. Then the control signal to VAV damper has to go down so that less conditioned air will be delivered to the zones to compensate this fault. Eventually, the discharge air temperature will be driven to a value well below the setpoint. In this situation, the control signal to the cooling coil will be saturating at the lower limit, i.e., 100% closed.

Two tests were performed to simulate these two situations, where the discharge air temperature setpoint was 55F (i.e., T_{daset} = 55F). Test 1 was simulated by causing the

cooling coil valve to stick at t = 519s at 30% open position. In the meantime, design loads were acting on the system. As shown in **Fig. 6-1**, the discharge air temperature was 64.3F (i.e., T_{da} = 64.3F), much higher than the setpoint value. It should also be noted that the zone temperatures (T_{z1} = 84.5F and T_{z2} = 84.1F) were greater than their respective setpoint values. This was because design loads could not be satisfied even though VAV damper control signals had saturated. Test 2 was simulated by causing the cooling coil valve to stick at t = 2,208s at 70% open position. In the meantime, the system was undergoing small partial loads. As shown in **Fig. 6-2**, the discharge air temperature was 50.3F (i.e., T_{da} = 50.3F), much less than the setpoint. However, unlike Test 1, both zone temperature setpoints were reached. This was because this fault could be compensated by decreasing VAV damper control signals.

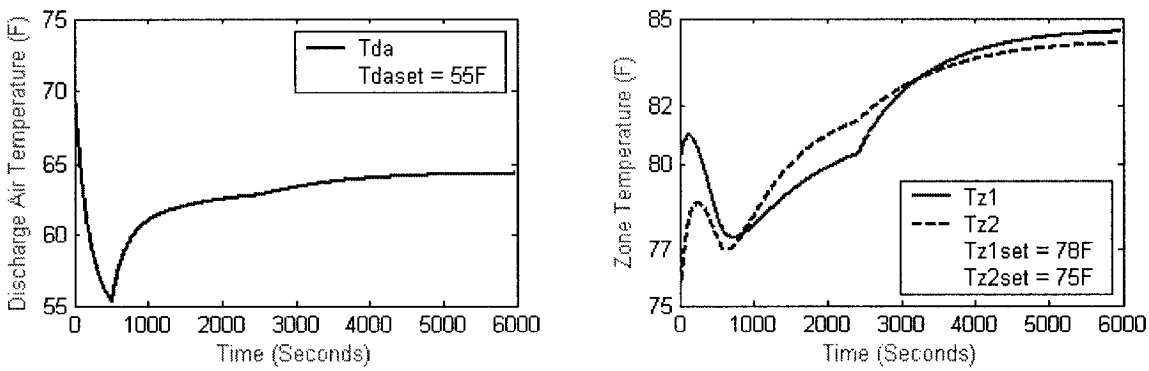


Figure 6-1 Response of temperatures

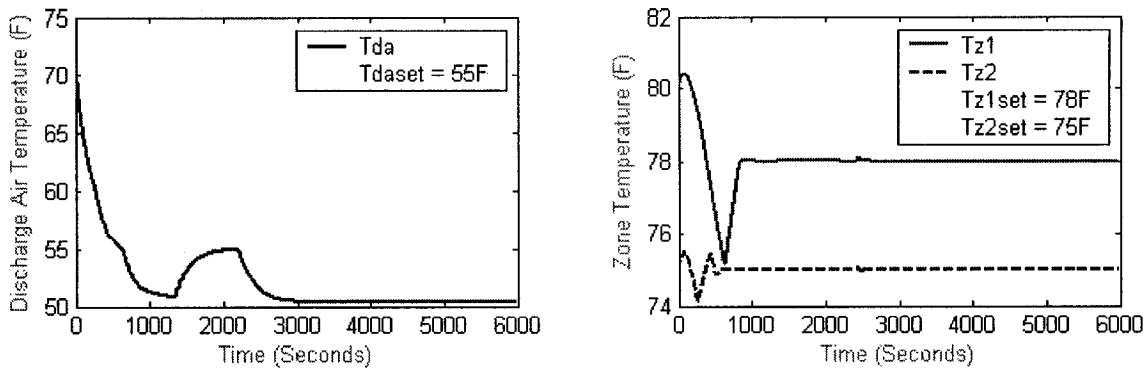


Figure 6-2 Response of temperatures

From the two tests, it is found that zone temperatures will not necessarily be affected when the cooling coil valve is stuck. The fact that the discharge air temperature cannot be reached is the primary symptom of this fault. Accordingly, the fault – *cooling coil valve stuck* – can be confirmed due to satisfaction of *Rules 3*, *4*, and *20*.

6.5.2 Case 2 – A Fouled Cooling Coil

Deposition of dirt and scale on a coil surface can greatly increase the resistance to heat transfer. As a result of the fact that fouling occurs gradually, this fault is hard to observe for some time. However, when the thermal resistance goes up to some degree, the cooling coil valve control signal will increase in order to increase the chilled water flow rate. Therefore, the required heat transfer will still be achieved to maintain the discharge air temperature at its setpoint. This thermal resistance may be represented by a fouling factor. A test was performed to simulate this fault, where a step change in the value was introduced to expedite the test. It was assumed that the fouling factor increased by 50

times. As shown in **Fig. 6-3**, the actual control signal of the cooling coil valve was 0.61 (U_{wact} = 0.61) and the expected value of the control signal was 0.52 (U_{wexp} = 0.52). The fact that the control signal to the cooling coil becomes larger than the expected value is the main feature of this fault. Accordingly, the fault – *a fouled cooling coil* – is diagnosed as the root cause due to satisfaction of *Rule 12*.

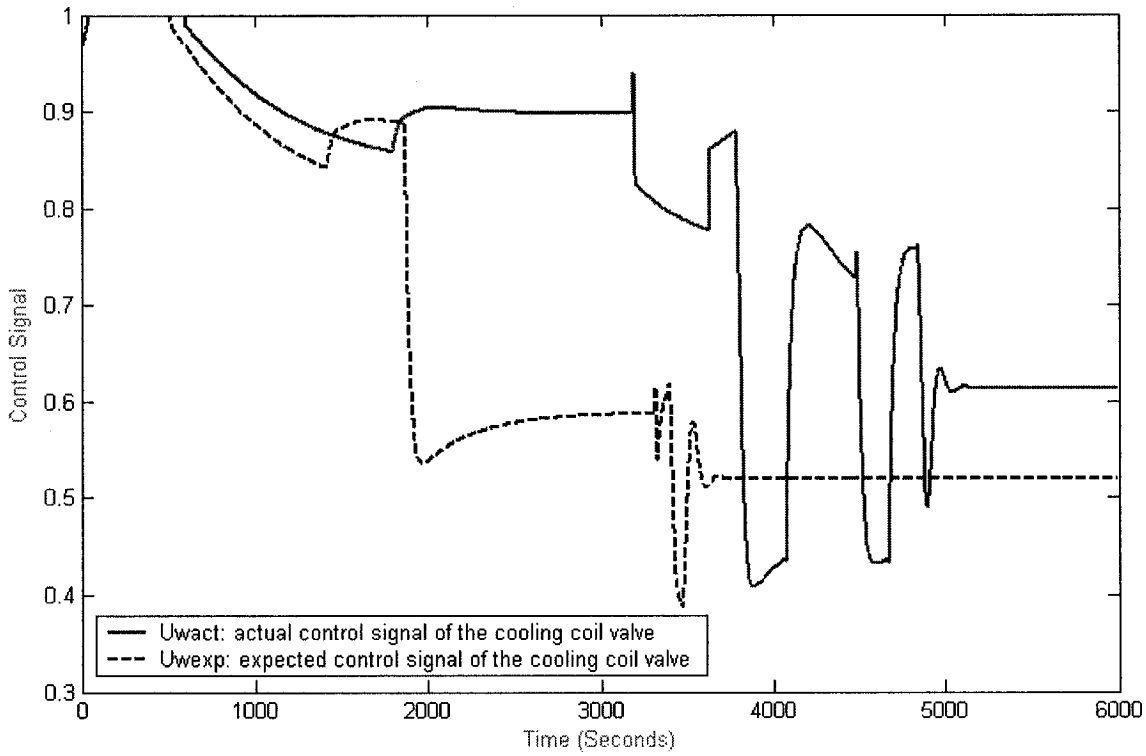


Figure 6-3 Control signal

6.5.3 Case 3 – A Stuck VAV Box Damper

This fault causes the zone temperature to drift away from the setpoint. Eventually, the control signal to VAV box damper will be saturating at one of its limits. The actual zone temperature will be either higher or lower than the setpoint value, depending on the

relation between the space load and the opening of VAV box damper. When the stuck damper delivers decreased airflow rate to the zone, the zone temperature will be higher than the setpoint. In this situation, VAV damper control signal will be saturating at its upper bound, i.e., 100% open. On the contrary, when the damper is stuck at such a position that excessive airflow rate is delivered to the zone, the zone temperature will be lower than the setpoint. In this situation, VAV damper control signal will be saturating at its lower bound, not 100% closed but a predetermined fraction to ensure the minimum airflow rate. A test was performed to simulate this fault, in which Zone 1 VAV damper was stuck at 60% open position at t = 1,701s, and Zone 2 VAV damper was stuck at 70% open position at t = 2,091s. As shown in **Fig.6-4**, Zone 1 was undercooled and Zone 2 was overcooled (i.e., T_{z1} = 84.1F and T_{z2} = 69.5F). That the zone temperature cannot be reached is the primary symptom of this fault. Accordingly, the fault – *supply air damper stuck* – is identified as the primary source of fault due to satisfaction of *Rules 9* and *10*.

6.5.4 Case 4 – An Undersized Cooling Coil

This fault cannot easily be seen in the case of partial load conditions because the cooling coil valve control signal can increase to compensate for the fault. It may be found when the system is experiencing design loads. When the cooling coil is undersized, the cooling capacity does not suffice for design conditions so that the discharge air temperature will be higher than the setpoint value when the steady state is reached. Also, zone temperature will be driven to a value well above the corresponding setpoint. A test was used to

simulate the fault. As shown in **Fig. 6-5**, the discharge air temperature was 64F (i.e., T_{da} = 64F). Moreover, the temperatures of both zones (T_{z1} = 86F and T_{z2} = 78.3F) were greater than their respective setpoints. It is noted that both discharge air temperature and zone temperature are higher than their associated setpoint values is the primary feature of this fault. Accordingly, *Rules 3* and 9 are satisfied confirming the fault – *an undersized cooling coil.*

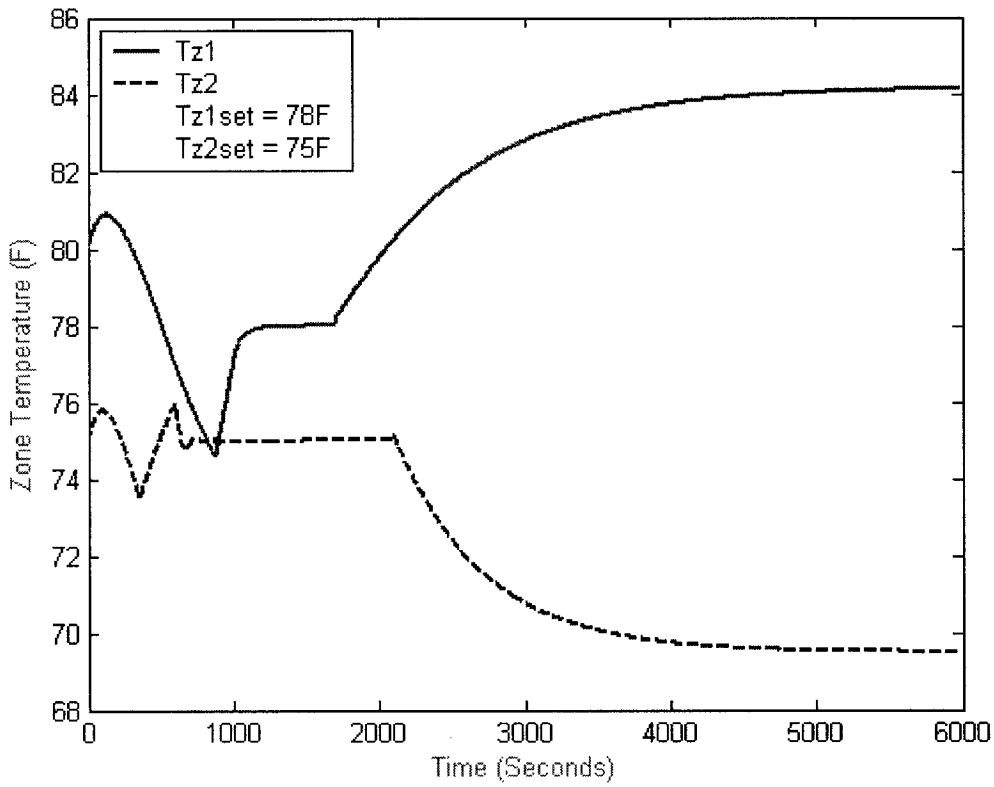


Figure 6-4 Response of zone temperature

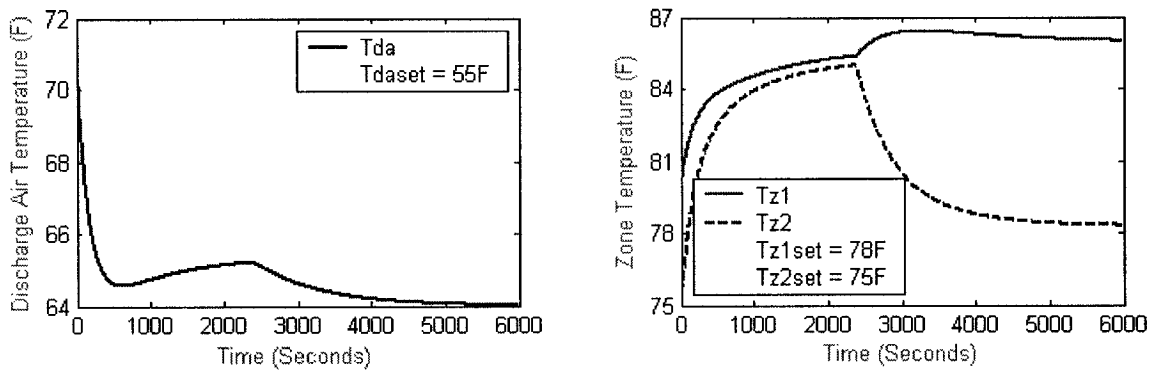


Figure 6-5 Response of temperatures

6.5.5 Case 5 – A Stuck Reheat Coil Valve

Unlike the above cases each of which has only one primary symptom, this fault presents two different features in two different operating conditions. When reheat is required, air temperature at the exit of the reheat coil will be fluctuating since the reheat coil is controlled using two-position control method. However, when the reheat coil valve is stuck, the discharge air temperature will eventually reach a constant temperature. If the valve is stuck closed, the constant temperature will be equal to discharge air temperature. If it is stuck open, this constant temperature will be higher than the discharge air temperature. On the other hand, when no reheat is required, the reheat coil valve is supposed to be closed. If the reheat coil valve is stuck open, hot water will be circulating inside the reheat coil, heating the entering air. As a result, supply air temperature will reach a value that is well above discharge air temperature.

Two tests were performed to simulate this fault in terms of the above situations. Test 1 was simulated by causing Zone 2 reheat coil valve to stick at 50% open position at t = 1,608s when reheat was needed. As shown in **Fig. 6-6**, the temperature of air delivered to Zone 2 stayed at 61F (i.e., T_{sa2} = 61F), not fluctuating while the temperature of air to Zone 1 was equal to discharge air temperature. Test 2 was simulated by causing Zone 2 reheat coil valve to stick at 80% open position at the beginning of the test when no reheat was required. As shown in **Fig. 6-7**, the temperature of air to Zone 2 was 59.7F (i.e., T_{sa2} = 59.7F), much higher than discharge air temperature. It should be noted that in both tests, zone temperature setpoints were still reached because VAV box damper control signal increased to compensate for the fault in terms of lower space loads acting on zones. It is concluded that the fault – *reheat coil valve stuck* – has occurred as per *Rule 8* when reheat is required and *Rule 6* when no reheat is required.

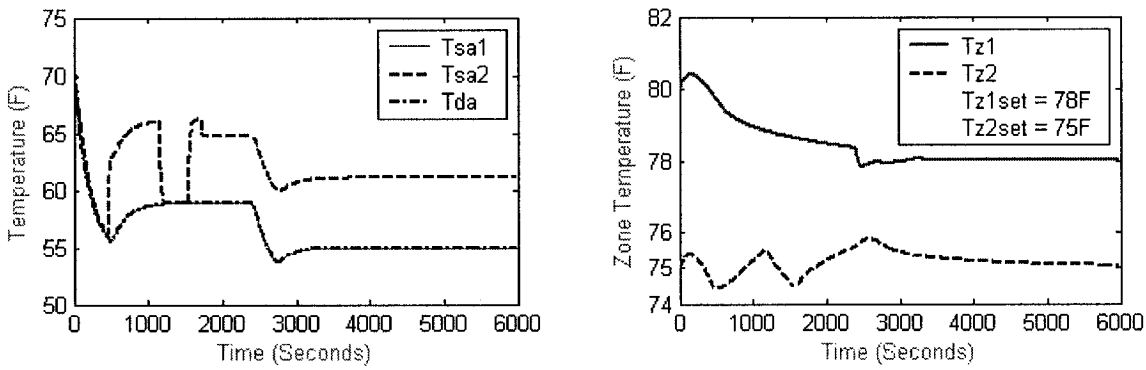


Figure 6-6 Response of temperatures

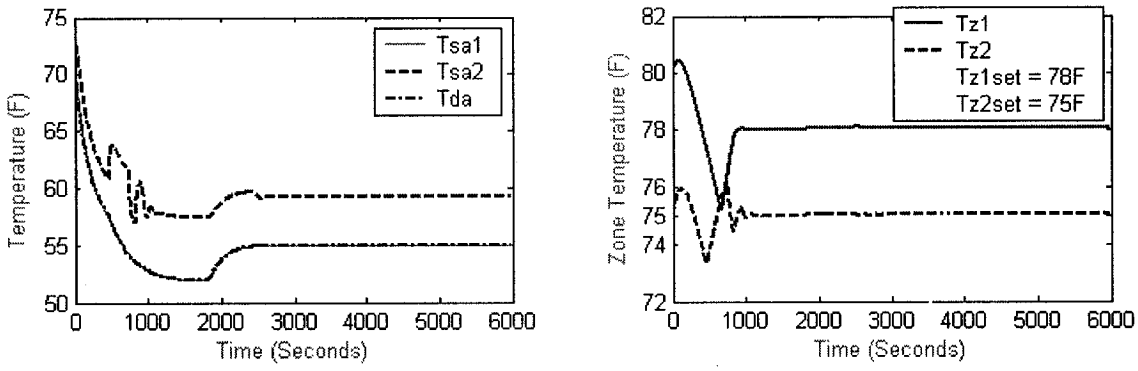


Figure 6-7 Response of temperatures

6.5.6 Case 6 – A Stuck Outdoor Air Damper

This fault belongs to economizer cycle control fault. It occurs due to a failure of a linkage in the mixing box dampers, which include outdoor air damper, recirculated air damper, and exhaust air damper. It presents three different symptoms under three different AHU operation modes. When either Mode#1 or Mode#3 is in effect, this fault will affect the airflow rates in the mixing box, and therefore, the actual mixed air temperature will be different from the expected value. In other words, for Mode#1, if the fault causes deficient outdoor air to be drawn into the HVAC system, the mixed air temperature will be decreasing, causing ratio of outdoor air intake to supply airflow rate to go down. On the contrary, if the fault causes excessive outdoor air to be drawn into the HVAC system, the mixed air temperature will be rising, causing ratio of outdoor air intake to supply airflow rate to go up. In these situations, discharge air temperature setpoint may still be reached. However, the consequence is that indoor air quality deteriorates because of insufficient fresh air in the first situation and more cooling energy is consumed in the second. For

Mode#3, if the fault causes excessive fresh air into the HVAC system, the mixed air temperature will be lower than the discharge air temperature setpoint. On the contrary, if fresh airflow rate decreases, the mixed air temperature will be greater than the setpoint. When Mode#2 takes effect, this fault will drive zone temperature away from the setpoint.

Three tests were performed to simulate the fault corresponding to the AHU operation modes. Test 1 was for Mode#1 and simulated by causing the outdoor air damper to stick at 8% open position at t = 1,266s. As shown in **Fig. 6-8**, the actual ratio of outdoor air intake to supply airflow rate was 0.14 (i.e., $R_{osact} = 0.14$), which was less than the expected ratio (i.e., $R_{osexp} = 0.18$). Test 2 was for Mode#2 and simulated by causing the outdoor air damper to stick at 60% open position at t = 1,230s. As shown in **Fig. 6-9**, Zone 1 temperature was 84F (i.e., $T_{z1} = 84F$). Test 3 was for Mode#3 and simulated by causing the outdoor air damper to stick at 30% open position at t = 1,413s. As shown in **Fig. 6-10**, the discharge air temperature was 57.7F (i.e., $T_{da} = 57.7F$), being greater than the setpoint as a result of the fact that deficient outdoor air was drawn into the system. From the above tests, it is concluded that the fault – *outdoor air damper stuck* – may be detected due to satisfaction of *Rule 14* for Mode#1, *Rule 9* for Mode#2, and *Rule 3* for Mode#3.

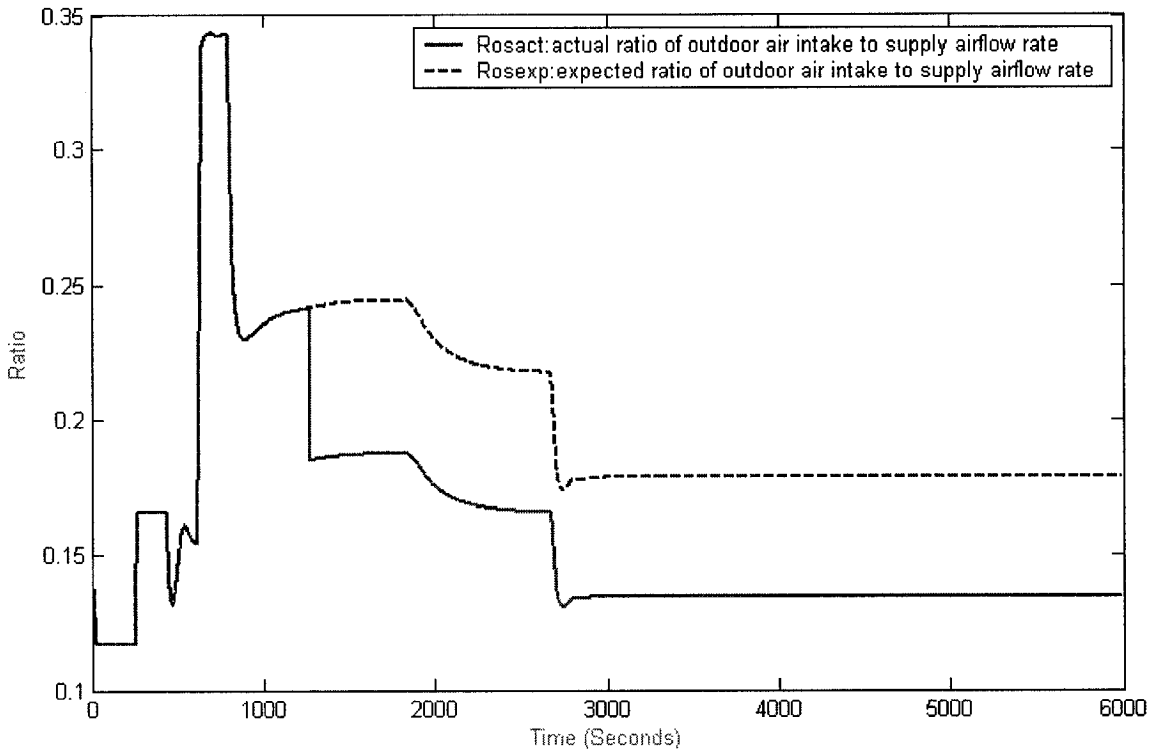


Figure 6-8 Ratio of outdoor air intake to supply airflow rate

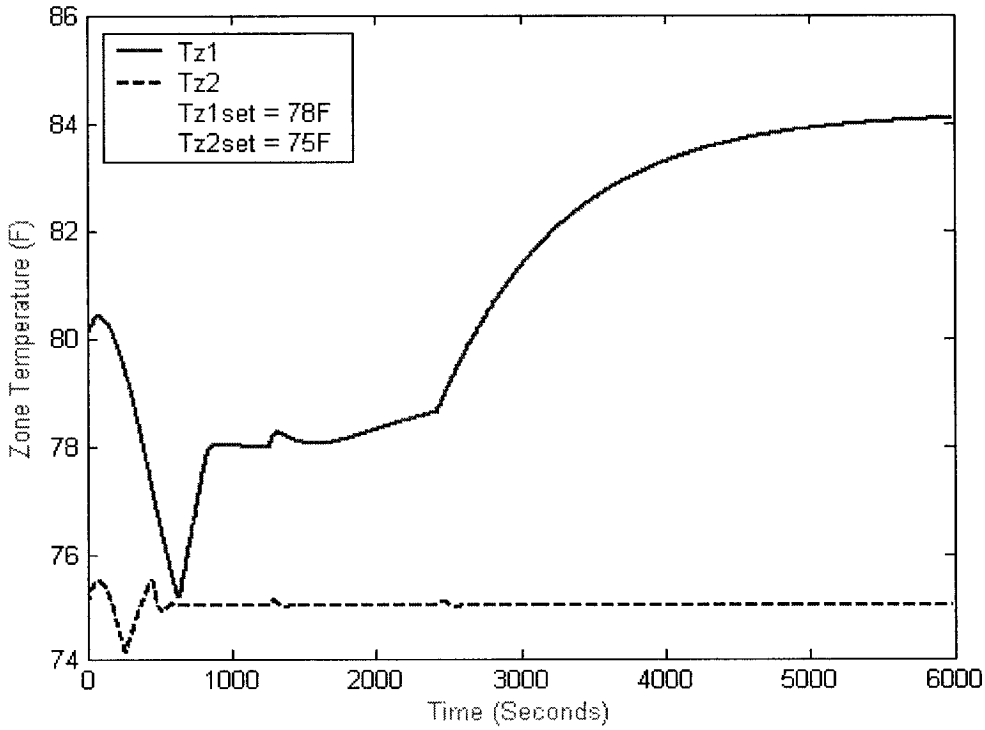


Figure 6-9 Response of zone temperature

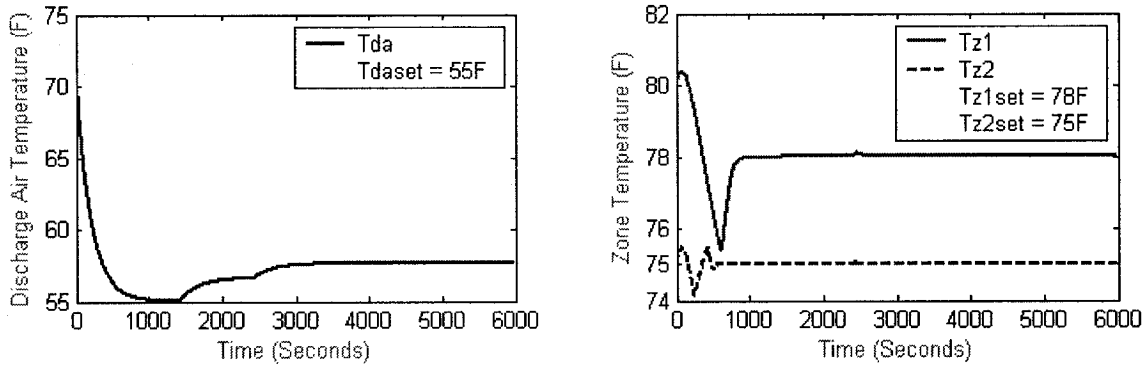


Figure 6-10 Response of temperatures

6.6 Summary and Conclusions

An expert rule set has been established to detect and diagnose faults on-line in the HVAC system. Tests have proven that the rules are effective in on-line detecting and diagnosing faults. It is found that most of HVAC faults may be detected by the abnormality of discharge air and zone air temperatures. Accordingly, the validity of outputs of the sensors is extremely important. If one or more sensors do not work, the outcome of the rules will be less reliable. Therefore, it is recommended that a pretest be performed to ensure that all the sensors are working well. Several rules can be used to detect sensor faults. The on-line FDD tool developed in this thesis is correctly identifying the faulty components or processes. It would be of interest to validate these algorithms on a real HVAC system.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Summary and Conclusions

In this thesis, a real-time monitoring, control, and fault detection system useful for on-line fault detection and diagnosis (FDD) of two-zone variable-air-volume terminal reheat (VAV-TRH) HVAC system is developed. This system consists of two control strategies - an optimal control strategy and a reheat control strategy, and a set of expert rules. Detailed simulation runs are made to compare these two control strategies. Also, real-time simulations are performed to test and analyze expert rules. A number of specific conclusions are presented at the end of the associated chapters. In this chapter, most important tasks and conclusions are summarized.

1. A software program to perform steady state simulations for a multi-zone (up to five) variable-air-volume terminal reheat (VAV-TRH) HVAC system was developed.
2. A dynamic model for a two-zone variable-air-volume terminal reheat (VAV-TRH) HVAC system was established.
3. The software was implemented to perform off-line and on-line simulations for the model.
4. Two control strategies of HVAC operation - optimal control strategy and reheat control strategy - were designed.

5. It was found that optimal control strategy could save up to 7 percent energy in HVAC systems in relative to reheat control strategy and that optimal control strategy was feasible most of the operation time.
6. It was noted that optimal control strategy would not be feasible if building loads became much less than their corresponding peak loads, and in this situation, reheat control strategy would be the proper choice.
7. A set of expert rules for fault detection and diagnosis were adapted from the literature and were enhanced to develop a real-time FDD tool for a single-duct variable-air-volume terminal reheat (VAV-TRH) HVAC system.
8. Results from the application of expert rules to the data from real-time simulations showed that the rules were efficient and the FDD program correctly identified the faults in real-time.

The major contributions of this study compared to other similar research projects reported in the literature are in the following three aspects:

i. HVAC fault detection and diagnosis was studied together with HVAC operation and control strategies.
ii. A more complete set of expert rules for on-line HVAC fault detection and diagnosis was defined for the chosen model.
iii. Expert rules were tested based on real-time simulation data for the modeled HVAC system rather than off-line simulation data. The real-time data function exactly the

same as field data as long as the model predictions are relatively accurate. Otherwise, the model needs to be updated or adapted periodically.

7.2 Recommendations for Future Work

Centered on on-line HVAC fault detection and diagnosis (FDD), lots of studies can be and should be explored in the future. First, in this thesis, the same value of ε_i (error or residual) was used in all IF-THEN rules. How to determine the exact value for each fault is necessary. A conservative value tends to degrade sensitivity while a non-conservative value causes fault alarms. A possible solution to this problem is to determine the error associated with each temperature senor measurement and then combine the error to produce the most appropriate representation of the rule. Such an approach will produce a compromise between sensitivity to faults and robustness against false alarms. Second, an expert system for HVAC FDD should be developed using the expert rules and be applied in the real HVAC system, at least it should be tested by data from real HVAC systems. This system should isolate the cause of problems and direct the HVAC operator to the proper corrective actions. For the best case, it should also highlight the energy and cost impacts due to improper operations or faults. Finally, for some special applications like computer rooms, humidity control is a must. Hence, fault detection and diagnosis related to this aspect should also be studied.

In general, performing FDD of a complex HVAC system is an extremely difficult task. Many refinements in the proposed approach are needed and numerous studies are also required. All these endeavors are worthwhile because FDD can help improve building operation; thus bringing improved comfort and air quality, longer equipment life, and lower costs.

References

1. Anderson, D., L. Graves, W. Reinert, J.F. Kreider, J. Dow□and H. Wubbens 1989 A quasi-real-time expert system for commercial building HVAC diagnostics. *ASHRAE Transactions* 95 (2): 954-960
2. ASHRAE Handbook, 1997. Fundamentals. Atlanta: American Society of Heating, Refrigerating, and Air-Conditioning Engineers.
3. ASHRAE Handbook, 2004. HVAC Systems and Equipment. Atlanta: American Society of Heating, Refrigerating, and Air-Conditioning Engineers.
4. Barr, A., and E. Feigenbaum. 1982 "The handbook of artificial intelligence." William Kaufmann Inc., Los Altos, California
5. Braun, J.E., S.A. Klein, J.W. Mitchell, and W.A. Beckman. 1989a Application of optimal control to chilled water system without storage. *ASHRAE Transactions* 95 (1): 587-600
6. Braun, J.E., S.A. Klein, J.W. Mitchell, and W.A. Beckman. 1989b Methodologies for optimal control of chilled water systems without storage. ASHRAE Winter Annual Meeting, Chicago 95
7. Braun, J.E. 1990 Reducing energy costs and peak electrical demand through optimal control of building thermal storage. *ASHRAE Transactions* 96 (2): 867-888
8. Clark, D.R. 1985 "HVACSIM^{+} building systems and equipment simulation reference manual." Technical Report NBSIR 84-2996 U.S. Department of Commerce, National Bureau of Standards, USA
9. Cooling, J. 2003 "Software engineering for real-time systems." Addison-Wesley Publishing Company, New York
10. Cumali, Z. 1988 Global optimization of HVAC system operations in real time. *ASHRAE Transactions* 94 (1): 1729-1744
11. Dodier, R.H., P.S. Curtiss, and J.F. Kreider. 1998 Small-scale on-line diagnostics for

an HVAC system. *ASHRAE Transactions* 104 (1A): 530-540

12. Grimmelius, H.T., J.K. Woud, and G.. Bean. 1995 On-line failure diagnosis for compression refrigeration plants. *International Journal of Refrigeration* 18: 34-41
13. Haberl, J.S., and D.E. Claridge. 1987 An expert system for building energy consumption analysis: prototype results. *ASHRAE Transactions* 93 (1): 979-986
14. Han, C.Y., Y.F. Xiao, and C.J. Ruther. 1999 Fault detection and diagnosis of HVAC system. *ASHRAE Transactions* 105 (1): 568-578
15. House, J.M., T.F. Smith, and J.S. Arora. 1991 Optimal control of a thermal system. *ASHRAE Transactions* 97 (1): 991-1001
16. House, J.M., and T.F. Smith. 1995 A system approach to optimal control for HVAC and building systems. *ASHRAE Transactions* 101 (1): 647-660
17. House, J.M., and D.R. Shin. 1999 Classification techniques for fault detection and diagnosis of an air-handling unit. *ASHRAE Transactions* 105 (1): 1087-1097
18. House, J.M., H. Vaezi-Nejad, and J.M. Whitcomb. 2002 An expert rule set for fault detection in air-handling units. NIST Report
19. Huang, W.Z. 2003 Dynamic simulation of energy management control functions in VAV-HVAC systems. Master thesis, Department of Building, Civil and Environmental Engineering, Concordia University, Montreal, Canada
20. Jacobson, I., M. Christerson, P. Jonsson, and G.. Overgaard. 1992 "Object-Oriented Software Engineering - A Use Case Driven Approach." Addison-Wesley Publishing Company, Wokingham, England
21. Jiang, Y, J. Li, and X. Yang. 1995 Fault direction space method for on-line fault detection. *ASHRAE Transactions* 101 (2): 219-228
22. Karkekar, B.V., and R.M. Desmond. 1982 "Heat transfer." 2nd edition. West Publishing CO.
23. Katipamula, S., R.G. Pratt, and D.P. Chassin 1999 Automated fault detection and diagnostics for outdoor-air ventilation systems and economizers: methodology and

results from field testing. *ASHRAE Transactions* 105 (1): 555-567

24. Kaya, A., C.S. Chen, S. Raina, and S.J. Alexander. 1982 Optimal control policies to minimize energy use in HVAC system. *ASHRAE Transactions* 88 (1): 235-248
25. Lee, W.Y., J.M. House, and D.R. Shin. 1997 Fault diagnosis and temperature sensor recovery for an air-handling unit. *ASHRAE Transactions* 103 (1): 621-633
26. Levenhagen, J.I., and D. H. Spethmann. 1993 "VAV controls and systems." New York: McGraw-Hill.
27. Li, H., C. Ganesh, and D.R. Munoz 1996 Optimal control of duct pressure in HVAC systems. *ASHRAE Transactions* 102 (2): 170-174
28. Li, X., J.C. Visier, and H. Vaezi-Nejad. 1997 A neural network prototype for fault detection and diagnosis of heating systems. *ASHRAE Transactions* 103 (1): 634-643
29. MATLAB Manual (Release 13SP1) 1997 The MathWorks, Inc. MA. USA
30. McQuiston, F.C., J.D. Parker, and J.D. Spitler. 2000 "Heating, ventilating, and air conditioning: analysis and design." 5th edition. John Wiley & Sons. Inc.
31. Nizet, J.L., J. Lecomte, and F.X. Liu, 1984 Optimal control applied to air conditioning in buildings. *ASHRAE Transactions* 90 (1b): 587-600
32. Norford, L. K., and A. Rabl 1987 Energy management systems as diagnostic tools for building managers and energy auditors. *ASHRAE Transactions* 93 (2): 2360-2365
33. Norford, L. K and R. D. Little 1993 Fault detection and load monitoring in ventilation systems. *ASHRAE Transactions* 99 (1): 590-602
34. Ormandjieva, O. 2003 Class notes for COMP354 (Software Engineering) Department of Computer Science, Concordia University, Montreal, Canada
35. Pape, F., J.W. Mitchell, and W.A. Beckman. 1991 Optimal control and fault detection in heating, ventilating, and air-conditioning systsm. *ASHRAE Transactions* 97 (1): 729-735
36. Parken, W.H., J. K. Kao, and G.E. Kelly. 1982. Strategies for energy conservation in small office buildings. NBSIR82-2489. Gaithersburg, MD: National Bureau of

Standards

37. Peitsman, H.C., and V.E. Bakker. 1996 Application of black-box models to HVAC systems for fault detection. *ASHRAE Transactions* 102 (1): 628-640
38. Peitsman, H.C., and L. L. Soethout. 1997 ARX models and real-time model-based diagnosis. *ASHRAE Transactions* 103 (1): 657-670
39. Porges. F. 2001 "HVAC Engineer's Handbook." 11th ed. Boston: Butterworth-Heinemann
40. So, A., W.L. Chan, and T.T. Chow. 1995 A computer-vision-based HVAC control system. *ASHRAE Transactions* 101 (2): 661-677
41. Stylianou, M., and D. Nikanpour. 1996 Performance monitoring, fault detection, and diagnosis of reciprocating chillers. *ASHRAE Transactions* 102 (1): 615-627
42. Stylianou, M. 1997 Application of classification functions to chiller fault detection and diagnosis. *ASHRAE Transactions* 103 (1): 645-654
43. Visier, J.C., H. Vaezi-Nejad, and P. Corraies., 1999 A fault detection tool for school buildings. ASHRAE Transactions 105 (1): 543-554
44. Wulfinghoff, D.R. 1999 "Energy efficiency manual." Maryland: Energy Institute Press
45. Zhang, Z., and R.M. Nelson. 1992 Parametric analysis of a building space conditioned by a VAV system. *ASHRAE Transactions* 98 (1): 43-48

APPENDIX A

VAV SIMULATOR

//Developed in partial fulfillment of the requirements for the degree of Master of
//Applied Science at Concordia University
//A Rea-time Simulation Tool for Fault Detection and Diagnosis of HVAC Systems
//Author: Yue MA
//June 2004

```
#if !defined(AFX_GENERALINFOINPUTDLG_H__26D74CE3_9024_440E_AB32_A3
EC88D67B20__INCLUDED_)
#define
AFX_GENERALINFOINPUTDLG_H__26D74CE3_9024_440E_AB32_A3EC88D67B
20__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// GeneralInfoInputDlg.h : header file

/////////////////////////////////////////////////////////////////////////
// CGeneralInfoInputDlg dialog

class CGeneralInfoInputDlg : public CDialog
{
// Construction
public:
    CGeneralInfoInputDlg(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
    //{{AFX_DATA(CGeneralInfoInputDlg)
    enum { IDD = IDD_GENERALINFOINPUT };
    double  m_EnterChilledWaterTemp;
    double  m_FanEfficiency;
    double  m_FanPressureRise;
    int     m_NumOfZones;
    double  m_OutsideAirFlowRate;
    double  m_OutsideRelHumidity;
    double  m_OutsideTemp;
    double  m_RelHumidityOfLeavingAir;
```

```
    double  m_TempOfSurroundingAir;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CGeneralInfoInputDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);      // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CGeneralInfoInputDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnButton1();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif
//  !defined(AFX_GENERALINFOINPUTDLG_H__26D74CE3_9024_440E_AB32_A3
EC88D67B20__INCLUDED_)

**********************************************************************
// GeneralInfoInputDlg.cpp : implementation file

#include "stdafx.h"
#include "MultiVAV5.h"
#include "GeneralInfoInputDlg.h"
#include "MultiVAV5Doc.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
```

```
static char THIS_FILE[] = __FILE__;
#endif

// CGeneralInfoInputDlg dialog

CGeneralInfoInputDlg::CGeneralInfoInputDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CGeneralInfoInputDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CGeneralInfoInputDlg)
    m_EnterChilledWaterTemp = 0.0;
    m_FanEfficiency = 0.0;
    m_FanPressureRise = 0.0;
    m_NumOfZones = 0;
    m_OutsideAirFlowRate = 0.0;
    m_OutsideRelHumidity = 0.0;
    m_OutsideTemp = 0.0;
    m_RelHumidityOfLeavingAir = 0.0;
    m_TempOfSurroundingAir = 0.0;
    //}}AFX_DATA_INIT
}

void CGeneralInfoInputDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CGeneralInfoInputDlg)
    DDX_Text(pDX, IDC_ENTERCHILLEDWATERTEMP,
m_EnterChilledWaterTemp);
    DDX_Text(pDX, IDC_FANEFFICIENCY, m_FanEfficiency);
    DDX_Text(pDX, IDC_FANPRESSURERISE, m_FanPressureRise);
    DDX_Text(pDX, IDC_NUMOFZONES, m_NumOfZones);
    DDX_Text(pDX, IDC_OUTSIDEAIRFLOWRATE, m_OutsideAirFlowRate);
    DDX_Text(pDX, IDC_OUTSIDERELHUMIDITY, m_OutsideRelHumidity);
    DDX_Text(pDX, IDC_OUTSIDETEMP, m_OutsideTemp);
    DDX_Text(pDX, IDC_RELHUMIDITYOFLEAVINGAIR,
m_RelHumidityOfLeavingAir);
    DDX_Text(pDX, IDC_TEMPOFSURROUNDINGAIR,
m_TempOfSurroundingAir);
    //}}AFX_DATA_MAP
}
```

```
BEGIN_MESSAGE_MAP(CGeneralInfoInputDlg, CDialog)
    //{{AFX_MSG_MAP(CGeneralInfoInputDlg)
    ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CGeneralInfoInputDlg message handlers

BOOL CGeneralInfoInputDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    m_EnterChilledWaterTemp = 7;
    m_FanEfficiency = 0;
    m_FanPressureRise = 0;
    m_OutsideAirFlowRate = 0;
    m_OutsideTemp = 30;
    m_OutsideRelHumidity = 56;
    m_RelHumidityOfLeavingAir = 95;
    m_TempOfSurroundingAir = 28;
    m_NumOfZones = 1;

    UpdateData(FALSE);
    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}

void CGeneralInfoInputDlg::OnButton1()
{
    UpdateData( );

    if( ( m_OutsideTemp < 20 ) || ( m_OutsideRelHumidity <= 0 )
        || ( m_OutsideAirFlowRate < 0 ) || ( m_FanPressureRise <= 0 )
        || ( m_FanEfficiency <= 0 ) || ( m_TempOfSurroundingAir <= 10 )
        || ( m_NumOfZones < 1 ) || ( m_RelHumidityOfLeavingAir < 50 ) )
        MessageBox ("Invalid inputs.");
    else
    {
        CMultiVAV5Doc*pDoc1
=(CMultiVAV5Doc*)((CMainFrame*)AfxGetMainWnd())->GetActiveDocument();
```

```
        pDoc1 -> EChilledWaterTemp = m_EnterChilledWaterTemp;
        pDoc1 -> FEfficiency = m_FanEfficiency;
        pDoc1 -> FPressureRise = m_FanPressureRise;
        pDoc1 -> OAirFlowRate = m_OutsideAirFlowRate;
        pDoc1 -> OTemp = m_OutsideTemp;
        pDoc1 -> ORelHumidity = m_OutsideRelHumidity;
        pDoc1 -> RHumidityOfLeavingAir = m_RelHumidityOfLeavingAir;
        pDoc1 -> TSurroundingAir = m_TempOfSurroundingAir;
        pDoc1 -> NZones = m_NumOfZones;
    }
}

**********************************************************************
#if !defined(AFX_ZONE1INPUTDLG_H__AE16A4D9_F3CC_47C2_A3F8_45ADDD7
C9D58__INCLUDED_)
#define
AFX_ZONE1INPUTDLG_H__AE16A4D9_F3CC_47C2_A3F8_45ADDD7C9D58__IN
CLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// Zone1InputDlg.h : header file
// CZone1InputDlg dialog

class CZone1InputDlg : public CDialog
{
// Construction
public:
    CZone1InputDlg(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
    //{{AFX_DATA(CZone1InputDlg)
    enum { IDD = IDD_ZONE1INPUT };
    double  m_DesignSensibleLoads1;
    double  m_DesignTotalLoads1;
    int     m_DiameterOfSupplyDuct1;
    int     m_DiameterOfReturnDuct1;
    int     m_HeightOfReturnDuct1;
    int     m_HeightOfSupplyDuct1;
```

```
    double  m_LengthOfReturnDuct1;
    double  m_LengthOfSupplyDuct1;
    double  m_MinPercentageOfAirFlow1;
    int     m_ReturnDuctShape1;
    double  m_SensibleLoads1;
    int     m_SupplyDuctShape1;
    double  m_TotalLoads1;
    int     m_WidthOfReturnDuct1;
    int     m_WidthOfSupplyDuct1;
    double  m_ZoneRelHumidity1;
    double  m_ZoneTemp1;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CZone1InputDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);      // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CZone1InputDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnButton2();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif
//  !defined(AFX_ZONE1INPUTDLG_H__AE16A4D9_F3CC_47C2_A3F8_45ADDD7
C9D58__INCLUDED_)

**********************************************************************
```

// Zone1InputDlg.cpp : implementation file

```
#include "stdafx.h"
#include "MultiVAV5.h"
#include "Zone1InputDlg.h"
#include "MultiVAV5Doc.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CZone1InputDlg dialog


CZone1InputDlg::CZone1InputDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CZone1InputDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CZone1InputDlg)
    m_DesignSensibleLoads1 = 0.0;
    m_DesignTotalLoads1 = 0.0;
    m_DiameterOfSupplyDuct1 = 0;
    m_DiameterOfReturnDuct1 = 0;
    m_HeightOfReturnDuct1 = 0;
    m_HeightOfSupplyDuct1 = 0;
    m_LengthOfReturnDuct1 = 0.0;
    m_LengthOfSupplyDuct1 = 0.0;
    m_MinPercentageOfAirFlow1 = 0.0;
    m_ReturnDuctShape1 = 0;
    m_SensibleLoads1 = 0.0;
    m_SupplyDuctShape1 = 0;
    m_TotalLoads1 = 0.0;
    m_WidthOfReturnDuct1 = 0;
    m_WidthOfSupplyDuct1 = 0;
    m_ZoneRelHumidity1 = 0.0;
    m_ZoneTemp1 = 0.0;
    //}}AFX_DATA_INIT
}
```

```
void CZone1InputDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CZone1InputDlg)
    DDX_Text(pDX, IDC_DESIGNSENSIBLELOADS1, m_DesignSensibleLoads1);
    DDX_Text(pDX, IDC_DESIGNTOTALLOADS1, m_DesignTotalLoads1);
    DDX_Text(pDX, IDC_DIAMETEROFSUPPLYDUCT1,
m_DiameterOfSupplyDuct1);
    DDX_Text(pDX, IDC_DIAMETEROFRETURNDUCT1,
m_DiameterOfReturnDuct1);
    DDX_Text(pDX, IDC_HEIGHTOFRETURNDUCT1, m_HeightOfReturnDuct1);
    DDX_Text(pDX, IDC_HEIGHTOFSUPPLYDUCT1, m_HeightOfSupplyDuct1);
    DDX_Text(pDX, IDC_LENGTHOFRETURNDUCT1, m_LengthOfReturnDuct1);
    DDX_Text(pDX, IDC_LENGTHOFSUPPLYDUCT1, m_LengthOfSupplyDuct1);
    DDX_Text(pDX, IDC_MINPERCENTAGEOFAIRFLOW1,
m_MinPercentageOfAirFlow1);
    DDX_Text(pDX, IDC_RETURNDUCTSHAPE1, m_ReturnDuctShape1);
    DDX_Text(pDX, IDC_SENSIBLELOADS1, m_SensibleLoads1);
    DDX_Text(pDX, IDC_SUPPLYDUCTSHAPE1, m_SupplyDuctShape1);
    DDX_Text(pDX, IDC_TOTALLOADS1, m_TotalLoads1);
    DDX_Text(pDX, IDC_WIDTHOFRETURNDUCT1, m_WidthOfReturnDuct1);
    DDX_Text(pDX, IDC_WIDTHOFSUPPLYDUCT1, m_WidthOfSupplyDuct1);
    DDX_Text(pDX, IDC_ZONERELHUMIDITY1, m_ZoneRelHumidity1);
    DDX_Text(pDX, IDC_ZONETEMP1, m_ZoneTemp1);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CZone1InputDlg, CDialog)
    //{{AFX_MSG_MAP(CZone1InputDlg)
    ON_BN_CLICKED(IDC_BUTTON2, OnButton2)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CZone1InputDlg message handlers

BOOL CZone1InputDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    m_DesignSensibleLoads1 = 0;
```

```
    m_DesignTotalLoads1 = 0;
    m_DiameterOfReturnDuct1 = 0;
    m_DiameterOfSupplyDuct1 = 0;
    m_HeightOfReturnDuct1 = 0;
    m_HeightOfSupplyDuct1 = 0;
    m_LengthOfReturnDuct1 = 0;
    m_LengthOfSupplyDuct1 = 0;
    m_ReturnDuctShape1 = 1;
    m_SupplyDuctShape1 = 1;
    m_SensibleLoads1 = 0;
    m_TotalLoads1 = 0;
    m_WidthOfReturnDuct1 = 0;
    m_WidthOfSupplyDuct1 = 0;
    m_ZoneRelHumidity1 = 50;
    m_ZoneTemp1 = 24;
    m_MinPercentageOfAirFlow1 = 40;

    UpdateData(FALSE);
    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}

void CZone1InputDlg::OnButton2()
{
    UpdateData( );

    if ( ( m_ZoneTemp1 >= 16 ) && ( m_ZoneRelHumidity1 > 0 ) &&
( m_DesignSensibleLoads1 > 0 )
        && ( m_DesignTotalLoads1 > 0 ) && ( m_DesignTotalLoads1 >
m_DesignSensibleLoads1 )
        && ( ( m_SupplyDuctShape1 == 1 ) && ( m_HeightOfSupplyDuct1 > 0 ) &&
( m_WidthOfSupplyDuct1 > 0 ) )
        || ( ( m_SupplyDuctShape1 == 2 ) && ( m_DiameterOfSupplyDuct1 > 0 ) )
        && ( m_LengthOfSupplyDuct1 > 0 )
        && ( ( m_ReturnDuctShape1 == 1 ) && ( m_HeightOfReturnDuct1 > 0 ) &&
( m_WidthOfReturnDuct1 > 0 ) )
        || ( ( m_ReturnDuctShape1 == 2 ) && ( m_DiameterOfReturnDuct1 > 0 ) )
        && ( m_LengthOfReturnDuct1 > 0 )
        && ( m_MinPercentageOfAirFlow1 > 0) && ( m_SensibleLoads1 > 0 )
        && ( m_TotalLoads1 > 0 ) && ( m_TotalLoads1 > m_SensibleLoads1 )
        && ( m_SensibleLoads1 < m_DesignSensibleLoads1 )
```

```
        && ( m_TotalLoads1 < m_DesignTotalLoads1 ) )
    {
        CMultiVAV5Doc*pDoc2
=(CMultiVAV5Doc*)((CMainFrame*)AfxGetMainWnd())->GetActiveDocument();

        //Store the inputs to the document.
        pDoc2 -> DSensibleLoads1 = m_DesignSensibleLoads1;
        pDoc2 -> DTotalLoads1 = m_DesignTotalLoads1;
        pDoc2 -> DReturnDuct1 = m_DiameterOfReturnDuct1;
        pDoc2 -> DSupplyDuct1 = m_DiameterOfSupplyDuct1;
        pDoc2 -> HReturnDuct1 = m_HeightOfReturnDuct1;
        pDoc2 -> HSupplyDuct1 = m_HeightOfSupplyDuct1;
        pDoc2 -> LReturnDuct1 = m_LengthOfReturnDuct1;
        pDoc2 -> LSupplyDuct1 = m_LengthOfSupplyDuct1;
        pDoc2 -> RDuctShape1 = m_ReturnDuctShape1;
        pDoc2 -> SDuctShape1 = m_SupplyDuctShape1;
        pDoc2 -> SLoads1 = m_SensibleLoads1;
        pDoc2 -> TLoads1 = m_TotalLoads1;
        pDoc2 -> WReturnDuct1 = m_WidthOfReturnDuct1;
        pDoc2 -> WSupplyDuct1 = m_WidthOfSupplyDuct1;
        pDoc2 -> ZRelHumidity1 = m_ZoneRelHumidity1;
        pDoc2 -> ZTemp1 = m_ZoneTemp1;
        pDoc2 -> MPercentageOfAirFlow1 = m_MinPercentageOfAirFlow1;
    }
    else
        MessageBox ("Invalid inputs.");
}


**********************************************************************
#if !defined(AFX_GENERALRESULTSOUTPUTDLG_H__2CF901B6_022A_4403_B5
FF_0451206B2F36__INCLUDED_)
#define
AFX_GENERALRESULTSOUTPUTDLG_H__2CF901B6_022A_4403_B5FF_045120
6B2F36__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// GeneralResultsOutputDlg.h : header file
```

```
/////////////////////////////////////////////////////////////////////////////
// CGeneralResultsOutputDlg dialog

class CGeneralResultsOutputDlg : public CDialog
{
// Construction
public:
    CGeneralResultsOutputDlg(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
    //{{AFX_DATA(CGeneralResultsOutputDlg)
    enum { IDD = IDD_GENERALRESULTSOUTPUT };
    CString m_CoilVolumeFlowRate;
    CString m_CoolingCapacity;
    CString m_ExhaustAirFlowRate;
    CString m_LeavingAirTemp;
    CString m_RelHumidityOfLeavingAir;
    CString m_ReturnAirFlowRate;
    CString m_ReturnAirTemp;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CGeneralResultsOutputDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CGeneralResultsOutputDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnButton7();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
```

previous line.

```
#endif
// !defined(AFX_GENERALRESULTSOUTPUTDLG_H__2CF901B6_022A_4403_B5F
F_0451206B2F36__INCLUDED_)

/////////////////////////////////////////////////////////////////////////////
// GeneralResultsOutputDlg.cpp : implementation file

#include "stdafx.h"
#include "MultiVAV5.h"
#include "GeneralResultsOutputDlg.h"
#include "moistair.h"
#include "MultiVAV5Doc.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CGeneralResultsOutputDlg dialog

CGeneralResultsOutputDlg::CGeneralResultsOutputDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CGeneralResultsOutputDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CGeneralResultsOutputDlg)
    m_CoilVolumeFlowRate = _T("");
    m_CoolingCapacity = _T("");
    m_ExhaustAirFlowRate = _T("");
    m_LeavingAirTemp = _T("");
    m_RelHumidityOfLeavingAir = _T("");
    m_ReturnAirFlowRate = _T("");
    m_ReturnAirTemp = _T("");
    //}}AFX_DATA_INIT
}
```

```
void CGeneralResultsOutputDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CGeneralResultsOutputDlg)
    DDX_Text(pDX, IDC_COILVOLUMEFLOWRATE, m_CoilVolumeFlowRate);
    DDX_Text(pDX, IDC_COOLINGCAPACITY, m_CoolingCapacity);
    DDX_Text(pDX, IDC_EXHAUSTAIRFLOWRATE, m_ExhaustAirFlowRate);
    DDX_Text(pDX, IDC_LEAVINGAIRTEMP, m_LeavingAirTemp);
    DDX_Text(pDX,                          IDC_RELHUMIDITYOFLEAVINGAIR,
m_RelHumidityOfLeavingAir);
    DDX_Text(pDX, IDC_RETURNAIRFLOWRATE, m_ReturnAirFlowRate);
    DDX_Text(pDX, IDC_RETURNAIRTEMP, m_ReturnAirTemp);
    //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CGeneralResultsOutputDlg, CDialog)
    //{{AFX_MSG_MAP(CGeneralResultsOutputDlg)
    ON_BN_CLICKED(IDC_BUTTON7, OnButton7)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CGeneralResultsOutputDlg message handlers

BOOL CGeneralResultsOutputDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // TODO: Add extra initialization here
    m_CoilVolumeFlowRate = "0.0";
    m_CoolingCapacity = "0.0";
    m_ExhaustAirFlowRate = "0.0";
    m_LeavingAirTemp = "0.0";
    //m_Reheat = "0.0";
    m_RelHumidityOfLeavingAir = "0.0";
    m_ReturnAirFlowRate = "0.0";
    m_ReturnAirTemp = "0.0";
    //m_SupplyAirFlowRate = "0.0";
    //m_SupplyAirTemp = "0.0";
    //m_TempBeforeReheat = "0.0";
    UpdateData(FALSE);
```

```
    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}

void CGeneralResultsOutputDlg::OnButton7()
{
    // TODO: Add your control notification handler code here
    CMultiVAV5Doc*                                                    pDoc7
=(CMultiVAV5Doc*)((CMainFrame*)AfxGetMainWnd())->GetActiveDocument();
    //Take general inputs from the document.
    double RelHumidityOfLeavingAir = pDoc7 -> RHumidityOfLeavingAir;
    double EnterChilledWaterTemp = pDoc7 -> EChilledWaterTemp;
    double OutsideTemp = pDoc7 -> OTemp;
    double OutsideRelHumidity = pDoc7 -> ORelHumidity;
    double OutsideAirFlowRate = pDoc7 -> OAirFlowRate;
    double FanEfficiency = pDoc7 -> FEfficiency;
    double FanPressureRise = pDoc7 -> FPressureRise;
    double TempOfSurroundingAir = pDoc7 -> TSurroundingAir;
    int NumOfZones = pDoc7 -> NZones;

    //Take inputs of zone1 from the document.
    double ZoneTemp1 = pDoc7 -> ZTemp1;
    double ZoneRelHumidity1 = pDoc7 -> ZRelHumidity1;
    double DesignSensibleLoads1 = pDoc7 -> DSensibleLoads1;
    double DesignTotalLoads1 = pDoc7 -> DTotalLoads1;
    int SupplyDuctShape1 = pDoc7 -> SDuctShape1;
    int WidthOfSupplyDuct1 = pDoc7 -> WSupplyDuct1;
    int HeightOfSupplyDuct1 = pDoc7 -> HSupplyDuct1;
    double LengthOfSupplyDuct1 = pDoc7 -> LSupplyDuct1;
    int DiameterOfSupplyDuct1 = pDoc7 -> DSupplyDuct1;
    int ReturnDuctShape1 = pDoc7 -> RDuctShape1;
    int WidthOfReturnDuct1 = pDoc7 -> WReturnDuct1;
    int HeightOfReturnDuct1 = pDoc7 -> HReturnDuct1;
    double LengthOfReturnDuct1 = pDoc7 -> LReturnDuct1;
    int DiameterOfReturnDuct1 = pDoc7 -> DReturnDuct1;
    double MinPercentageOfAirFlow1 = pDoc7 -> MPercentageOfAirFlow1;
    double SensibleLoads1 = pDoc7 -> SLoads1;
    double TotalLoads1 = pDoc7 -> TLoads1;

    //Take inputs of zone2 from the document.
    double ZoneTemp2 = pDoc7 -> ZTemp2;
```

```
double ZoneRelHumidity2 = pDoc7 -> ZRelHumidity2;
double DesignSensibleLoads2 = pDoc7 -> DSensibleLoads2;
double DesignTotalLoads2 = pDoc7 -> DTotalLoads2;
int SupplyDuctShape2 = pDoc7 -> SDuctShape2;
int WidthOfSupplyDuct2 = pDoc7 -> WSupplyDuct2;
int HeightOfSupplyDuct2 = pDoc7 -> HSupplyDuct2;
double LengthOfSupplyDuct2 = pDoc7 -> LSupplyDuct2;
int DiameterOfSupplyDuct2 = pDoc7 -> DSupplyDuct2;
int ReturnDuctShape2 = pDoc7 -> RDuctShape2;
int WidthOfReturnDuct2 = pDoc7 -> WReturnDuct2;
int HeightOfReturnDuct2 = pDoc7 -> HReturnDuct2;
double LengthOfReturnDuct2 = pDoc7 -> LReturnDuct2;
int DiameterOfReturnDuct2 = pDoc7 -> DReturnDuct2;
double MinPercentageOfAirFlow2 = pDoc7 -> MPercentageOfAirFlow2;
double SensibleLoads2 = pDoc7 -> SLoads2;
double TotalLoads2 = pDoc7 -> TLoads2;

//Take inputs of zone3 from the document.
double ZoneTemp3 = pDoc7 -> ZTemp3;
double ZoneRelHumidity3 = pDoc7 -> ZRelHumidity3;
double DesignSensibleLoads3 = pDoc7 -> DSensibleLoads3;
double DesignTotalLoads3 = pDoc7 -> DTotalLoads3;
int SupplyDuctShape3 = pDoc7 -> SDuctShape3;
int WidthOfSupplyDuct3 = pDoc7 -> WSupplyDuct3;
int HeightOfSupplyDuct3 = pDoc7 -> HSupplyDuct3;
double LengthOfSupplyDuct3 = pDoc7 -> LSupplyDuct3;
int DiameterOfSupplyDuct3 = pDoc7 -> DSupplyDuct3;
int ReturnDuctShape3 = pDoc7 -> RDuctShape3;
int WidthOfReturnDuct3 = pDoc7 -> WReturnDuct3;
int HeightOfReturnDuct3 = pDoc7 -> HReturnDuct3;
double LengthOfReturnDuct3 = pDoc7 -> LReturnDuct3;
int DiameterOfReturnDuct3 = pDoc7 -> DReturnDuct3;
double MinPercentageOfAirFlow3 = pDoc7 -> MPercentageOfAirFlow3;
double SensibleLoads3 = pDoc7 -> SLoads3;
double TotalLoads3 = pDoc7 -> TLoads3;

//Take inputs of zone4 from the document.
double ZoneTemp4 = pDoc7 -> ZTemp4;
double ZoneRelHumidity4 = pDoc7 -> ZRelHumidity4;
double DesignSensibleLoads4 = pDoc7 -> DSensibleLoads4;
double DesignTotalLoads4 = pDoc7 -> DTotalLoads4;
```

```
int SupplyDuctShape4 = pDoc7 -> SDuctShape4;
int WidthOfSupplyDuct4 = pDoc7 -> WSupplyDuct4;
int HeightOfSupplyDuct4 = pDoc7 -> HSupplyDuct4;
double LengthOfSupplyDuct4 = pDoc7 -> LSupplyDuct4;
int DiameterOfSupplyDuct4 = pDoc7 -> DSupplyDuct4;
int ReturnDuctShape4 = pDoc7 -> RDuctShape4;
int WidthOfReturnDuct4 = pDoc7 -> WReturnDuct4;
int HeightOfReturnDuct4 = pDoc7 -> HReturnDuct4;
double LengthOfReturnDuct4 = pDoc7 -> LReturnDuct4;
int DiameterOfReturnDuct4 = pDoc7 -> DReturnDuct4;
double MinPercentageOfAirFlow4 = pDoc7 -> MPercentageOfAirFlow4;
double SensibleLoads4 = pDoc7 -> SLoads4;
double TotalLoads4 = pDoc7 -> TLoads4;

//Take inputs of zone5 from the document.
double ZoneTemp5 = pDoc7 -> ZTemp5;
double ZoneRelHumidity5 = pDoc7 -> ZRelHumidity5;
double DesignSensibleLoads5 = pDoc7 -> DSensibleLoads5;
double DesignTotalLoads5 = pDoc7 -> DTotalLoads5;
int SupplyDuctShape5 = pDoc7 -> SDuctShape5;
int WidthOfSupplyDuct5 = pDoc7 -> WSupplyDuct5;
int HeightOfSupplyDuct5 = pDoc7 -> HSupplyDuct5;
double LengthOfSupplyDuct5 = pDoc7 -> LSupplyDuct5;
int DiameterOfSupplyDuct5 = pDoc7 -> DSupplyDuct5;
int ReturnDuctShape5 = pDoc7 -> RDuctShape5;
int WidthOfReturnDuct5 = pDoc7 -> WReturnDuct5;
int HeightOfReturnDuct5 = pDoc7 -> HReturnDuct5;
double LengthOfReturnDuct5 = pDoc7 -> LReturnDuct5;
int DiameterOfReturnDuct5 = pDoc7 -> DReturnDuct5;
double MinPercentageOfAirFlow5 = pDoc7 -> MPercentageOfAirFlow5;
double SensibleLoads5 = pDoc7 -> SLoads5;
double TotalLoads5 = pDoc7 -> TLoads5;

double DesignSensibleLoads[5] = { DesignSensibleLoads1, DesignSensibleLoads2,
    DesignSensibleLoads3, DesignSensibleLoads4, DesignSensibleLoads5 };
double DesignTotalLoads[5] = { DesignTotalLoads1, DesignTotalLoads2,
    DesignTotalLoads3, DesignTotalLoads4, DesignTotalLoads5 };
double ZoneTemp[5] = { ZoneTemp1, ZoneTemp2, ZoneTemp3, ZoneTemp4,
ZoneTemp5 };
double ZoneRelHumidity[5] = { ZoneRelHumidity1, ZoneRelHumidity2,
    ZoneRelHumidity3, ZoneRelHumidity4, ZoneRelHumidity5 };
```

```
    MoistAir ZoneAir1 (ZoneTemp[0], ZoneRelHumidity[0] );
    MoistAir ZoneAir2 (ZoneTemp[1], ZoneRelHumidity[1] );
    MoistAir ZoneAir3 (ZoneTemp[2], ZoneRelHumidity[2] );
    MoistAir ZoneAir4 (ZoneTemp[3], ZoneRelHumidity[3] );
    MoistAir ZoneAir5 (ZoneTemp[4], ZoneRelHumidity[4] );
    MoistAir ZoneAir[5] = { ZoneAir1, ZoneAir2, ZoneAir3, ZoneAir4, ZoneAir5 };

    //Find the zone number with the largest design sensible loads.
    int k = MaxSubscript( DesignSensibleLoads );

    //Locate indoor air state point of the zone with the largest
    //design sensible loads.
    double Wz = ZoneAir[k].getHumidityRatio( );
    double Hz = ZoneAir[k].getEnthalpy( );
    double Vz = ZoneAir[k].getSpecificVolume( );

    //Determine the temperature of air leaving the cooling coil.
    double LeavingCoilTemp = ComputeLeavingCoilTemp( DesignSensibleLoads[k],
DesignTotalLoads[k], Hz,
                                ZoneTemp[k],        RelHumidityOfLeavingAir,
EnterChilledWaterTemp );

    if ( LeavingCoilTemp )
    {
        //Locate the outdoor air state point.
        MoistAir OutsideAir( OutsideTemp, OutsideRelHumidity );

        double Wo = OutsideAir.getHumidityRatio( );
        double Ho = OutsideAir.getEnthalpy( );
        double Vo = OutsideAir.getSpecificVolume( );
        double Mo = OutsideAirFlowRate / ( Vo * 3600 );

        //Locate the leaving coil air state point.
        MoistAir LeavingCoilAir( LeavingCoilTemp, RelHumidityOfLeavingAir );

        double Wleavingcoil = LeavingCoilAir.getHumidityRatio( );
        double Hleavingcoil = LeavingCoilAir.getEnthalpy( );

        //Compute the temperature rise cause by fan.
        //2004 ASHREA Handbook - HVAC Systems and Equipment (SI) 18.6.
        double FanTempRise = 82.9 * FanPressureRise / FanEfficiency;
```

```
int SupplyDuctShape[5] = { SupplyDuctShape1, SupplyDuctShape2,
SupplyDuctShape3, SupplyDuctShape4, SupplyDuctShape5 };

int ReturnDuctShape[5] = { ReturnDuctShape1, ReturnDuctShape2,
ReturnDuctShape3, ReturnDuctShape4, ReturnDuctShape5 };
double DesignMassFlowRate[5] = { 0, 0, 0, 0, 0 };
double DesignDuctTempRise[5] = { 3, 3, 3, 3, 3 };
double MassFlowRate[5] = { 0, 0, 0, 0, 0 };
double MinMassFlowRate[5] = { 0, 0, 0, 0, 0 };
double DuctTempRise1[5] = { 2, 2, 2, 2, 2 };
double AreaOfSupplyDuctWalls[5] = { 0, 0, 0, 0, 0 };
double AreaOfReturnDuctWalls[5] = { 0, 0, 0, 0, 0 };
double DuctTempRise2[5] = { 0, 0, 0, 0, 0 };
double Reheat[5] = { 0, 0, 0, 0, 0 };
double TempBeforeReheat[5] = { 0, 0, 0, 0, 0 };
double SupplyAirTemp[5] = { 0, 0, 0, 0, 0 };
double ReturnDuctTempRise[5] = { 0, 0, 0, 0, 0 };

int WidthOfSupplyDuct[5] = { WidthOfSupplyDuct1, WidthOfSupplyDuct2,
    WidthOfSupplyDuct3, WidthOfSupplyDuct4, WidthOfSupplyDuct5 };

int HeightOfSupplyDuct[5] = { HeightOfSupplyDuct1, HeightOfSupplyDuct2,
HeightOfSupplyDuct3, HeightOfSupplyDuct4, HeightOfSupplyDuct5 };

double LengthOfSupplyDuct[5] = { LengthOfSupplyDuct1,
LengthOfSupplyDuct2, LengthOfSupplyDuct3, LengthOfSupplyDuct4,
LengthOfSupplyDuct5 };

double LeakageFactor1 = 0;
double TotalMassFlowRate = 0;

int DiameterOfSupplyDuct[5] = { DiameterOfSupplyDuct1,
DiameterOfSupplyDuct2, DiameterOfSupplyDuct3, DiameterOfSupplyDuct4,
DiameterOfSupplyDuct5 };

int WidthOfReturnDuct[5] = { WidthOfReturnDuct1, WidthOfReturnDuct2,
    WidthOfReturnDuct3, WidthOfReturnDuct4, WidthOfReturnDuct5 };

int HeightOfReturnDuct[5] = { HeightOfReturnDuct1, HeightOfReturnDuct2,
HeightOfReturnDuct3, HeightOfReturnDuct4, HeightOfReturnDuct5 };
```

```
        double LengthOfReturnDuct[5] = { LengthOfReturnDuct1,
LengthOfReturnDuct2,
            LengthOfReturnDuct3, LengthOfReturnDuct4, LengthOfReturnDuct5 };

        double MinPercentageOfAirFlow[5] = { MinPercentageOfAirFlow1,
MinPercentageOfAirFlow2, MinPercentageOfAirFlow3, MinPercentageOfAirFlow4,
MinPercentageOfAirFlow5 };

        int DiameterOfReturnDuct[5] = { DiameterOfReturnDuct1,
DiameterOfReturnDuct2, DiameterOfReturnDuct3, DiameterOfReturnDuct4,
DiameterOfReturnDuct5 };

        double SensibleLoads[5] = { SensibleLoads1, SensibleLoads2, SensibleLoads3,
SensibleLoads4, SensibleLoads5 };

        double HumidityRatio[5] = { 0, 0, 0, 0, 0 };
        double SupplyMassFlowRate[5] = { 0, 0, 0, 0, 0 };
        double ReturnMassFlowRate[5] = { 0, 0, 0, 0, 0 };

        double Tr = 0;
        double Wr = 0;
        double Hr = 0;
        double Hmixing = 0;

        //<<Air conditioning>>(Chinese edition) p.151.
        double ExfilRate = 0.07;
        double TotalReturnMassTemp = 0;
        double TotalReturnMassHumidityRatio = 0;
        double TotalReturnMassFlowRate = 0;
        double ReturnAirFlowRate = 0;
        double CoolingCapacity = 0;
        double ExhaustAirFlowRate = 0;

        //Compute humidity ratios for each zone.
        for ( int b = 0; b < NumOfZones; b++ )
            HumidityRatio[b] = ZoneAir[b].getHumidityRatio( );

        //Compute area of supply air duct walls for each zone.
        for ( int i = 0; i < NumOfZones; i++ )
        {
            if ( SupplyDuctShape[i] == 1 )
```

```
                AreaOfSupplyDuctWalls[i] = ( 2 * ( WidthOfSupplyDuct[i] +
HeightOfSupplyDuct[i] )* LengthOfSupplyDuct[i] ) / 1000;
            if ( SupplyDuctShape[i] == 2 )
                AreaOfSupplyDuctWalls[i] = ( 3.14 * DiameterOfSupplyDuct[i]
                                            * LengthOfSupplyDuct[i] ) / 1000;
        }

        //Compute area of return air duct walls for each zone.
        for ( int l = 0; l < NumOfZones; l++ )
        {
            if ( ReturnDuctShape[l] == 1 )
                AreaOfReturnDuctWalls[l] = ( 2 * ( WidthOfReturnDuct[l] +
HeightOfReturnDuct[l] )* LengthOfReturnDuct[l] ) / 1000;

            if ( ReturnDuctShape[l] == 2 )
                AreaOfReturnDuctWalls[l] = ( 3.14 * DiameterOfReturnDuct[l]
                                            * LengthOfReturnDuct[l] ) / 1000;
        }

        //Determine design and minimum air mass flow rates for each zone.
        for ( int j = 0; j < NumOfZones; j++ )
        {
            MassFlowRateAndDuctTempRise( TempOfSurroundingAir, ZoneTemp[j],
DesignSensibleLoads[j], SupplyDuctShape[j], AreaOfSupplyDuctWalls[j],
LeavingCoilTemp, FanTempRise, DesignMassFlowRate[j], DesignDuctTempRise[j] );

            MinMassFlowRate[j] = ( MinPercentageOfAirFlow[j] *
DesignMassFlowRate[j] ) / 100;
        }

        //Consider off-design conditions.
        //Compute before reheat temperature, supply air temperature
        //and reheat amount(if required) for each zone.
        for ( int m = 0; m < NumOfZones; m++ )
        {
            MassFlowRateAndDuctTempRise( TempOfSurroundingAir, ZoneTemp[m],
SensibleLoads[m], SupplyDuctShape[m], AreaOfSupplyDuctWalls[m],
LeavingCoilTemp, FanTempRise, MassFlowRate[m], DuctTempRise1[m] );

            if ( SupplyDuctShape[m] == 1 )
                LeakageFactor1 = 0.05;
```

```
            if ( SupplyDuctShape[m] == 2 )
                LeakageFactor1 = 0.02;

            if ( MassFlowRate[m] >= MinMassFlowRate[m] )
            {
                //No reheat is required.
                TempBeforeReheat[m] = LeavingCoilTemp + FanTempRise +
DuctTempRise1[m];
                SupplyAirTemp[m] = TempBeforeReheat[m];
                Reheat[m] = 0;
            }
            else
            {
                //Reheat is required.
                MassFlowRate[m] = MinMassFlowRate[m];
                //HVAC Engineer's Handbook p.163.
                DuctTempRise2[m] = ( TempOfSurroundingAir - LeavingCoilTemp -
FanTempRise )
                                / ( 0.5 + ( 439 * MassFlowRate[m] ) /
AreaOfSupplyDuctWalls[m] );

                TempBeforeReheat[m] = LeavingCoilTemp + FanTempRise +
DuctTempRise2[m];

                SupplyAirTemp[m] = ZoneTemp[m] - SensibleLoads[m] / ( 1.02 *
MassFlowRate[m] * ( 1 - LeakageFactor1 ) );
                Reheat[m] = 1.02 * MassFlowRate[m] * ( 1 - LeakageFactor1 )
                        * ( SupplyAirTemp[m] - TempBeforeReheat[m] );
            }

            SupplyMassFlowRate[m] = MassFlowRate[m] * ( 1 - LeakageFactor1 );

            //Store the results in the document.
            pDoc7 -> TBeforeReheat[m] = TempBeforeReheat[m];
            pDoc7 -> SAirTemp[m] = SupplyAirTemp[m];
            pDoc7 -> Rheat[m] = Reheat[m];
            pDoc7 -> SAirFlowRate[m] =  ( SupplyMassFlowRate[m] * 1000 ) / 1.2;}

        //Compute the total mass flow rate.
        for ( int n = 0; n < NumOfZones; n++ )
            TotalMassFlowRate = TotalMassFlowRate + MassFlowRate[n];
```

```
        //Compute return air mass flow rate for each zone
        //and the total return air mass flow rate.
        for ( int c = 0; c < NumOfZones; c++ )
        {
            if ( SupplyDuctShape[c] == 1 )
                LeakageFactor1 = 0.05;
            if ( SupplyDuctShape[c] == 2 )
                LeakageFactor1 = 0.02;

            ReturnMassFlowRate[c] = SupplyMassFlowRate[c] * (1 - ExfilRate )
                                    * (1 - LeakageFactor1);

            TotalReturnMassFlowRate = TotalReturnMassFlowRate +
ReturnMassFlowRate[c];}
        //Compute return duct temperature rise corresponding to each zone.
        for ( int d = 0; d < NumOfZones; d++ )
            //HVAC Engineer's Handbook p.163.
            ReturnDuctTempRise [d] = (TempOfSurroundingAir - ZoneTemp[d])
                                / (0.5 + (439 * ReturnMassFlowRate[d]) /
AreaOfReturnDuctWalls[d] );

        //Locate the overall return air state point.
        for ( int e = 0; e < NumOfZones; e++ )
        {
            TotalReturnMassTemp = TotalReturnMassTemp + ReturnMassFlowRate[e]
            * (ZoneTemp[e] + ReturnDuctTempRise [e]);

            TotalReturnMassHumidityRatio = TotalReturnMassHumidityRatio
                                           + ReturnMassFlowRate[e] *
HumidityRatio[e];
        }

        //Return air.
        Tr = TotalReturnMassTemp / TotalReturnMassFlowRate;

        Wr = TotalReturnMassHumidityRatio / TotalReturnMassFlowRate;

        Hr = 1.006 * Tr + Wr * (2501 + 1.805 * Tr);

        //Determine exhaust air flow rate.
        ReturnAirFlowRate = TotalMassFlowRate - Mo;
```

```
        ExhaustAirFlowRate = TotalReturnMassFlowRate - ReturnAirFlowRate;

        //Mixing air.
        Hmixing = (Mo * Ho + ReturnAirFlowRate * Hr) / TotalMassFlowRate;


        //Determine cooling capacity of the coil.
        CoolingCapacity = TotalMassFlowRate * ( Hmixing - Hleavingcoil );

        //Print the results.
        m_CoolingCapacity.Format ("%6.1f", CoolingCapacity);

        m_CoilVolumeFlowRate.Format("%6.2f", ( TotalMassFlowRate * 1000 ) /1.2);

        m_ExhaustAirFlowRate.Format("%6.2f", ( ExhaustAirFlowRate * 1000 ) / 1.2);

        m_RelHumidityOfLeavingAir.Format("%6.0f", RelHumidityOfLeavingAir);

        m_LeavingAirTemp.Format ("%6.1f", LeavingCoilTemp);
        //m_Reheat.Format ("%6.1f", Reheat);

        m_ReturnAirFlowRate.Format ("%6.2f", (ReturnAirFlowRate * 1000) / 1.2);

        m_ReturnAirTemp.Format("%6.1f",Tr);

        //m_SupplyAirFlowRate.Format ("%6.2f", (MassFlowRate * (1 -
LeakageFactor1) * 1000) / 1.2);

        //m_TempBeforeReheat.Format ("%6.1f", TempBeforeReheat);

        //m_SupplyAirTemp.Format ("%6.1f", SupplyAirTemp);
    }
    else
        MessageBox ("Supply air temperature is too low to achieve by the cooling
coil.");

    UpdateData(FALSE);
}
```

```
#if !defined(AFX_ZONE1OUTPUTDLG_H__B04CD06A_8D74_4F08_B160_AD32C5
92FD90__INCLUDED_)
#define
AFX_ZONE1OUTPUTDLG_H__B04CD06A_8D74_4F08_B160_AD32C592FD90__I
NCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// Zone1OutputDlg.h : header file
//

/////////////////////////////////////////////////////////////////////////
// CZone1OutputDlg dialog

class CZone1OutputDlg : public CDialog
{
// Construction
public:
    CZone1OutputDlg (CWnd* pParent = NULL);    // standard constructor

// Dialog Data
    //{{AFX_DATA(CZone1OutputDlg)
    enum { IDD = IDD_ZONE1OUTPUT };
    CString m_Reheat1;
    CString m_SupplyAirFlowRate1;
    CString m_SupplyAirTemp1;
    CString m_TempBeforeReheat1;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CZone1OutputDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
```

```
    //{{AFX_MSG(CZone1OutputDlg)
    virtual BOOL OnInitDialog();
    afx_msg void OnButton8();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

#endif
// !defined(AFX_ZONE1OUTPUTDLG_H__B04CD06A_8D74_4F08_B160_AD32C592FD90__INCLUDED_)

/////////////////////////////////////////////////////////////////////////////
// Zone1OutputDlg.cpp : implementation file

#include "stdafx.h"
#include "MultiVAV5.h"
#include "Zone1OutputDlg.h"

//#include "moistair.h"
#include "MultiVAV5Doc.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CZone1OutputDlg dialog

CZone1OutputDlg::CZone1OutputDlg (CWnd* pParent /*=NULL*/)
    : CDialog(CZone1OutputDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CZone1OutputDlg)
    m_Reheat1 = _T("");
```

```
    m_SupplyAirFlowRate1 = _T("");
    m_SupplyAirTemp1 = _T("");
    m_TempBeforeReheat1 = _T("");
    //}}AFX_DATA_INIT
}

void CZone1OutputDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CZone1OutputDlg)
    DDX_Text(pDX, IDC_REHEAT1, m_Reheat1);
    DDX_Text(pDX, IDC_SUPPLYAIRFLOWRATE1, m_SupplyAirFlowRate1);
    DDX_Text(pDX, IDC_SUPPLYAIRTEMP1, m_SupplyAirTemp1);
    DDX_Text(pDX, IDC_TEMPBEFOREREHEAT1, m_TempBeforeReheat1);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CZone1OutputDlg, CDialog)
    //{{AFX_MSG_MAP(CZone1OutputDlg)
    ON_BN_CLICKED(IDC_BUTTON8, OnButton8)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CZone1OutputDlg message handlers

BOOL CZone1OutputDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // TODO: Add extra initialization here
    m_Reheat1 = "0.0";
    m_SupplyAirFlowRate1 = "0.0";
    m_SupplyAirTemp1 = "0.0";
    m_TempBeforeReheat1 = "0.0";

    UpdateData (FALSE);

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}
```

```
void CZone1OutputDlg::OnButton8()
{
    CMultiVAV5Doc*          pDoc8          =          (CMultiVAV5Doc*)
((CMainFrame*)AfxGetMainWnd())->GetActiveDocument();

    //Print the results.
    m_TempBeforeReheat1.Format("%6.1f", pDoc8 -> TBeforeReheat[0]);
    m_SupplyAirTemp1.Format("%6.1f", pDoc8 -> SAirTemp[0]);
    m_SupplyAirFlowRate1.Format("%6.2f", pDoc8 -> SAirFlowRate[0]);
    m_Reheat1.Format("%6.1f", pDoc8 -> Rheat[0]);

    UpdateData(FALSE);
}
```

APPENDIX B

OPTIMIZER

```
//Developed in partial fulfillment of the requirements for the degree of Master of
//Applied Science at Concordia University
//A Real-time Simulation Tool for Fault Detection and Diagnosis of HVAC Systems
//Author: Yue MA
//December 2004

%Only suitable for Mode#1
function main

global Qs1;
global Qs2;%Msa1, Msa2
global Cpa;
%global Tma;
global Toa;
global Moa;
%global Rminoa;%(Minimum ratio of outdoor air intake to supply airflow rate)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Input the required values for optimization
%Begin of input
%Transient loads of Zones1&2
Qs1 = 30560;%(Btu/hr)
Qs2 = 15980;%(Btu/hr)

%Mixed (entering coil) air temperature
%Tma = 80;

%Outdoor air temperature
Toa = 69;%(F)

%Required outdoor air intake
CFMoa = 1120;%(cfm)

%End of input
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Input the design mass flow rates of Zones1&2
Msa1d = 25490;%(lbm/hr)
Msa2d = 18611;%(lbm/hr)

%Input the design chilled water flow rate
GPMd = 54;%(gpm)

Moa = 4.61 * CFMoa;

%Air specific heat
Cpa = 0.24;

%Water specific heat
Cw = 1;

%Temperature difference of chilled water
ChilledTempDiff = 10;

X0 = [78, 75, 55];

lb = [73, 73, 55];
ub = [78, 78, 64];

b1 = -Qs1/(Cpa*Msa1d) - 1;
b2 = -Qs2/(Cpa*Msa2d) - 1;
b3 = Qs1/(0.3*Cpa*Msa1d) + 1;
b4 = Qs2/(0.3*Cpa*Msa2d) + 1;
b5 = -Toa - 1;

%A12: Matrix for Modes 1&2
A12 = [-1, 0, 1;
     0, -1, 1;
     1, 0, -1;
      0, 1, -1];

%b12: Vector for Modes 1&2
b12 = [b1, b2, b3, b4];
```

```
%A3: Matrix for Mode3
A3 = [-1, 0, 1;
      0, -1, 1;
      1, 0, -1;
      0, 1, -1;
       0, 0, -1];

%b3: Vector for Mode3
b3 = [b1, b2, b3, b4, b5];

X = fmincon(@myfun1, X0, A12, b12, [], [], lb, ub);

%Output the values corresponding to the optimal state
Tz1 = X(1);
Tz2 = X(2);
Tda = X(3);
% Twi = Tda - 11

%Supply fan temperature rise is assumed to be 1 F.
Msa1 = Qs1/(Cpa*(Tz1 - Tda - 1));
Msa2 = Qs2/(Cpa*(Tz2 - Tda - 1));

%Return fan temperature rise is assumed to be 1 F.
Tra = ( Msa1 * Tz1 + Msa2 * Tz2 )/(Msa1 + Msa2) + 1;

Tma = ( Moa * ( Toa - Tra ) + ( Msa1 * Tz1 + Msa2 * Tz2 ) )/(Msa1 + Msa2);

GPM = (Msa1 + Msa2)*Cpa*( Tma - Tda)/(Cw*ChilledTempDiff*495);

if (Tra <= Toa)%then Mode1
      Tz1set = Tz1
      Tz2set = Tz2
      Tdaset = Tda
      Twi = Tdaset - 11
      CFMsa1 = 0.217 * Msa1
      CFMsa2 = 0.217 * Msa2;

      Tra = ( Msa1 * Tz1set + Msa2 * Tz2set )/(Msa1 + Msa2) + 1

      Tma = ( Moa * Toa + ( Msa1 + Msa2 - Moa ) * Tra )/(Msa1 + Msa2)
```

```
    %Output the open position of supply air damper for Zones 1&2
    Usa1 = Msa1/Msa1d
    Usa2 = Msa2/Msa2d

    %Output the open position of the cooling coil valve
    Uw = GPM/GPMd
    Mode = 1

else%then Modes2&3
    X= fmincon(@myfun2, X0, A12, b12, [], [], lb, ub);

    %Output the values corresponding to the optimal state
    Tz1s = X(1);
    Tz2s = X(2);
    Tdas = X(3);

    %Supply fan temperature rise is assumed to be 1 F.
    Msa1 = Qs1/(Cpa*(Tz1s - Tdas - 1));
    Msa2 = Qs2/(Cpa*(Tz2s - Tdas - 1));

    GPM = (Msa1 + Msa2)*Cpa*( Toa - Tdas)/(Cw*ChilledTempDiff*495);
%   Usa1 = Msa1/Msa1d
%   Usa2 = Msa2/Msa2d
    Uw = GPM/GPMd;

    if (Uw <= 0)%then Mode3
        %('No mechanical cooling is required.')

        X= fmincon(@myfun3, X0, A3, b3, [], [], lb, ub);

        %Output the values corresponding to the optimal state
        Tz1set = X(1)
        Tz2set = X(2)
        Tdaset = X(3)
        %Twi = Tdaset - 11;

        %Supply fan temperature rise is assumed to be 1 F.
        Msa1 = Qs1/(Cpa*(Tz1set - Tdaset - 1));
        Msa2 = Qs2/(Cpa*(Tz2set - Tdaset - 1));

        Usa1 = Msa1/Msa1d
```

```
        Usa2 = Msa2/Msa2d
        Mode = 3
    else%then Mode2

        %Output the values corresponding to the optimal state
        Tz1set = X(1)
        Tz2set = X(2)
        Tdaset = X(3)
        Twi = Tdaset - 11

        %Supply fan temperature rise is assumed to be 1 F.
        Msa1 = Qs1/(Cpa*(Tz1set - Tdaset - 1));
        Msa2 = Qs2/(Cpa*(Tz2set - Tdaset - 1));

        Usa1 = Msa1/Msa1d
        Usa2 = Msa2/Msa2d
        Uw = GPM/GPMd
        Mode = 2
    end
end


%Mode#1
function f1 = myfun1(X)


global Qs1;
global Qs2;
global Cpa;
%global Tma;
global Toa;
global Moa;
%Tma = 70;
%Qs1 = 139890;
%Qs2 = 63450;
SupplyFanPressureDiff = 1;
SupplyFanEff = 0.75;

ReturnFanPressureDiff = 0.8;
ReturnFanEff = 0.65;
```

```
COP = 1.04;%(ton/hp)

%Moa = 4.61 * CFMoa;

%Supply fan temperature rise is assumed to be 1 F.
Ma1 = Qs1/(Cpa * (X(1) - X(3) - 1));
Ma2 = Qs2/(Cpa * (X(2) - X(3) - 1));

Mat = Ma1 + Ma2;

Tra = (Ma1 * X(1) + Ma2 * X(2))/Mat + 1;

Tma = (Moa * ( Toa - Tra ) + (Ma1 * X(1) + Ma2 * X(2)))/Mat;

f1 = (Mat * Cpa * ( Tma - X(3)))/COP + SupplyFanPressureDiff * 0.217 * (Ma1 + Ma2)/(6350 * SupplyFanEff) + ReturnFanPressureDiff * 0.217 * 0.9 * (Ma1 + Ma2)/(6350 * ReturnFanEff);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Mode#2
function f2 = myfun2(X)

global Qs1;
global Qs2;
global Cpa;
%global Tma;
global Toa;
%global Moa;
%Tma = 70;
%Qs1 = 139890;
%Qs2 = 63450;
SupplyFanPressureDiff = 1;
SupplyFanEff = 0.75;

ReturnFanPressureDiff = 0.8;
ReturnFanEff = 0.65;

COP = 1.04;%(ton/hp)

%Moa = 4.61 * CFMoa;
```

```
%Supply fan temperature rise is assumed to be 1 F.
Ma1 = Qs1/(Cpa * (X(1) - X(3) - 1));
Ma2 = Qs2/(Cpa * (X(2) - X(3) - 1));


Mat = Ma1 + Ma2;

f2 = (Mat * Cpa * ( Toa - X(3)))/COP + SupplyFanPressureDiff * 0.217 * (Ma1 +
Ma2)/(6350 * SupplyFanEff) + ReturnFanPressureDiff * 0.217 * 0.9 * (Ma1 +
Ma2)/(6350 * ReturnFanEff);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Mode#3
function f3 = myfun3(X)


global Qs1;
global Qs2;
global Cpa;
%global Tma;
% global Toa;
%global Moa;
%Tma = 70;
%Qs1 = 139890;
%Qs2 = 63450;
SupplyFanPressureDiff = 1;
SupplyFanEff = 0.75;
ReturnFanPressureDiff = 0.8;
ReturnFanEff = 0.65;

COP = 1.04;%(ton/hp)

%Moa = 4.61 * CFMoa;

%Supply fan temperature rise is assumed to be 1 F.
Ma1 = Qs1/(Cpa * (X(1) - X(3) - 1));
Ma2 = Qs2/(Cpa * (X(2) - X(3) - 1));

Mat = Ma1 + Ma2;
```

```
f3 = SupplyFanPressureDiff * 0.217 * Mat/(6350 * SupplyFanEff) + ReturnFanPressureDiff * 0.217 * 0.9 * Mat/(6350 * ReturnFanEff);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function   [f2,f3] = mycon(X)
% global Qs1 Qs2
% global Cpa;
% global Tma;
% global Toa;
% global CFMoa;
% global Moa;
% Moa = 4.61 * CFMoa;
% f2 = Qs2 * X(1) + Qs1 * X(2) - ( Qs1 + Qs2 ) * X(3) - 10 * Cpa * Moa * X(3)^2 + 10 * Cpa * Moa * ( X(1) + X(2) ) * X(3) - 10 * Cpa * Moa * X(1) * X(2);
% f3=[];
```

APPENDIX C

CASE STUDIES

➢ Case 7

Case 7 simulated a leaking cooling coil valve. This fault may be easily seen when Mode# 3 is taking effect. The input data is listed in **Table D.1**. In Mode#3, no mechanical cooling is needed and the cooling coil valve is expected to be completely closed. In this test, when time = 828s, the cooling coil valve began to leak. It was assumed that the leakage flow rate was 10% of the design value. In the presence of this fault, the air temperature at the exit of the cooling coil was lower than the mixed air temperature, i.e., the discharge air was overcooled. As shown in **Fig. D-1**, the discharge air temperature arrived at 50.6F (i.e., T_{da} = 50.6F). It should be noted that the fault did not cause the zone temperatures to drift away from the setpoint values. This was because the supply air damper control signals had decreased to compensate for the fault. Accordingly, it is concluded that the fault – *cooling coil valve leaking* – may be observed due to satisfaction of *Rule 4*.

Table D.1 Input of case 7

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,400	43,860	18,740	53	400	220	3
2,403	6,000	71,960	30,660	53	400	220	3

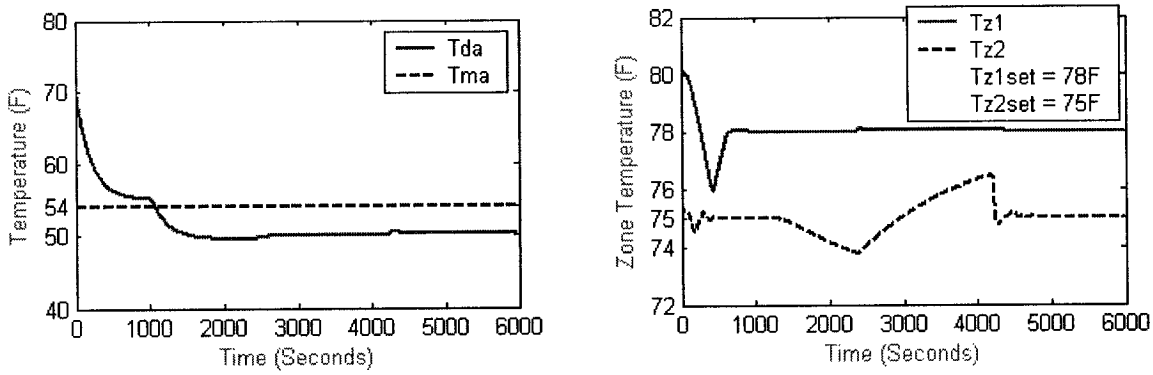


Figure D-1 Result of case 7

➢ **Case 8**

Case 8 simulated the fault that design chilled water flow rate was too low. This fault can only be observed when the system is experiencing the design load. The input data is listed in **Table D.2**. In this test, the discharge air temperature setpoint was 55F (i.e., T_{daset} = 55F). It was assumed that the cooling coil had 25 gpm of the design flow rate. Therefore, the cooling capacity would not suffice for design conditions. As a result, the discharge air temperature setpoint could not be achieved. As shown in **Fig. D-2**, the discharge air temperature was 57.7F (i.e., T_{da} = 57.7F). It was also noted that Zone 1 temperature setpoint value was reached and Zone 2 temperature (T_{z2} = 77.6F) was greater than the setpoint value. Zone 1 was experiencing a partial load; therefore, the increased supply air temperature was compensated by increasing the supply air damper control signal. However, the peak load was acting on Zone 2; therefore, the supply air damper control signal saturated at its high limit (i.e., 100% open). Therefore, for Zone 2, there was no way to compensate for the fault. Accordingly, it is concluded that the fault – *design chilled*

water flow rate was low - may be found due to satisfaction of *Rule 3*.

Table D.2 Input of case 8

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,400	110,390	85,280	88	700	420	1
2,403	6,000	124,220	88,820	90	700	420	1

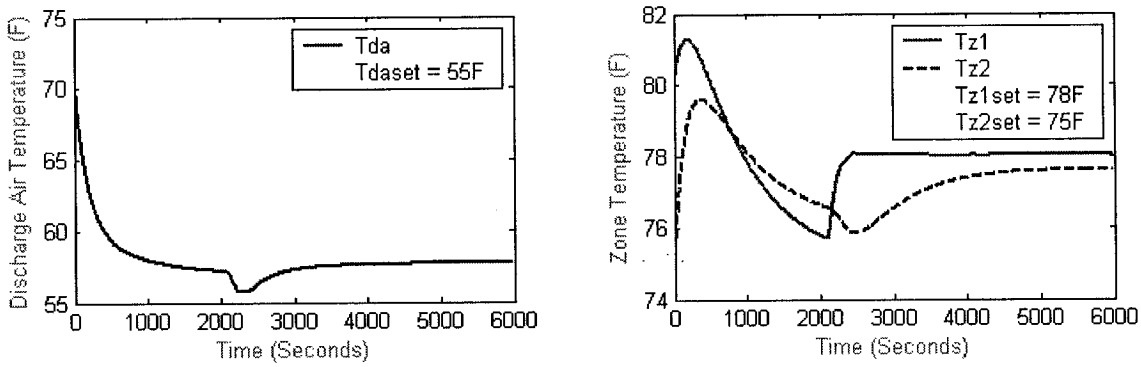


Figure D-2 Result of case 8

➢ **Case 9**

Case 9 simulated the fault that design supply airflow rates were too low. The input data is listed in **Table D.3**. In this test, it was assumed that the design supply airflow rates to Zones 1 and 2 were *4000 cfm* and *3000 cfm* respectively. Since the supply airflow rates did not suffice for design conditions, the temperatures of both zones would drift away from the setpoint values. As shown in **Fig. D-3**, Zone 1 temperature was 83F (i.e., T_{z1} = 83F) and Zone 2 temperature was 81.8F (i.e., T_{z2} = 81.8F). Accordingly, it is concluded that the

fault – *design supply airflow rate too low* - may be detected due to satisfaction of *Rule 9*.

Table D.3 Input of case 9

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone1	Zone2		Zone1	Zone2	
0	3,009	110,390	85,280	88	700	420	1
3,012	6,000	124,220	88,820	90	700	420	1

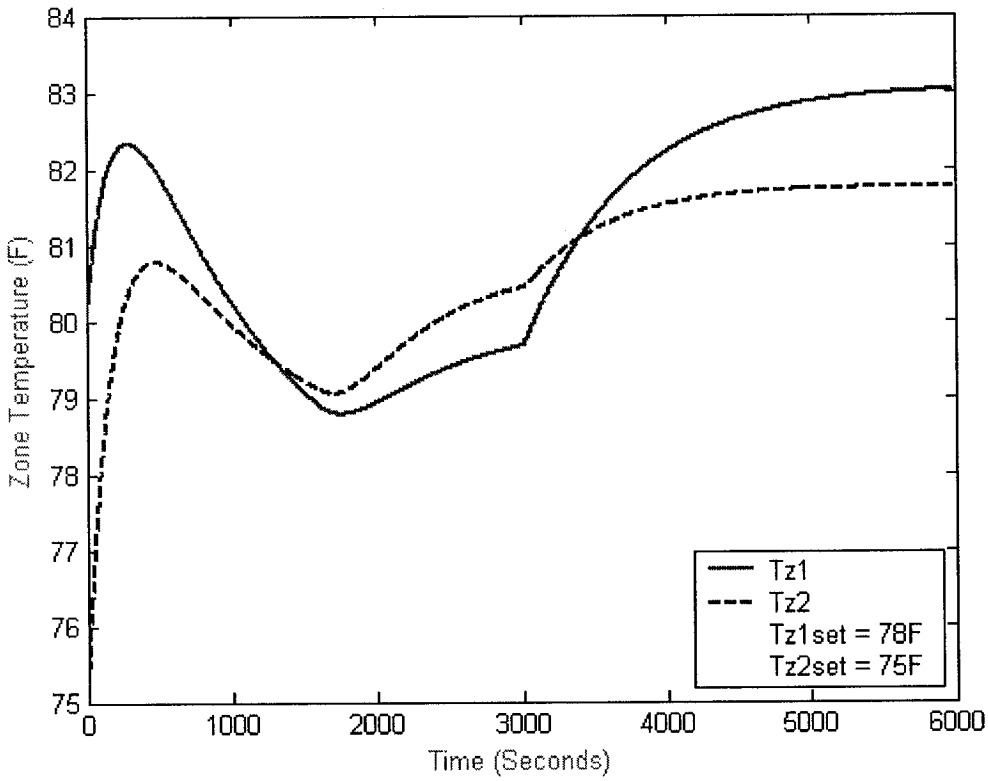


Figure D-3 Result of case 9

➢ **Case 10**

Case 10 simulated a leaking reheat coil valve. The input data is listed in **Table D.4**.Whenever neither of the zones requires reheat, the reheat coil valves are expected to be closed. Therefore, the temperature of air supplied to Zones 1 and 2 are exactly the same as the discharge air temperature. In this test, when time = 1,515s, the valve of the reheat coil of Zone 2 began to leak. It was assumed that the leakage flow rate was 10% of the design value. As a result, the supply air to Zone 2 was heated so that the supply air temperature was greater than discharge air temperature. As shown in **Fig. D-4**, temperature of the air supplied to Zone 2 arrived at 57F (i.e., T_{sa2} = 57F). However, this fault did not cause Zone 2 undercooled because the supply air damper had opened further to compensate for the fault. Accordingly, it is concluded that the fault – *reheat coil valve leaking* - may be detected due to satisfaction of *Rule 6*.

Table D.4 Input of case 10

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone1	Zone2	
0	2,400	71,960	30,660	82	400	220	1
2,403	6,000	88,790	36,800	88	400	220	1

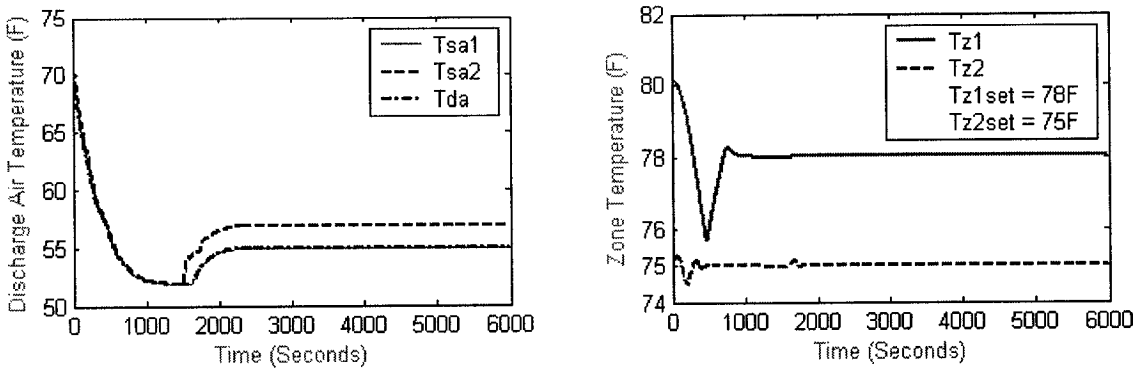


Figure D-4 Result of case 10

➢ **Case 11**

Case 11 simulated a leaking return air damper. The input data is listed in **Table D.5**. In Mode#2, the return air damper is supposed to be closed completely. Accordingly, the mixed air temperature would be the same as the outdoor air temperature. In this test, when time = 1,251s, the return air damper began to leak. It was assumed that the leakage airflow rate was 7% of the design value. As a result, the temperature of mixed air was greater than that of the outdoor air. As shown in **Fig. D-5**, the mixed air temperature arrived at 66.5F (i.e., T_{ma} = 66.5F). It should also be noted that the discharge air temperature setpoint was still be achieved. This was because that the cooling coil valve control signal had increased to compensate for this fault. When this happened, mechanical cooling energy would be used more than necessary. Accordingly, it is concluded that the fault – *return air damper leaking* - may be detected due to satisfaction of *Rule 18*.

Table D.5 Input of case 11

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,679	71,960	30,660	65	300	120	2
2,682	6,000	88,790	36,800	65	300	120	2

➢ **Case 12**

Case 12 simulated a stuck return air damper, which belonged to economizer cycle control faults. This fault is a failure of a linkage in the mixing box dampers. It affects the airflow rates in the mixing box and, therefore, the actual mixed air temperature is different from the expected value. For each operation mode, the fault pattern is different. Therefore, the case consisted of three tests, each for one mode.

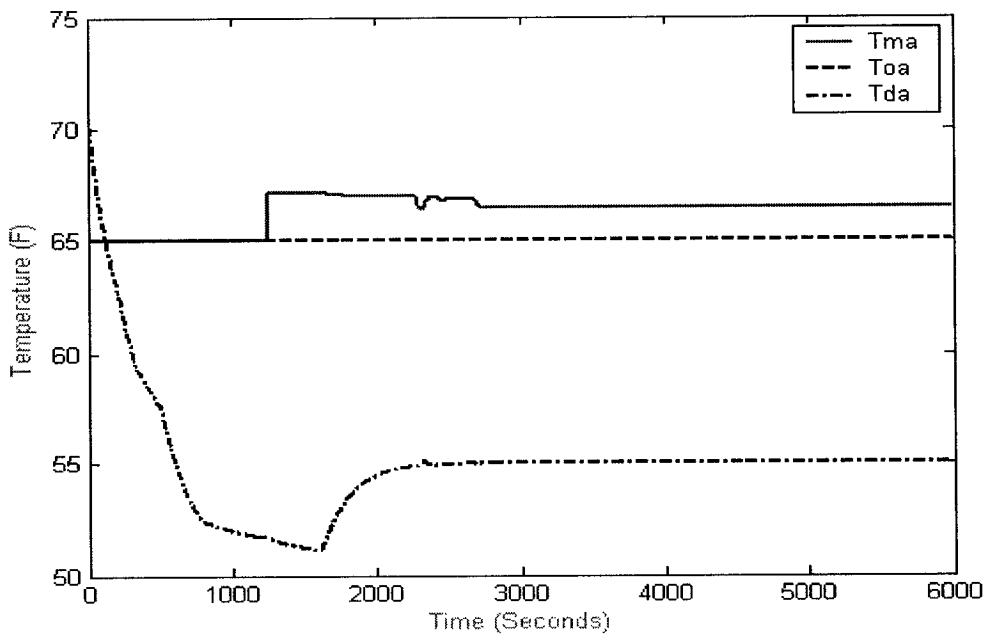


Figure D-5 Result of case 11

i. Test1

The input data is listed in **Table D.6a**. In Mode#1, minimum outdoor air intake is needed. In this test, when time = 1,233s, the return air damper was stuck at 30% open position so that return airflow rate was the less than the expected value. As a result, the mixed air temperature went up so that the actual ratio of outdoor air intake to supply airflow rate was greater that the expected ratio. As the result showed, the actual ratio was 0.52 (i.e., R_{osact} = 0.52), which was greater than 0.18 of the expected ratio of outdoor air intake to supply airflow rate (i.e., R_{osexp} = 0.18).

Table D.6a Input of test 1

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,337	88,790	36,800	88	400	220	1
2,340	6,000	107,100	45,360	88	400	220	1

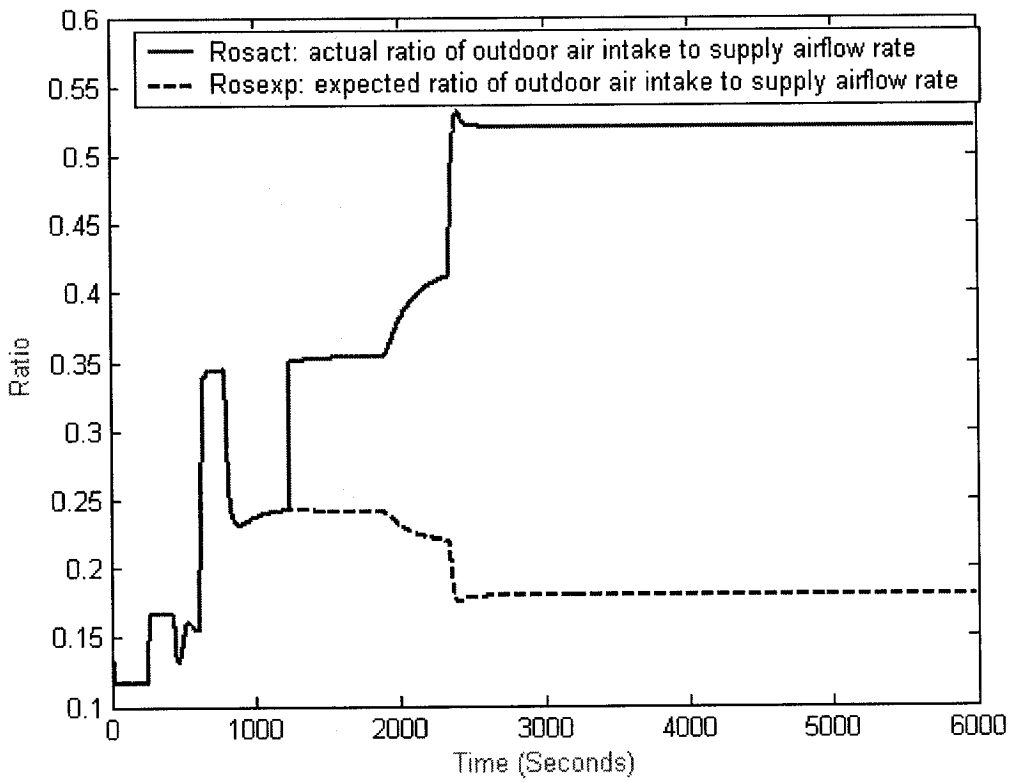


Figure D-6a Result of test 1

ii. Test 2

The input data is listed in **Table D.6b**. In Mode#2 the return air damper is supposed to be 100% closed. Accordingly, the mixed air temperature would be equal to the outdoor air temperature. In this test, when time = 1,500s, the return air damper was stuck at 17% open position so that some amount of return air was mixed with the outdoor air. As a result, the mixed air temperature went up. As shown in **Fig. D-6b**, the mixed air temperature was 73.5F (i.e., T_{ma} = 73.5F). Therefore, natural cooling effect of outdoor air was not taken full advantage of and mechanical cooling was used excessively.

Table D.6b Input of test 2

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone1	Zone2		Zone1	Zone2	
0	2,508	88,790	36,800	72	400	220	2
2,511	6,000	107,100	45,360	72	400	220	2

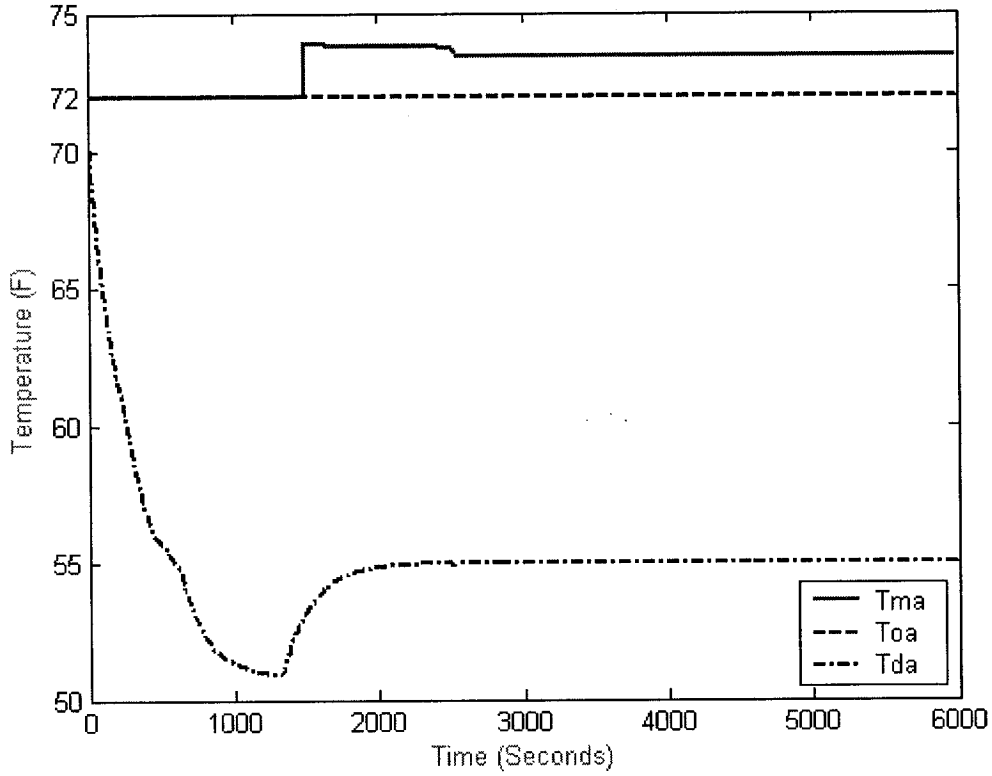


Figure D-6b Result of test 2

iii. Test 3

The input data is listed in **Table D.6c**. In Mode#3, no mechanical cooling is needed. Economizers use controllable dampers to mix outdoor air and return air in appropriate quantities to maintain discharge air temperature at the setpoint value. In this test, when time = 1,815s, the return air damper was stuck at 6% open position so that less than

required return airflow rate was mixed with the outdoor air. As a result, the mixed air temperature was lower than it was supposed to be, causing the discharge air temperature less than the setpoint value. As shown in **Fig. D-6c**, the discharge air temperature was 52F (i.e., T_{da} = 52F). It should be noted that the zones would not be overcooled because the supply air damper control signals had decreased to compensate for the fault.

Table D.6c Input of test 3

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,361	88,790	36,800	48	400	220	3
2,364	6,000	107,100	45,360	48	400	220	3

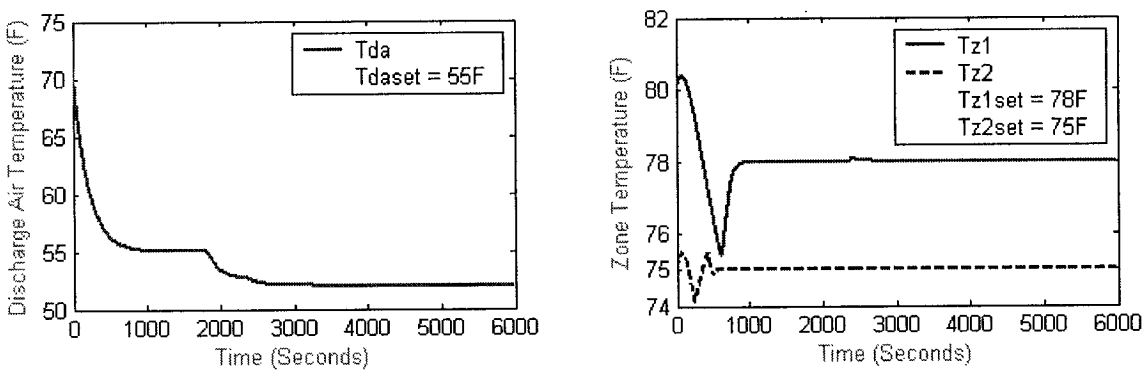


Figure D-6c Result of test 3

From the three tests, it is concluded that the fault – *return air damper stuck* – may be seen due to satisfaction of *Rules 14*, *18* and *4* for Modes 1 thru 3 respectively.

➢ **Case 13**

Case 13 simulated the fault that the chilled water controller is improperly tuned. The input data is listed in **Table D.7** and the result is shown in **Fig. D-7**. If the controller is properly tuned (e.g., K_p = 1 and K_i = 0.005), the discharge air temperature will be equal to the setpoint when the steady state is reached. In this test where the controller was improperly tuned (e.g., K_p = 3 and K_i = 2), the discharge air temperature could not be stable but fluctuate about the setpoint. Accordingly, it is concluded that the fault – *chilled water PI controller tuning error* – may be seen due to satisfaction of *Rule 5*.

Table D.7 Input of case 13

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,382	71,960	30,660	70	300	120	2
2,385	6,000	88,790	36,800	70	300	120	2

➢ **Case 14**

Case 14 simulated the fault that VAV box damper controllers were improperly tuned. The input data is listed in **Table D.8** and the result is shown in **Fig. D-8**. If the VAV box controllers are properly tuned (e.g., K_p = 1 and K_i = 0.01), the zone temperatures will be maintained at the setpoint values when the steady state is reached. In the test where the controllers were improperly tuned (e.g., K_p = 3 and K_i = 1.5), neither of the zone temperature setpoints could be stable but fluctuate about the respective setpoints.

Accordingly, it is concluded that the fault – *VAV box PI controllers tuning error* - may be detected due to satisfaction of *Rule 11*.

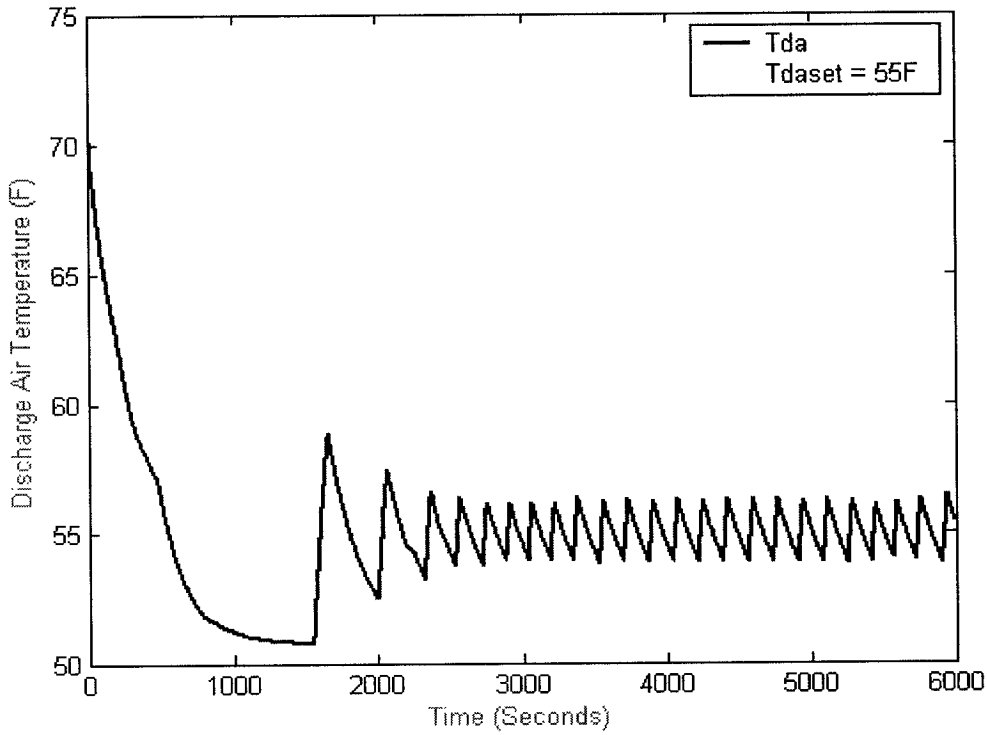


Figure D-7 Result of case 13

Table D.8 Input of case 14

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	4,050	71,960	30,660	70	300	120	2
4,053	6,000	88,790	36,800	70	300	120	2

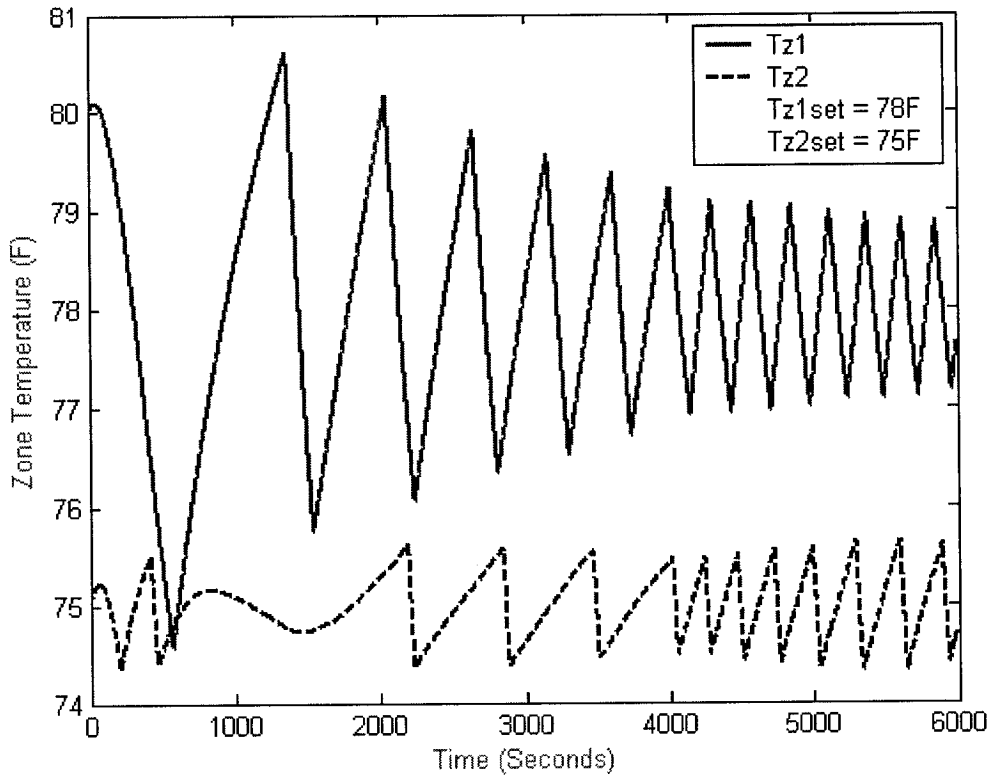


Figure D-8 Result of case 14

➢ **Case 15**

Case 15 simulated a chilled water circulating pump that was experiencing a degradation performance. The input data is listed in **Table D.9**. In this test, when time = 2,532s, the chilled water circulating pump began to experience a faulty operation so that less than required chilled water flow rate entered the cooling coil. As a result, the discharge air temperature was greater than it was supposed to be. As shown in **Fig. D-9**, the discharge air temperature was 60.5F (i.e., T_{da} = 60.5F). It should be noted that the fault did not cause the zone temperatures to drift away from the setpoint values. This was because that both supply air dampers had opened further to compensate for the fault. Accordingly, it is

concluded that the fault – *chilled water circulating pump fault* – may be found due to satisfaction of *Rule 3*.

Table D.9 Input of case 15

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,364	71,960	30,660	90	300	120	1
2,367	6,000	88,790	36,800	90	300	120	1

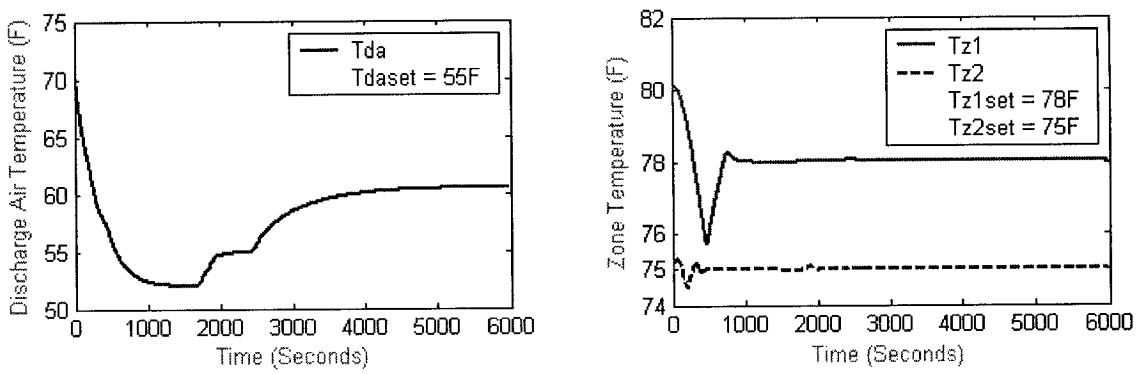


Figure D-9 Result of case 15

➢ **Case 16**

Case 16 simulated an undersized chilled water circulating pump. The input data is listed in **Table D.10**. In this test, an undersized chilled water circulating pump caused less than required chilled water flow rate enter the cooling coil when the design load was acting on the system. As a result, the discharge air temperature was higher that it was supposed to be. As shown in **Fig. D-10**, the discharge air temperature was 57.7F (i.e., T_{da} = 57.7F). It

should also be noted that the fault caused Zone 2 temperature to drift away from the setpoint value. Peak load was acting on Zone 2 so that the supply air damper control signal saturated. Therefore, the fault could not be compensated for Zone 2. Accordingly, it is concluded that the fault – *undersized chilled water circulating pump* – may be detected due to satisfaction of *Rules 3* and *9*.

Table D.10 Input of case 16

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,454	110,390	85,280	90	700	420	1
2,457	6,000	124,220	88,820	90	700	420	1

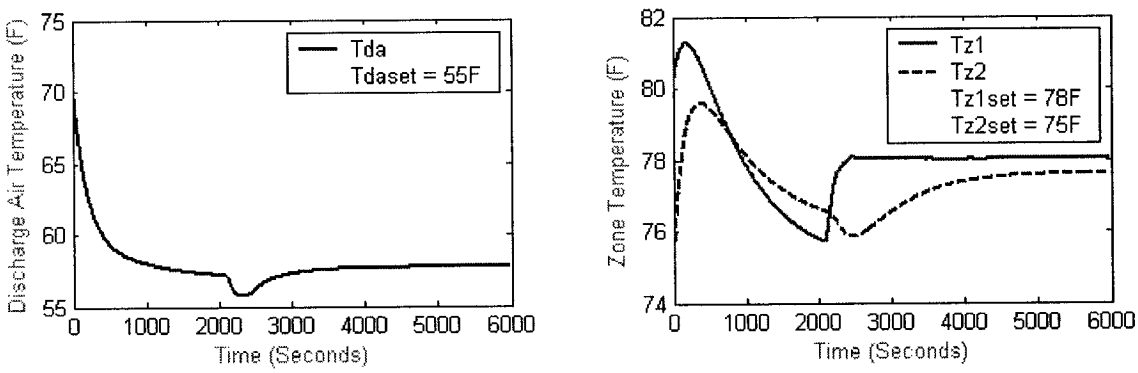


Figure D-10 Result of case 16

➢ **Case 17**

Case 17 simulated the fault that the chilled water supply temperature was too high. The case consisted of two tests.

i. Test 1

The input data is listed in **Table D.11a**. In this test, when time = 1,992s, the chilled water supply temperature became 50F. As a result, the discharge air temperature was higher than it was expected to be. As shown in **Fig. D-11a**, the discharge air temperature was 58.2F (i.e., T_{da} = 58.2F). It should be noted that the fault forced Zone 2 temperature to drift away from the setpoint (i.e., T_{z2} = 78F). Peak load was acting on Zone 2 so the supply air damper saturated. Therefore, the fault was not compensated for Zone 2.

Table D.11a Input of test 1

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,400	110,390	85,280	90	700	420	1
2,403	6,000	124,220	88,820	90	700	420	1

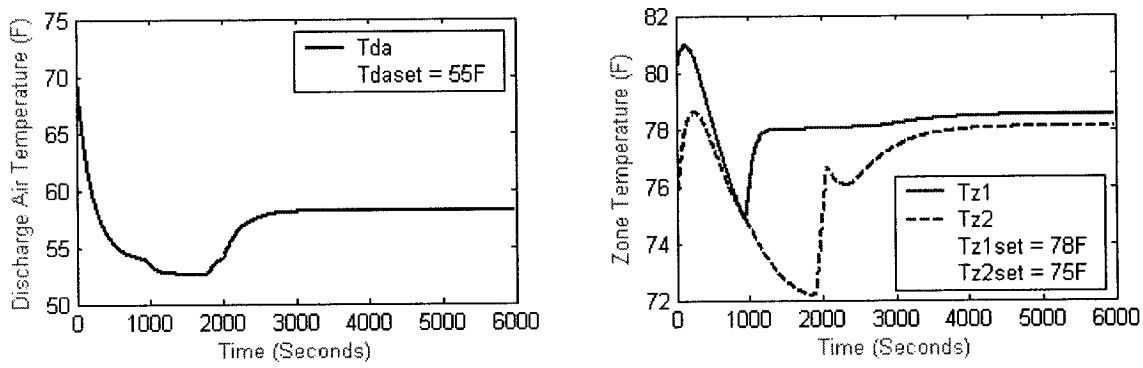


Figure D-11a Result of test 1

ii. Test 2

The input data is listed in **Table D.11b**. In this test, when time = 1,320s, the chilled water supply temperature became 50F. As a result, the discharge air temperature was higher than it was expected to be. As shown in **Fig. D-11b**, the discharge air temperature was 57F (i.e., T_{da} = 57F). It should be noted that the fault did not cause the zone temperatures to drift away from the setpoint values. This was because the VAV box dampers had increased to compensate for the fault.

From the two tests, it is concluded that the fault – *chilled water supply temperature too high* – may be found due to satisfaction of *Rule 3*.

Table D.11b Input of test 2

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,379	71,960	30,660	85	300	120	1
2,382	6,000	88,790	36,800	85	300	120	1

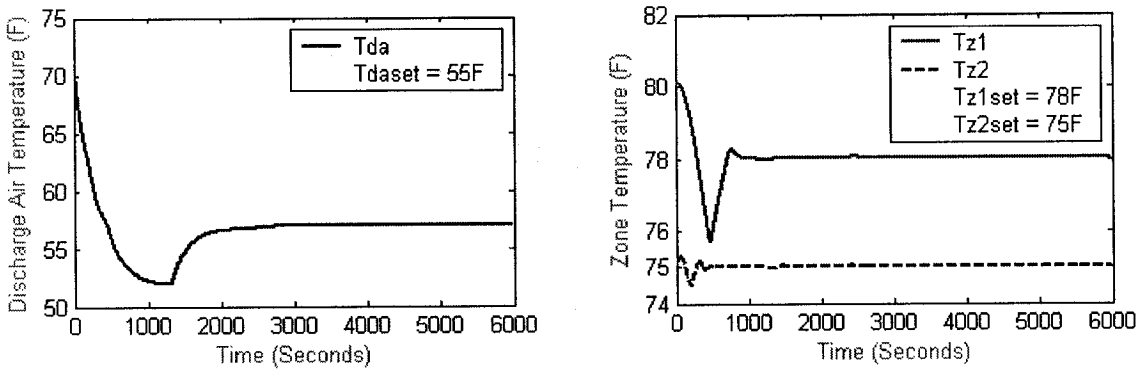


Figure D-11b Result of test 2

➢ **Case 18**

Case 18 simulated the supply fan that was experiencing a performance degradation that might be caused by belt slippage or a decrease in the motor efficiency. The input data is listed in **Table D.12**. In this test, when time = 1,251s, the supply fan began to experience a faulty operation so that less than required airflow rate was supplied. As a result, at least one zone was undercooled. As shown in **Fig. D-12**, Zone 1 temperature was 83F (i.e., T_{z1} = 83F) and Zone 2 temperature was 81.7F (i.e., T_{z2} = 81.7F). Accordingly, it is concluded that the fault – *supply fan fault* - may be detected due to satisfaction of *Rule 9*.

Table D.12 Input of case 18

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,400	134,320	63,750	85	400	220	1
2,403	6,000	139,890	63,450	85	400	220	1

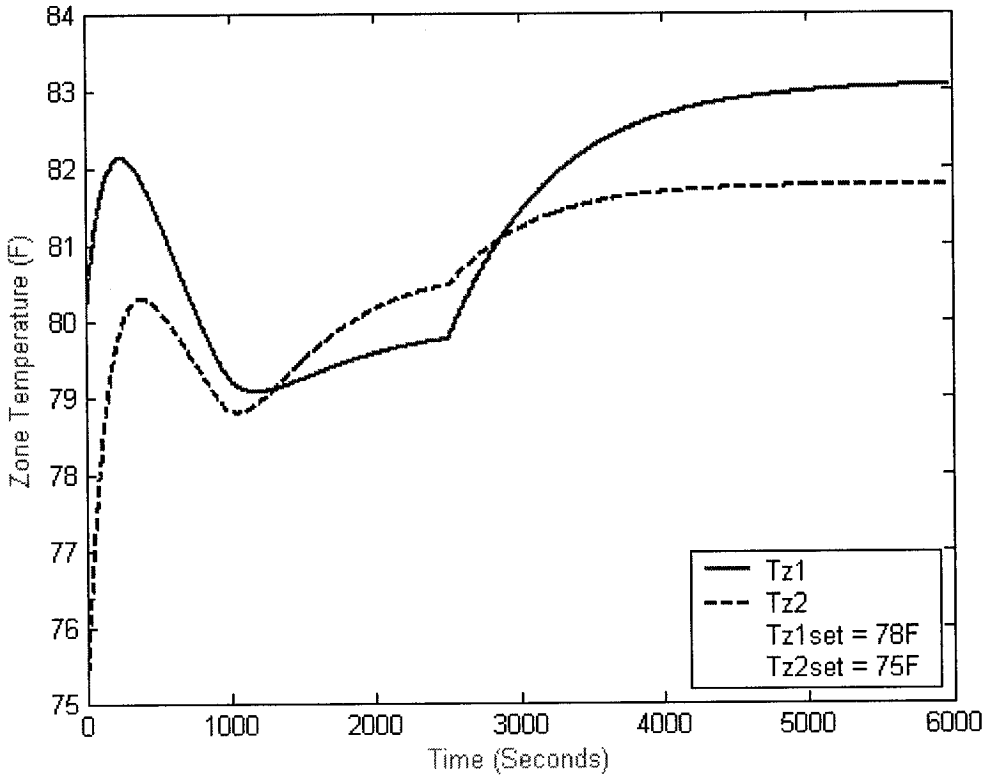


Figure D-12 Result of case 18

➢ **Case 19**

Case 19 simulated an undersized supply fan. The input data is listed in **Table D.13**. In this test, an undersized supply fan caused insufficient supply airflow rate for the design conditions. As a result, at least one zone was undercooled. As shown in **Fig. D-13**, Zone 1 temperature was 84F (i.e., T_{z1} = 84F) and Zone 2 temperature was 83.4F (i.e., T_{z2} = 83.4F). Accordingly, it is concluded that the fault – *undersized supply fan* - may be detected due to satisfaction of *Rule 9*.

Table D.13 Input of case 19

Time (seconds)		Sensible Loads (Btu/hr)		Outdoor Air Temperature (F)	Outdoor Air Intake Required (cfm)		Mode in Effect
Start	End	Zone 1	Zone 2		Zone 1	Zone 2	
0	2,541	110,390	85,280	90	400	120	1
2,544	6,000	124,220	88,820	90	400	120	1

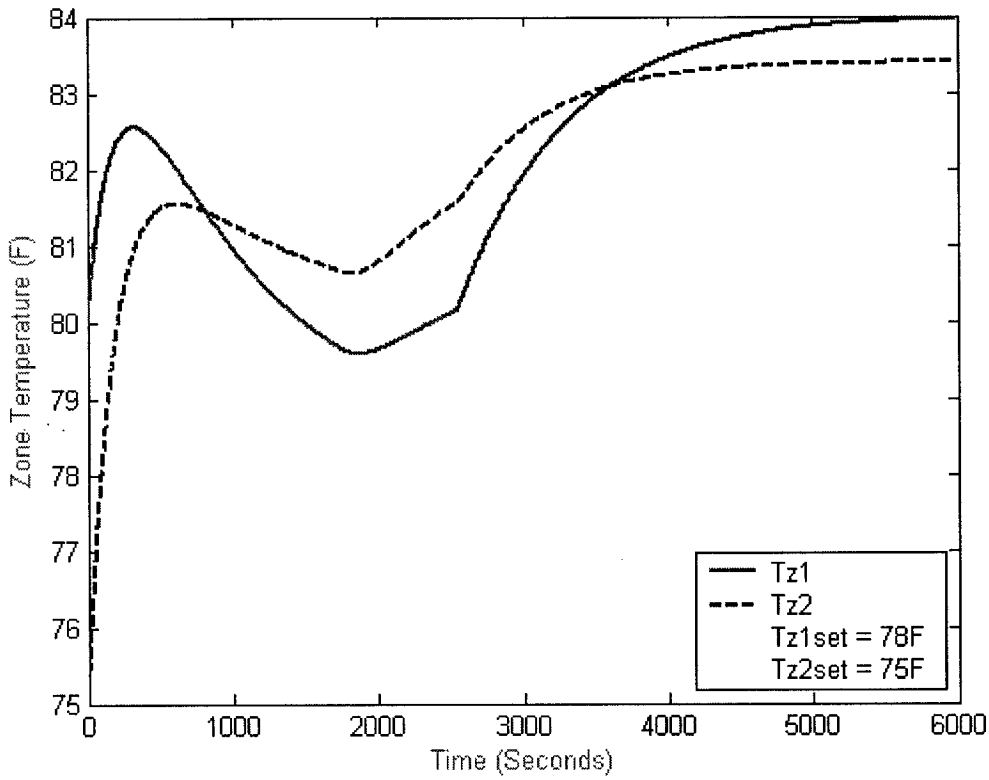


Figure D-13 Result of case 19